Secure Code-Based Key Encapsulation Mechanism with Short Ciphertext and Secret Key

Jayashree Dey and Ratna Dutta

Department of Mathematics, Indian Institute of Technology Kharagpur, Kharagpur-721302, India
deyjayashree@iitkgp.ac.in,ratna@maths.iitkgp.ernet.in

Abstract. Code-based public key cryptosystems are one of the main techniques available in the area of Post-Quantum Cryptography. This work aims to propose a *key encapsulation mechanism* (KEM) with short ciphertext and secret key. Our goal is achieved in two steps. We first present a *public key encryption* (PKE) scheme, basicPKE, using a parity check matrix of *Maximum Distance Separable* (MDS) code as the public key matrix. In our construction, we exploit the structure of a companion matrix to obtain an MDS code which significantly reduces the storage of the secret key. The scheme basicPKE provides security against *Indistinguishability under Chosen Plaintext Attacks* (IND-CPA). Secondly, following the design framework of basicPKE, we construct another PKE scheme, fullPKE, that leads us to design our KEM scheme, fullKEM. We have shown that the scheme fullPKE is secure against *One-Wayness under Plaintext and Validity Checking Attacks* (OW-PCVA) and the scheme fullKEM achieves security against *Indistinguishability under Chosen Ciphertext Attacks* (IND-CCA) in the *random oracle model*. Moreover, our KEM can be shown to accomplish post-quantum security in the quantum random oracle model.

Keywords: Public key encryption · Key encapsulation mechanism · MDS code · Companion matrix.

1 Introduction

The security of widely used classical cryptosystems depends on the hardness of number theory based problems like factorization and the discrete logarithm problem. Due to Shor's algorithm [39], most of these cryptosystems can be broken when sufficiently strong quantum computers become available. Therefore, designing alternatives is essential to survive against quantum attacks while offering reasonable performance with solid security guarantees. Cryptography based on error-correcting codes is one of the main post-quantum techniques as they are usually very fast and can be implemented on several platforms. The security relies on the following two computational assumptions: (i) the hardness of generic decoding which is NP-complete and also believed to be hard even against quantum attackers (ii) the pseudorandomness of the underlying code C for the scheme which says that distinguishing a random matrix from a generator (or parity check) matrix of C used as a part of the public key of the scheme is hard. The important fact in devising code-based cryptosystems is to employ an error-correcting code in such a way that the public key is indistinguishable from a random key. A codeword is used as ciphertext to which random errors are added. The decryptor, knowing a trapdoor, performs

decoding, eliminates the errors and recovers the plaintext. Adversaries are reduced to a generic decoding problem and the system remains safe against a quantum attacker.

In 1978, R. J. McEliece [33] devised the first approach to design a public key encryption scheme using binary Goppa code that has resisted all cryptanalytic attempts so far. It suffers from a large public key size. Later in 1986, Niederreiter [36] proposed a scheme that provides slightly improved efficiency with equivalent security using a parity check matrix as the public key. Several proposals were already endeavored to solve the problem of large key size by replacing binary Goppa codes although they could not last due to their weak security. The scheme McBits [15] is one of the earlier KEM constructions based on the McEliece structure with binary Goppa codes and provides large public key. A number of proposals for KEM are offered to NIST call in 2016 for standardization of quantum-safe cryptography ([10], [42], [14], [30], [1], [2], [3], [4], [9], [7], [40], [41], [5], [35], [6] , [5], [35] , [6], [34]).

Although there have been several secure KEMs based on error-correcting codes, a major concern is still there regarding key size and ciphertext size. The use of the quasi-cyclic and the quasi-dyadic property has been effective to reduce the public key sizes in the schemes ([10], [9]) while the efficient encapsulation techniques in ([10], [2], [14]) offer compact ciphertexts. But, most of the schemes provide a much larger secret key. Therefore, it is essential to search for a way so that the size of the secret key can be greatly reduced.

Our Contribution. In this paper, we aim to devise an IND-CCA secure efficient code-based KEM based on the hardness of the syndrome decoding problem. More specifically, our focus is to design a KEM with relatively short ciphertext and short secret key.

To fulfil the goal of achieving IND-CCA secure KEM, we first design a PKE scheme where we use the structure of a companion matrix to form an MDS code. We employ the Niederreiter framework to design our PKE scheme, basicPKE, using a syndrome as ciphertext and obtain IND-CPA security. Informally, the notion of *Indistinguishability against Chosen Plaintext Attacks* (IND-CPA) requires that no efficient adversary can recognize which of two messages is encrypted in a ciphertext. In our PKE, we use the parity check matrix of the MDS code as the public key matrix. We exploit the structure of the companion matrix to get an MDS code and keep the last row of the companion matrix as secret key which helps to reduce the size of the secret key significantly. The syndrome of a vector is considered as the ciphertext where the vector is set by parsing two vectors – the first vector is an error vector which is generated by a deterministic error vector derivation algorithm and the second vector is constructed from a random vector. This reduces the ciphertext size, making the scheme useful in applications with limited communication bandwidth. Also, utilizing the parity check matrix directly in computing the ciphertext is fast and efficient. For decryption, we form the companion matrix using the secret

key, get a parity check matrix to decode the ciphertext and proceed to recover the message. Note that, MDS codes satisfy the Singleton bound and have the benefits of an efficient decoding algorithm.

Next, following the framework of basicPKE, we design another PKE scheme, fullPKE, which is proven to be OW-PCVA secure in the random oracle model. The notion of *One-Wayness under Plaintext and Validity Checking Attacks* (OW-PCVA) is a non-standard security notion where the adversary can not obtain any information about the encrypted message, having access to plaintext checking oracle and ciphertext validity oracle. In the scheme, the ciphertext is the syndrome of a vector that is formed by concatenating an error vector and a vector which is formed from a hash value of the message. During decryption, the ciphertext is decoded by setting a parity check matrix using the secret key to obtain the message.

Finally, we build fullKEM, a KEM scheme from fullPKE. In the encapsulation phase, we consider the syndrome of a vector as the ciphertext header where the vector is composed by concatenating two vectors. The first vector is an error vector and the second vector is formed from a hash value of a randomly chosen message. In the decapsulation phase, we decode the ciphertext header by forming a parity check matrix from the secret key and then proceed to get the decapsulation key. The scheme fullKEM provides IND-CCA security in random oracle model. Intuitively. the security notion of *Indistinguishability under Chosen Ciphertext Attacks* (IND-CCA) for a KEM claims that no efficient adversary, with access to the decapsulation oracle, can recognize whether the key is randomly chosen or obtained from the encapsulation.

Technical Overview. Let us first discuss our basic techniques in constructing the IND-CPA secure public key encryption scheme $\mathsf{basicPKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$. In Setup, the global public parameters $\mathsf{pp}_{\mathsf{basicPKE}} = (k, k', w, q, m, \gamma)$ are generated and published by a trusted authority taking security parameter λ as input. Here, $q = 2^m$, m being a positive integer. In key generation, a companion matrix associated to the polynomial $g(X) = z_0 + z_1X + z_2X^2 + \cdots + z_{k-1}X_{k-1} + X^k \in \mathsf{GF}(q)[X]$ is utilized by a user to generate a parity check matrix $H \in (\mathsf{GF}(q))^{(n-k)\times n}$ of an $[n, k, k+1]$ MDS code C where $n = 2k$. Then two random permutation matrices $P \in (\mathsf{GF}(q))^{(n-k)\times(n-k)}$ and $Q \in (\mathsf{GF}(q))^{n\times n}$ are selected to compute a parity check matrix $H' = PHQ \in (\mathsf{GF}(q))^{(n-k)\times n}$ of code C' which is equivalent to the MDS code C with parity check matrix H. The matrix H' is then transformed into a matrix $\widehat{H} = [\widehat{M}|I_{(n-k)m}] \in (\mathsf{GF}(2))^{(n-k)m\times nm}$ where $\widehat{M}$ is an $(n-k)m \times km$ matrix. Then the user sets its public key as $\mathsf{pk} = \widehat{M}$ and the secret key as $\mathsf{sk} = (z_0, z_1, \ldots, z_{k-1})$. For encrypting a message $\mathbf{m} \in (\mathsf{GF}(2))^{k'm}$, an encryptor chooses $\mathbf{r}$ randomly from $(\mathsf{GF}(2))^{km}$, parses it as $\mathbf{r} = (\boldsymbol{\rho}||\boldsymbol{\sigma})$ and sets $\boldsymbol{\mu} = (\boldsymbol{\rho}||\mathbf{m}) \in (\mathsf{GF}(2))^{km}$ with $\boldsymbol{\rho} \in (\mathsf{GF}(2))^{(k-k')m}$, $\boldsymbol{\sigma} \in (\mathsf{GF}(2))^{k'm}$. From the public key $\mathsf{pk} = \widehat{M}$, it constructs the matrix $\widehat{H} = (\widehat{M}|I_{(n-k)m})$ which is indistinguishable from a random matrix over $\mathsf{GF}(2)$. Finally, the encryptor computes the ciphertext $\mathbf{c} = \widehat{H}(\mathbf{e}')^T$ with $\mathbf{e}' = (\mathbf{e}||\boldsymbol{\mu})$

where the binary vector $\mathbf{e}$ having length $(n-k)m$ and weight $w-\mathsf{wt}(\boldsymbol{\mu})$ is generated deterministically using $\boldsymbol{\sigma}$ as a seed. In the decryption phase, the ciphertext $\mathbf{c}$ is decoded to get error vector $\mathbf{e}''$ using the decoding technique for MDS codes where the parity check matrix H over $\mathsf{GF}(q)$ is formed by the decryptor using its secret key $\mathsf{sk} = (z_0, z_1, \ldots, z_{k-1})$. Then the message is recovered by parsing the error vector $\mathbf{e}''$ as $\mathbf{e}'' = (\mathbf{e}_0||\boldsymbol{\mu}')$ where $\boldsymbol{\mu}'=(\boldsymbol{\rho}'||\mathbf{m}')$ with $\mathbf{e}_0 \in (\mathsf{GF}(2))^{(n-k)m}$, $\boldsymbol{\rho}' \in (\mathsf{GF}(2))^{(k-k')m}$, $\mathbf{m}' \in (\mathsf{GF}(2))^{k'm}$. The parity check matrix $\widehat{H}$ over $\mathsf{GF}(2)$ derived from the public key pk is a parity check matrix of the MDS code and does not provide any benefits to decode $\mathbf{c}$ as the syndrome decoding problem is hard over $\mathsf{GF}(2)$. To apply decoding procedure for the MDS code, one needs a parity check matrix H over $\mathsf{GF}(q)$ which requires the knowledge of the secret key sk.

Next we present the basic ideas in designing our OW-PCVA secure public key encryption scheme $\mathsf{fullPKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$. In Setup, the global public parameters $\mathsf{pp}_{\mathsf{fullPKE}} = (k, k', w, q, m, \gamma, \mathcal{H}, \mathcal{H}_1)$ are generated by running $\mathsf{basicPKE.Setup}$ and choosing two cryptographically secure hash functions $\mathcal{H} : (\mathsf{GF}(2))^* \longrightarrow (\mathsf{GF}(2))^{km}$ and $\mathcal{H}_1 : (\mathsf{GF}(2))^* \longrightarrow (\mathsf{GF}(2))^{k'm}$ on input a security parameter λ. Here $q = 2^m$ and m is a positive integer. The public key $\mathsf{pk} = \widehat{M}$ and secret key $\mathsf{sk} = (z_0, z_1, \ldots, z_{k-1})$ are obtained by executing the same steps as in $\mathsf{basicPKE.KeyGen}$. In the encryption procedure, an encryptor computes $\mathbf{r} = \mathcal{H}(\mathbf{m}) \in (\mathsf{GF}(2))^{km}, \mathbf{d} = \mathcal{H}_1(\mathbf{m}) \in (\mathsf{GF}(2))^{k'm}$ for a message $\mathbf{m} \in (\mathsf{GF}(2))^{k'm}$, parses it as $\mathbf{r} = (\boldsymbol{\rho}||\boldsymbol{\sigma})$ and sets $\boldsymbol{\mu} = (\boldsymbol{\rho}||\mathbf{m})$ with $\boldsymbol{\rho} \in (\mathsf{GF}(2))^{(k-k')m}$, $\boldsymbol{\sigma} \in (\mathsf{GF}(2))^{k'm}$. The ciphertext component $\mathbf{c}$ is computed as $\mathbf{c} = \widehat{H}(\mathbf{e}')^T$ where $\widehat{H} = (\widehat{M}|I_{(n-k)m})$ is constructed using the public key $\mathsf{pk} = \widehat{M}$, $\mathbf{e}' = (\mathbf{e}||\boldsymbol{\mu})$ and the binary vector $\mathbf{e}$ of length $(n-k)m$ and weight $w - \mathsf{wt}(\boldsymbol{\mu})$ is generated deterministically using $\boldsymbol{\sigma}$ as a seed. The encryptor sets the ciphertext as $\mathsf{CT} = (\mathbf{c}, \mathbf{d})$. In the decryption phase, the ciphertext component $\mathbf{c}$ is decoded by the decryptor to get error vector $\mathbf{e}''$ using the decoding procedure for the MDS code where H is the parity check matrix over $\mathsf{GF}(q)$ formed using the secret key $\mathsf{sk} = (z_0, z_1, \ldots, z_{k-1})$. Then the decryptor parses the vector $\mathbf{e}''$ as $\mathbf{e}'' = (\mathbf{e}_0||\boldsymbol{\mu}')$ where $\boldsymbol{\mu}' = (\boldsymbol{\rho}'||\mathbf{m}')$ with $\mathbf{e}_0 \in (\mathsf{GF}(2))^{(n-k)m}$, $\boldsymbol{\rho}' \in (\mathsf{GF}(2))^{(k-k')m}$, $\mathbf{m}' \in (\mathsf{GF}(2))^{k'm}$ and computes $\mathbf{r}' = \mathcal{H}(\mathbf{m}') = (\boldsymbol{\rho}''||\boldsymbol{\sigma}') \in (\mathsf{GF}(2))^{km}$ and $\mathbf{d}' = \mathcal{H}_1(\mathbf{m}') \in (\mathsf{GF}(2))^{k'm}$ where $\boldsymbol{\rho}'' \in (\mathsf{GF}(2))^{(k-k')m}$, $\boldsymbol{\sigma}' \in (\mathsf{GF}(2))^{k'm}$. A binary error vector $\mathbf{e}'_0$ of length $(n-k)m$ and weight $w - \mathsf{wt}(\boldsymbol{\mu}')$ is then generated in a deterministic way taking $\boldsymbol{\sigma}'$ as seed. The decryptor outputs the symbol $\perp$ if $(\mathbf{e}_0 \neq \mathbf{e}'_0) \vee (\boldsymbol{\rho}' \neq \boldsymbol{\rho}'') \vee (\mathbf{d} \neq \mathbf{d}')$, indicating a decryption failure; otherwise it outputs $\mathbf{m}'$ as the recovered message.

Finally, let us make a primary discussion of our $\mathsf{IND\text{-}CCA}$ secure KEM protocol $\mathsf{fullKEM} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$. In Setup, the global public parameters $\mathsf{pp}_{\mathsf{fullKEM}} = (k, k', w, r, q, m, \gamma, \mathcal{H}, \mathcal{H}_1, \mathcal{H}_2)$ are generated and published by running $\mathsf{fullPKE.Setup}$ algorithm and choosing additionally a cryptographically secure hash function $\mathcal{H}_2 : (\mathsf{GF}(2))^* \longrightarrow (\mathsf{GF}(2))^r$ where λ is a security parameter and the positive integer r is the desired key length. The algorithm $\mathsf{fullKEM.KeyGen}$ follows the same steps as in $\mathsf{fullPKE.KeyGen}$. The encapsulator selects a random message

$\mathbf{m} \in (\mathsf{GF}(2))^{k'm}$, runs the steps as in $\mathsf{fullPKE.Enc}$ and outputs a ciphertext header $\mathsf{CT} = (\mathbf{c}, \mathbf{d})$ along with an encapsulation key $K = \mathcal{H}_2(\mathbf{m})$. On input the public parameters $\mathsf{pp}_{\mathsf{fullKEM}}$, secret key $\mathsf{sk} = (z_0, z_1, \ldots, z_{k-1})$ and ciphertext header CT, the decapsulation procedure outputs the encapsulation key $K = \mathcal{H}_2(\mathbf{m}')$ after obtaining $\mathbf{m}'$ by executing the steps as in $\mathsf{fullPKE.Dec}$.

Table 1. Summary of IND-CCA secure KEMs using random oracles

Scheme	pk size (in bits)	sk size (in bits)	CT size (in bits)	Code used	Cyclic/Dyadic	Correctness error
NTS-KEM [2]	$(n-k)k$	$2(n-k+r)m + nm + r$	$(n-k+r)$	Binary Goppa code	–	No
BIKE-1 [4]	n	$n + w \cdot \lceil \log_2 k \rceil$	n	MDPC code	Quasi-cyclic	Yes
BIKE-2 [4]	k	$n + w \cdot \lceil \log_2 k \rceil$	k	MDPC code	Quasi-cyclic	Yes
BIKE-3 [4]	n	$n + w \cdot \lceil \log_2 k \rceil$	n	MDPC code	Quasi-cyclic	Yes
Classic McEliece [14]	$k(n-k)$	$n + mt + mn$	$(n-k) + r$	Binary Goppa code	–	No
BIG QUAKE [10]	$\frac{k}{\ell}(n-k)$	$mt + mn$	$(n-k) + 2r$	Binary Goppa code	Quasi-cyclic	No
DAGS [8]	$\frac{k}{s}(n-k))\log_2 q$	$2mn \log_2 q$	$[n+k']\log_2 q$	GS code	Quasi-dyadic	No
[18]	$\frac{k}{s}(n-k)\log_2 q$	$2mn \log_2 q$	$[k'+(n-k)]\log_2 q$	GS code	Quasi-dyadic	No
This work	k^2m^2	km	$[k'+k]m$	MDS code	-	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

In Table 1, we provide a theoretical comparison of our KEM construction, fullKEM, with other recently proposed code-based KEMs ([2], [4],[14], [10], [8], [18]). All the schemes use finite fields with characteristic 2. As exhibited in Table 1, we use MDS code in our work, owning the binary structure. The use of the companion matrix helps to reduce the secret key size. In our construction, the secret key size is comparatively shorter than the schemes ([2], [14], [10], [8], [18]). However, the size of the public key remains large. Although the BIKE variants are efficient in terms of key sizes and achieve IND-CCA security, they experience a small decoding failure rate. For suitably chosen parameters, our KEM performs better in terms of ciphertext size over DAGS [8]. Our construction uses parity check matrix which leads faster encapsulation than schemes like DAGS and NTS-KEM. In fact, our encapsulation procedure is closest to the work in [18]. The ciphertext size of our KEM is comparatively better than that of the scheme [18] for suitable parameters, more specifically when k is very less than n.

In Table 1, we mainly highlight the KEMs based on error correcting codes from the class of Alternant codes except BIKE variants which use *quasi-cyclic* (QC) MDPC codes. We keep out the schemes like LEDAkem by Baldi et al. [7], Ramstake by Szepieniec et al. [40], RLCE-KEM by Wang et al. [41], LAKE by Aragon et al.

[5], Ouroboros-R by Melchor et al. [35], LOCKER by Aragon et al. [6], QC-MDPC by Yamada et al. [42], McNie by Kim et al. [30] etc. In fact, the schemes LAKE, Ouroboros-R, LOCKER are based on rank metric codes (Low Rank Parity Check (LRPC) codes) while RLCE-KEM relies on a random linear code and McNie uses any error-correcting code, specially QC-LRPC codes. LEDAkem features a small little decoding failure rate as it uses QC-LDPC codes. Besides, it has risks of a reaction attack by Fabšič et al. [19] for some particular cases. The schemes HQC and RQC suggested by Melchor et al. [1] are also excluded as both the works utilize any decodable linear code. Additionally, HQC has decryption failure issues and RQC is based on rank metric codes. The protocol QC-MDPC has a high risk of GJS attack by Guo et al. [22] as it suffers from a high decoding failure rate for some specific parameters. Another KEM protocol CAKE by Barreto et al. [12] is merged with an independent scheme Ouroboros by Deneuville et al. [17] to get the protocol BIKE.

To prove the security of our KEM, we follow the generic transformations by Hofheinz et al. [27]. We show that our public key encryption scheme basicPKE obtains IND-CPA security under the hardness of syndrome decoding problem and the indistinguishability of the public key matrix from a random matrix. Then we show that breaking OW-PCVA security of fullPKE would lead to breaking the IND-CPA security of basicPKE considering $\mathcal{H}$ as a random oracle. Also, OW-PCVA security always implies OW-VA (One-Wayness under Validity Attacks) 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. Next we show that the OW-VA security of fullPKE implies the IND-CCA security of fullKEM considering $\mathcal{H}_2$ as a random oracle. Therefore, we arrive at the following result.

Theorem 1. *(Informal) Assuming the hardness of decisional syndrome decoding problem and indistinguishability of the public key matrix $\widehat{H}$ (derived from the public key* pk *by running* $\mathsf{fullKEM.KeyGen}(\mathsf{pp}_{\mathsf{fullKEM}})$ *where* $\mathsf{pp}_{\mathsf{fullKEM}} \longleftarrow \mathsf{fullKEM.Setup}(\lambda)$*, λ being the security parameter), our scheme* $\mathsf{fullKEM} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ *provides* IND-CCA *security in the random oracle model.*

We can extend our security proof in the quantum random oracle following the work by Hofheinz et al. [27] and get the following result.

Theorem 2. *(Informal) Assuming the hardness of decisional syndrome decoding problem and indistinguishability of the public key matrix $\widehat{H}$ (derived from the public key* pk *by running* $\mathsf{fullKEM.KeyGen}(\mathsf{pp}_{\mathsf{fullKEM}})$ *where* $\mathsf{pp}_{\mathsf{fullKEM}} \longleftarrow \mathsf{fullKEM.Setup}(\lambda)$*, λ being the security parameter), our scheme* $\mathsf{fullKEM} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ *provides* IND-CCA *security when the hash functions are modeled as quantum random oracles.*

Organization of the Paper. The rest of the paper is organized as follows. In Section 2, we explain the essential background related to our work. We illustrate

our approach to design the IND-CPA secure PKE scheme basicPKE with a discussion of its security in Section 3. In Section 4, we describe the construction and the security of the PKE scheme fullPKE which is required in designing our KEM. We discuss the KEM protocol fullKEM in Section 5. Lastly, we conclude in Section 6.

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 denoted by $U_{c,d}$.

2.1 Public key encryption

Definition 1. (Public Key Encryption) A *public key encryption* (PKE) scheme is a tuple PKE=(Setup, KeyGen, Enc, Dec) of four probabilistic polynomial time algorithms (PPT) with the following specifications.

- $\mathsf{PKE.Setup}(\lambda) \longrightarrow \mathsf{pp}$: A trusted authority runs the Setup algorithm which takes a security parameter λ as input and publishes the global public parameters pp.
- $\mathsf{PKE.KeyGen}(\mathsf{pp}) \longrightarrow (\mathsf{pk}, \mathsf{sk})$: The key generation algorithm, run by a user, takes pp as input and returns a public-secret key pair $(\mathsf{pk}, \mathsf{sk})$. The public key pk is published while the secret key sk is kept secret to the user.
- $\mathsf{PKE.Enc}(\mathsf{pp}, \mathsf{pk}, \mathbf{m}; \mathbf{r}) \longrightarrow \mathsf{CT}$: The encryption algorithm, run by an encryptor, outputs a ciphertext $\mathsf{CT} \in \mathcal{C}$ using a randomness $\mathbf{r} \in \mathcal{R}$ on input the public key pk, a plaintext $\mathbf{m} \in \mathcal{M}$ and the public parameters pp. Here $\mathcal{M}$ is the message space, $\mathcal{C}$ is the ciphertext space and $\mathcal{R}$ is the space of randomness.
- $\mathsf{PKE.Dec}(\mathsf{pp}, \mathsf{sk}, \mathsf{CT}) \longrightarrow \mathbf{m} \vee \perp$: A decryptor runs the decryption algorithm that takes the secret key sk, a ciphertext $\mathsf{CT} \in \mathcal{C}$ and public parameters pp as input and gets either a plaintext $\mathbf{m} \in \mathcal{M}$ or $\perp$ where the symbol $\perp$ indicates the decryption failure.

Correctness. A PKE scheme is δ-correct if for any security parameter λ, $\mathsf{pp} \longleftarrow \mathsf{PKE.Setup}(\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \longleftarrow \mathsf{PKE.KeyGen}(\mathsf{pp})$ and $\mathsf{CT} \longleftarrow \mathsf{PKE.Enc}(\mathsf{pp}, \mathsf{pk}, \mathbf{m}; \mathbf{r})$, it holds that $\Pr[\mathsf{PKE.Dec}(\mathsf{pp}, \mathsf{sk}, \mathsf{CT}) \neq \mathbf{m}] \leq \delta$. The PKE scheme is said to be correct if $\delta = 0$.

Definition 2. (γ-uniformity of PKE [20]). For $\mathbf{m} \in \mathcal{M}$, $\mathsf{pp} \longleftarrow \mathsf{PKE.Setup}(\lambda)$ and $(\mathsf{pk}, \mathsf{sk}) \longleftarrow \mathsf{PKE.KeyGen}(\mathsf{pp})$, a PKE scheme is said to be γ-*uniform* if for every possible ciphertext $\mathsf{CT} \in \mathcal{C}$, $\Pr_{\mathbf{r} \longleftarrow \mathcal{R}}[\mathsf{CT} \longleftarrow \mathsf{PKE.Enc}(\mathsf{pp}, \mathsf{pk}, \mathbf{m}; \mathbf{r})] \leq \gamma$ for a real number γ.

A PKE scheme is said to be γ-spread if it is $2^{-\gamma}$-uniform.

Definition 3. (Indistinguishability under Chosen Plaintext Attack (IND-CPA) [21]). The IND-CPA game between a challenger $\mathcal{S}$ and a PPT adversary $\mathcal{A}$ for a public key encryption scheme PKE=(Setup, KeyGen, Enc, Dec) is described below.

1. The challenger $\mathcal{S}$ generates $\mathsf{pp} \longleftarrow \mathsf{PKE.Setup}(\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \longleftarrow \mathsf{PKE.KeyGen}(\mathsf{pp})$ where λ is a security parameter and sends pp, pk to $\mathcal{A}$.
2. The adversary $\mathcal{A}$ sends a pair of messages $\mathbf{m}_0, \mathbf{m}_1 \in \mathcal{M}$ of the same length to $\mathcal{S}$.
3. The challenger $\mathcal{S}$ picks a random bit $b \in \{0, 1\}$, computes a challenge ciphertext $\mathsf{CT} \longleftarrow \mathsf{PKE.Enc}(\mathsf{pp}, \mathsf{pk}, \mathbf{m}_b; \mathbf{r}_b)$ and sends it to $\mathcal{A}$.
4. The adversary outputs a bit b'.

The adversary $\mathcal{A}$ wins the game if $b' = b$. We define the advantage of $\mathcal{A}$ against the above IND-CPA security game for the PKE scheme as

$$\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{A}) = |\Pr[b' = b] - 1/2|.$$

A PKE scheme is IND-CPA secure if $\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{A})$ is negligible.
We also define the following four security notions for PKE scheme that are (i) One-Wayness under Chosen Plaintext Attacks (OW-CPA), (ii) One-Wayness under Plaintext Checking Attacks (OW-PCA), (iii) One-Wayness under Validity Checking Attacks (OW-VA) and (iv) One-Wayness under Plaintext and Validity Checking Attacks (OW-PCVA).

Definition 4. (OW-ATK [27]). For $\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{PCA}, \mathsf{VA}, \mathsf{PCVA}\}$, the OW-ATK game between a challenger $\mathcal{S}$ and a PPT adversary $\mathcal{A}$ for a public key encryption scheme PKE = (Setup, KeyGen, Enc, Dec) is outlined below where $\mathcal{A}$ can make polynomially many queries to the oracle O_{ATK} given by

$$O_{\mathsf{ATK}} = \begin{cases} - & \mathsf{ATK} = \mathsf{CPA} \\ \mathsf{PCO}(\cdot, \cdot) & \mathsf{ATK} = \mathsf{PCA} \\ \mathsf{CVO}(\cdot) & \mathsf{ATK} = \mathsf{VA} \\ \mathsf{PCO}(\cdot, \cdot), \mathsf{CVO}(\cdot) & \mathsf{ATK} = \mathsf{PCVA} \end{cases}$$

with the Plaintext Checking Oracle $\mathsf{PCO}(\cdot, \cdot)$ and Ciphertext Validity Oracle $\mathsf{CVO}(\cdot)$ as described in Figure 1.

1. The challenger $\mathcal{S}$ generates $\mathsf{pp} \longleftarrow \mathsf{PKE.Setup}(\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \longleftarrow \mathsf{PKE.KeyGen}(\mathsf{pp})$ where λ is a security parameter and sends pp, pk to $\mathcal{A}$.

2. The challenger $\mathcal{S}$ chooses a message $\mathbf{m}^* \in \mathcal{M}$, computes the challenge ciphertext $\mathsf{CT}^* \longleftarrow \mathsf{PKE.Enc}(\mathsf{pp}, \mathsf{pk}, \mathbf{m}^*; \mathbf{r}^*)$ and sends it to $\mathcal{A}$.
3. The adversary $\mathcal{A}$ having access to the oracle O_{ATK}, outputs $\mathbf{m}'$.

The adversary $\mathcal{A}$ wins the game if $\mathbf{m}' = \mathbf{m}^*$. We define the advantage of $\mathcal{A}$ against the above OW-ATK security game for PKE scheme as $\mathsf{Adv}^{\mathsf{OW\text{-}ATK}}_{\mathsf{PKE}}(\mathcal{A}) = \Pr[\mathbf{m}' = \mathbf{m}^*]$. The PKE scheme is said to be OW-ATK secure if $\mathsf{Adv}^{\mathsf{OW\text{-}ATK}}_{\mathsf{PKE}}(\mathcal{A})$ is negligible.

$\mathsf{PCO}(\mathbf{m} \in \mathcal{M}, \mathsf{CT})$	$\mathsf{CVO}(\mathsf{CT} \neq \mathsf{CT}^*)$
1. **if** $\mathsf{PKE.Dec}(\mathsf{pp}, \mathsf{sk}, \mathsf{CT}) \longrightarrow \mathbf{m}$	1. $\mathbf{m} \longleftarrow \mathsf{PKE.Dec}(\mathsf{pp}, \mathsf{sk}, \mathsf{CT})$;
2. **return** 1;	2. **if** $\mathbf{m} \in \mathcal{M}$
3. **else**	3. **return** 1;
4. **return** 0;	4. **else**
5. **end if**	5. **return** 0;
	6. **end if**

Fig. 1. Plaintext Checking Oracle $\mathsf{PCO}(\cdot,\cdot)$ and Ciphertext Validity Oracle $\mathsf{CVO}(\cdot)$ for OW-ATK security game, $\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{PCA}, \mathsf{VA}, \mathsf{PCVA}\}$

Remark 1. [27] For any adversary $\mathcal{B}$ there exists an adversary $\mathcal{A}$ with the same running time as that of $\mathcal{B}$ such that $\mathsf{Adv}^{\mathsf{OW\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{B}) \leq \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{A}) + 1/|\mathcal{M}|$ where $\mathcal{M}$ is the message space.

Remark 2. The OW-PCVA security is also OW-VA security with zero queries to the $\mathsf{PCO}(\cdot,\cdot)$ oracle.

2.2 Key encapsulation mechanism

Definition 5. (Key Encapsulation Mechanism). A *key encapsulation mechanism* (KEM) is a tuple of four PPT algorithms $\mathsf{KEM} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ with the following requirements.

- $\mathsf{KEM.Setup}(\lambda) \longrightarrow \mathsf{pp}$: A trusted authority runs the Setup algorithm which takes a security parameter λ as input and publishes the global public parameters pp.
- $\mathsf{KEM.KeyGen}(\mathsf{pp}) \longrightarrow (\mathsf{pk}, \mathsf{sk})$: The key generation algorithm, run by a user, takes public parameters pp as input and outputs a public-secret key pair $(\mathsf{pk}, \mathsf{sk})$. The public key pk is published while the secret key sk is kept secret to the user.
- $\mathsf{KEM.Encaps}(\mathsf{pp}, \mathsf{pk}) \longrightarrow (\mathsf{CT}, K)$: An encapsulator runs the encapsulation algorithm that takes the public key pk and public parameters pp as input and outputs a ciphertext header $\mathsf{CT} \in \mathcal{C}$ together with a key $K \in \mathcal{K}$. The ciphertext header CT is broadcasted publicly and the encapsulation key K is kept secret to the encapsulator. Here $\mathcal{C}$ is the ciphertext space and $\mathcal{K}$ is the key space.
- $\mathsf{KEM.Decaps}(\mathsf{pp}, \mathsf{sk}, \mathsf{CT}) \longrightarrow K \vee \perp$: A decapsulator runs the decapsulation algorithm on inputs the secret key sk, a ciphertext header CT and public parameters pp. It returns the key K or $\perp$ where $\perp$ is a designated symbol indicating failure.

Correctness. A KEM is δ-correct if for any security parameter λ, $\mathsf{pp} \longleftarrow \mathsf{KEM.Setup}(\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \longleftarrow \mathsf{KEM.KeyGen}(\mathsf{pp})$ and $(\mathsf{CT}, K) \longleftarrow \mathsf{KEM.Encaps}(\mathsf{pp}, \mathsf{pk})$, it holds that $\Pr[\mathsf{KEM.Decaps}(\mathsf{pp}, \mathsf{sk}, \mathsf{CT}) \neq K] \leq \delta$. The KEM is correct if $\delta = 0$.

Definition 6. (Indistinguishability under Chosen Ciphertext Attack (IND-CCA) [38]). The IND-CCA game between a challenger $\mathcal{S}$ and a PPT adversary $\mathcal{A}$ for a key encapsulation mechanism KEM=(Setup, KeyGen, Encaps, Decaps) is described below.

1. The challenger $\mathcal{S}$ generates $\mathsf{pp} \longleftarrow \mathsf{KEM.Setup}(\lambda)$ and $(\mathsf{pk}, \mathsf{sk}) \longleftarrow \mathsf{KEM.KeyGen}(\mathsf{pp})$ where λ is a security parameter and sends pp, pk to $\mathcal{A}$.
2. The PPT adversary $\mathcal{A}$ has access to the decapsulation oracle KEM.Decaps to which $\mathcal{A}$ can make polynomially many ciphertext queries CT_i and gets the corresponding key $K_i \in \mathcal{K}$ from $\mathcal{S}$.
3. The challenger $\mathcal{S}$ chooses a random bit b from $\{0, 1\}$, runs $\mathsf{KEM.Encaps}(\mathsf{pp}, \mathsf{pk})$ to generate a ciphertext-key pair (CT^*, K_0^*) with $\mathsf{CT}^* \neq \mathsf{CT}_i$, selects randomly $K_1^* \in \mathcal{K}$ and sends the pair (CT^*, K_b^*) to $\mathcal{A}$.
4. The adversary $\mathcal{A}$ having the pair (CT^*, K_b^*) keeps submitting polynomially many decapsulation queries on $\mathsf{CT}_i \neq \mathsf{CT}^*$ and finally outputs b'.

The adversary succeeds the game if $b' = b$. We define the advantage of $\mathcal{A}$ against the above IND-CCA security game for the KEM as

$$\mathsf{Adv}_{\mathsf{KEM}}^{\mathsf{IND\text{-}CCA}}(\mathcal{A}) = |\Pr[b' = b] - 1/2|.$$

A KEM is IND-CCA secure if $\mathsf{Adv}_{\mathsf{KEM}}^{\mathsf{IND\text{-}CCA}}(\mathcal{A})$ is negligible.

2.3 MDS codes

Definition 7. (**MDS Code [31]**). 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 8. (**MDS Matrix [26]**). 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 3. ([31]) *An $[n, k, d]$ 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 9. (**Companion Matrix [23]**). 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 4. (**[16]**) *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 results 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.*

Lemma 2. *A block matrix $M \in \mathsf{Mat}(n, m)$ is MDS if and only if its inverse M^{-1} 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 5. **[23]** *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 10. (**Permutation Equivalent Matrices [28]**) 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 3. **[28]** *Suppose that two matrices M and M' are permutation equivalent. Then M is MDS if and only if M' is MDS.*

Definition 11. (**Expanded Codes [29]**). 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 4. ([**29**]). *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 11.*

(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*

$$\widehat{G} = [\phi_n(g_1), \phi_n(\gamma g_1), \ldots, \phi_n(\gamma^{m-1} g_1), \phi_n(g_2), \phi_n(\gamma g_2), \ldots, \phi_n(\gamma^{m-1} g_2), \ldots, \phi_n(g_k), \phi_n(\gamma g_k), \ldots, \phi_n(\gamma^{m-1} g_k)]^T.$$

(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*

$$\widehat{H} = [\phi_{n-k}(h_1)^T, \phi_{n-k}(\gamma h_1)^T, \ldots, \phi_{n-k}(\gamma^{m-1} h_1)^T, \phi_{n-k}(h_2)^T, \phi_{n-k}(\gamma h_2)^T, \ldots, \phi_{n-k}(\gamma^{m-1} h_2)^T, \ldots, \phi_{n-k}(h_n)^T, \phi_{n-k}(\gamma h_n)^T, \ldots, \phi_{n-k}(\gamma^{m-1} h_n)^T].$$

(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* $y \in (GF(q^m))^n$.

2.4 Decoding procedure for MDS codes

Reed-Solomon (RS) and extended Reed-Solomon (RS) codes are the most important classes of MDS codes. RS codes are special cases of Bose Chaudhuri Hocquenghem (BCH) codes and can be decoded by the decoding technique of BCH codes. The decoding procedure of MDS codes is similar to that of BCH codes. In case of RS codes or MDS codes, the designed distance $\delta = d$ where d is the minimum distance of the code. Any $[n, k, d]$ linear code satisfies the Singleton bound $d \leq n - k + 1$ where n is the length of the code and k is the dimension of the code. For MDS codes, the Singleton bound is attained, i.e., $d = n - k + 1$.

• **Decoding of BCH code. [31]** Let C be an $[n, k, d]$ binary BCH code of odd designed distance δ. Suppose the codeword $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1}) \in \mathbb{Z}_2^n$ is transmitted and the vector $\mathbf{y} = \mathbf{c} + \mathbf{e}$ is received where $\mathbf{e} = (e_0, e_1, \ldots, e_{n-1}) \in \mathbb{Z}_2^n$ is the error vector. Let $c(x) = \sum_{i=0}^{n-1} c_i x^i, e(x) = \sum_{i=0}^{n-1} e_i x^i, y(x) = \sum_{i=0}^{n-1} y_i x^i$. Suppose

$$H = \begin{bmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \\ 1 & \alpha^3 & \alpha^6 & \cdots & \alpha^{3(n-1)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & \alpha^{\delta-2} & \alpha^{2(\delta-2)} & \cdots & \alpha^{(\delta-2)(n-1)} \end{bmatrix}$$

be a parity check matrix of the binary BCH code where α is a primitive n-th root of unity. Then, $\alpha, \alpha^3, \ldots, \alpha^{\delta-2}$ are the zeros of BCH code and $c(\alpha^l) = 0$ for $1 \leq l \leq \delta - 1$ as $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1})$ is a codeword. Suppose w errors have occurred in locations $X_1 = \alpha^{i_1}, X_2 = \alpha^{i_2}, \ldots, X_w = \alpha^{i_w}$ and the error values are $Y_1 = e_{i_1}, Y_2 = e_{i_2}, \ldots, Y_w = e_{i_w}$. For binary BCH code, $e_{i_1} = e_{i_2}, = \cdots = e_{i_w} = 1$ and $e_i = 0$ for $i \in \{0, 1, \ldots, n-1\} \setminus \{i_1, i_2, \ldots, i_w\}$. The decoding procedure completes in the following three steps:

Step 1. *Find the syndrome*: The syndrome S is

$$S = Hy^T = \begin{bmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \\ 1 & \alpha^3 & \alpha^6 & \cdots & \alpha^{3(n-1)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & \alpha^{\delta-2} & \alpha^{2(\delta-2)} & \cdots & \alpha^{(\delta-2)(n-1)} \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{i=0}^{n-1} y_i \alpha^i \\ \sum_{i=0}^{n-1} y_i \alpha^{3i} \\ \vdots \\ \sum_{i=0}^{n-1} y_i \alpha^{(\delta-2)i} \end{bmatrix} = \begin{bmatrix} y(\alpha) \\ y(\alpha^3) \\ \vdots \\ y(\alpha^{(\delta-2)}) \end{bmatrix} = \begin{bmatrix} A_1 \\ A_3 \\ \vdots \\ A_{\delta-2} \end{bmatrix}$$

where $A_l = y(\alpha^l)$. Note that $A_{2r} = y(\alpha^{2r}) = y(\alpha^r)^2 = A_r^2$. Alternatively, compute A_l from $y(x)$ as follows using the minimal polynomial $M^{(l)}(x)$ of α^l i.e. $M^{(l)}(x)$ is the lowest degree polynomial over $\mathbb{Z}_2$ having α^l as its zero. Let $y(x) = Q(x)M^{(l)}(x) + R(x), \mathsf{deg}R(x) < \mathsf{deg}M^{(l)}(x)$. Then $A_l = y(\alpha^l) = R(\alpha^l)$ as $M^{(l)}(\alpha^l) = 0, l = 1, 3, \ldots, \delta-2$. Note that $A_2 = A_1^2, A_4 = A_2^2, \ldots, A_{\delta-1} = A_{(\delta-1)/2}^2$ are easily found if needed.

Step 2. *Find the error locator polynomials and error evaluator polynomials*: The error locator polynomial

$$\sigma(z) = \prod_{i=1}^{w}(1 - X_i z) = \sum_{i=0}^{w} \sigma_i z^i, \ \sigma_0 = 1$$

which has reciprocal of error locations $(\frac{1}{X_i}, i = 1, 2, \ldots, w)$ at its zeros. As $c(\alpha^l) = 0$ for $1 \leq l \leq \delta - 1$, we have

$$\begin{aligned} A_l &= y(\alpha^l) \\ &= c(\alpha^l) + e(\alpha^l) \\ &= e(\alpha^l) \\ &= e_{i_1}(\alpha^l)^{i_1} + e_{i_2}(\alpha^l)^{i_2} + \cdots + e_{i_w}(\alpha^l)^{i_w} \\ &= Y_1(\alpha^{i_1})^l + Y_2(\alpha^{i_2})^l + \cdots + Y_w(\alpha^{i_w})^l \\ &= Y_1(X_1)^l + Y_2(X_2)^l + \cdots + Y_w(X_w)^l \\ &= \sum_{i=1}^{w} Y_i X_i^l. \end{aligned}$$

For binary BCH code, we have $Y_l = 1$ for $1 \leq l \leq w$ and so $A_l = \sum_{i=1}^{w} X_i^l$.
Assuming that w errors occurred, the σ_i's and A_l's are related by the recurrence

$$A_{j+w} + \sigma_1 A_{j+w-1} + \sigma_2 A_{j+w-2} + \cdots + \sigma_w A_j = 0 \tag{1}$$

for all j. Taking $j = 1, 2, \ldots, w$ in Equation 1, we get

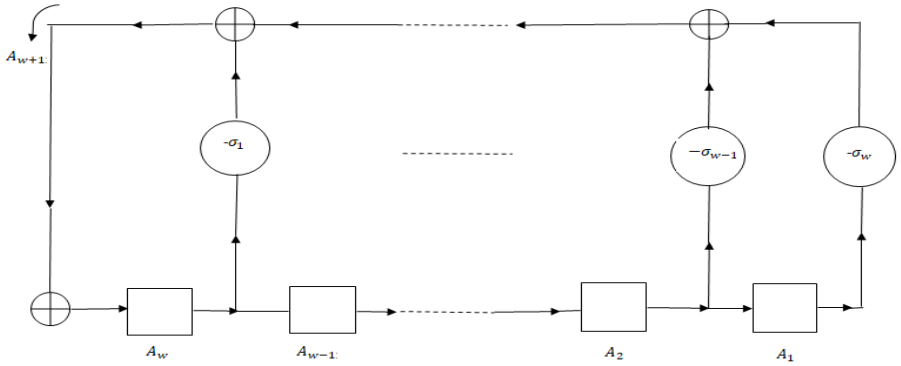
$$\begin{bmatrix} A_w & A_{w-1} & \cdots & A_1 \\ A_{w+1} & A_w & \cdots & A_2 \\ \cdots & \cdots & \cdots & \cdots \\ A_{2w-1} & A_{2w-2} & \cdots & A_w \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \vdots \\ \sigma_w \end{bmatrix} = - \begin{bmatrix} A_{w+1} \\ A_{w+2} \\ \vdots \\ A_{2w} \end{bmatrix}. \tag{2}$$

The recurrence relation in Equation (1) can be interpreted as saying that the A_l's are the output from a *Linear Feedback Shift Register*(LFSR) of w stages with initial contents $A_1, A_2, \ldots, A_w$ as shown in Figure 2. The register is shown at the instant when it contains $A_1, A_2, \ldots, A_w$ and $A_{w+1} = -\sigma_1 A_w - \sigma_2 A_{w-1} - \cdots - \sigma_w A_1$ is being formed.
The decoder's problem is: given the sequence $A_1, A_2, \ldots, A_{\delta-1}$ find that linear feedback shift register of shortest length w which produces $A_1, A_2, \ldots, A_{\delta-1}$ as output when initially loaded with $A_1, A_2, \ldots, A_w$. There is an efficient algorithm due to Berlekamp–Massey [32] for finding such a shift register and hence the error locator polynomial $\sigma(z)$.

Step 3. *Find the locations, values of the errors and correct them*: To find the error locations X_i, compute the reciprocals of the roots of $\sigma(z)$ for $i = 1, 2, \ldots, w$.

- **Decoding non-binary BCH codes**.

Fig. 2. The A_l's are produced by a shift register.



Algorithm 1 Berlekamp–Massey Algorithm

Input: A sequence $\mathbf{s} = \{s_0, s_1, \ldots, s_{n-1}\}$ of n elements of $\mathsf{GF}(q)$.
Output: The feedback polynomial P of an LFSR of length L which generates $\mathbf{s}$.

$P(X) \longleftarrow 1, Q(X) \longleftarrow 1, L \longleftarrow 0, m \longleftarrow -1, d_1 \longleftarrow 1;$
for $(t = 0$ **to** $n-1)$ **do**
  $d \leftarrow s_t + \sum_{i=1}^{L} p_i s_{t-i}$; // p_i's are coefficients of $P(X)$
  if $d \neq 0$ **then**
    $T(X) \longleftarrow P(X);$
    $P(X) \longleftarrow P(X) - d(d_1)^{-1} Q(X) X^{t-m};$
    if $L \leq t/2$ **then**
      $L \longleftarrow t + 1 - L;$
      $m \longleftarrow t;$
      $Q(X) \longleftarrow T(X);$
      $d_1 \longleftarrow d;$
    end if
  end if
end for
return P

1. Find $A_1, A_2, \ldots, A_{\delta-1}$ as in Step 1 above.
2. Use Equation 2 to find the error locator polynomial $\sigma(z)$. The error evaluator polynomial

$$\omega(z) = \sigma(z) + \sum_{\nu=1}^{w} z X_\nu Y_\nu \prod_{j=1, \nu \neq j}^{w} (1 - X_j z)$$

satisfies

$$\omega(z) = (1 + S(z))\sigma(z)$$

where $S(z) = \sum_{i=1}^{\infty} A_i z^i$ and $Y_\nu = e_{i_\nu}$ is the error values for $\nu = 1, 2, \ldots, w$.
3. Finally, find the roots X_γ^{-1} of $\sigma(z)$ for $\gamma = 1, 2, \ldots, w$ and compute the value of errors $Y_\nu = e_{i_\nu}$ using the $\omega(z)$ as $Y_\nu = \frac{\omega(X_\nu^{-1})}{\prod_{\nu \neq j} (1 - X_j X_\nu^{-1})}$.

2.5 Hardness assumptions

Definition 12. ((Search) (q-ary) Syndrome Decoding (SD) Problem [11]). 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 $H\mathbf{e}^T = \mathbf{c}$.

More formally, suppose $\mathcal{D}$ is a PPT algorithm and $U_{(n-k),n}$ is the uniform distribution over $(n-k) \times n$ random q-ary matrices. For every positive integer λ, we define the advantage of $\mathcal{D}$ in solving the SD problem by
$\mathsf{Adv}^{\mathsf{search}}_{\mathcal{D},\mathsf{SD}}(\lambda) = \Pr[\mathcal{D}(H, \mathbf{c}) = \mathbf{e} \in (\mathsf{GF}(q))^n, \mathsf{wt}(\mathbf{e}) = w \text{ and } H\mathbf{e}^T = \mathbf{c} | H \xleftarrow{U} U_{(n-k),n}, \mathbf{c} \in (\mathsf{GF}(q))^n]$.
Also, we define $\mathsf{Adv}^{\mathsf{search}}_{\mathsf{SD}}(\lambda) = \max_{\mathcal{D}}[\mathsf{Adv}^{\mathsf{search}}_{\mathcal{D},\mathsf{SD}}(\lambda)]$ where the maximum is taken over all $\mathcal{D}$. The SD problem is said to be hard if $\mathsf{Adv}^{\mathsf{search}}_{\mathsf{SD}}(\lambda)$ is negligible.
The corresponding decision problem is proven to be NP-complete [13] in case of binary codes. Later, Barg [11] proved that this result holds for codes over all finite fields.

Definition 13. ((Decision) (q-ary) Syndrome Decoding (SD) Problem [11]). Given a full-rank matrix $H_{(n-k)\times n}$ over $\mathsf{GF}(q)$, a vector $\mathbf{e} \in (\mathsf{GF}(q))^n$ and a non-negative integer w, the decision version of q-ary SD problem is to decide whether it is possible to distinguish between a random vector $\mathbf{s} \in (\mathsf{GF}(q))^{n-k}$ from the syndrome $H\mathbf{e}^T$ associated to a w-weight vector $\mathbf{e}$?

Suppose $\mathcal{D}$ is a probabilistic polynomial time algorithm and $U_{(n-k),n}$ be the uniform distribution over $(n-k) \times n$ random q-ary matrices. For every positive integer λ, we define the advantage of $\mathcal{D}$ in solving the decisional SD problem by

$$\mathsf{Adv}^{\mathsf{decision}}_{\mathcal{D},\mathsf{SD}}(\lambda) = |\Pr[\mathcal{D}(H, H\mathbf{e}^T) = 1 \mid \mathbf{e} \in (\mathsf{GF}(q))^n \text{ with } \mathsf{wt}(\mathbf{e}) = w, H \xleftarrow{U} U_{(n-k),n}] - \Pr[\mathcal{D}(H, \mathbf{s}) = 1 \mid \mathbf{s} \xleftarrow{U} U_{(n-k),1}, H \xleftarrow{U} U_{(n-k),n}]|$$

Also, we define $\mathsf{Adv}^{\mathsf{decision}}_{\mathsf{SD}}(\lambda) = \max_{\mathcal{D}}[\mathsf{Adv}^{\mathsf{decision}}_{\mathcal{D},\mathsf{SD}}(\lambda)]$ where the maximum is taken over all $\mathcal{D}$. The decisional SD problem is said to be hard if $\mathsf{Adv}^{\mathsf{decision}}_{\mathsf{SD}}(\lambda)$ is negligible.
By running key generation algorithm, most of the code-based PKE schemes output a public key that is either a generator matrix or a parity check matrix and require the following computational assumption.

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.

Definition 14. (Indistinguishability of public key matrix H [37]). Let $\mathcal{D}$ be a probabilistic polynomial time algorithm and $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public

key encryption scheme that uses an $(n-k)\times n$ matrix H as a public key over $\mathsf{GF}(q)$. For every positive integer λ, we define the advantage of $\mathcal{D}$ in distinguishing the public key matrix H from a random matrix R as

$$\mathsf{Adv}^{\mathsf{IND}}_{\mathcal{D},H}(\lambda) = \Pr[\mathcal{D}(H)=1|(\mathsf{pk}=H,\mathsf{sk}) \longleftarrow \mathsf{PKE.KeyGen}(\mathsf{pp}), \mathsf{pp} \longleftarrow \mathsf{PKE.Setup}(\lambda)] - \Pr[\mathcal{D}(R)=1|R \xleftarrow{U} U_{(n-k),n}]$$

where $U_{(n-k),n}$ is the uniform distribution over $(n-k)\times n$ random q-ary matrices. We define $\mathsf{Adv}^{\mathsf{IND}}_{H}(\lambda) = \max_{\mathcal{D}}[\mathsf{Adv}^{\mathsf{IND}}_{\mathcal{D},H}(\lambda)]$ where the maximum is over all $\mathcal{D}$. The matrix H is said to be *indistinguishable* if $\mathsf{Adv}^{\mathsf{IND}}_{H}(\lambda)$ is negligible.

3 basicPKE : an IND-CPA secure public key encryption

We now present the details of our public key encryption scheme $\mathsf{basicPKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ following the specifications of Definition 1.

- $\mathsf{basicPKE.Setup}(\lambda) \longrightarrow \mathsf{pp}_{\mathsf{basicPKE}}$: 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 $k' \in \mathbb{Z}^+$ with $k' < k$.
(iii) Publish the global parameters $\mathsf{pp}_{\mathsf{basicPKE}} = (k, k', w, q, m, \gamma)$.

- $\mathsf{basicPKE.KeyGen}(\mathsf{pp}_{\mathsf{basicPKE}}) \longrightarrow (\mathsf{pk}, \mathsf{sk})$: A user on input $\mathsf{pp}_{\mathsf{basicPKE}}$, performs the following steps to generate the 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. ([24], [25]).
(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 5 as $g(X)$ satisfies the conditions stated in this theorem. Therefore, every square submatrix of $\widetilde{M}$ is non-singular by Theorem 4. Hence by Theorem 3, 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) Select two permutation matrices P of order $(n-k)\times(n-k)$ and Q of order $n\times n$ and compute $H' = PHQ \in (\mathsf{GF}(q))^{(n-k)\times n}$. Then H and H' are permutation equivalent (see Definition 10). Let C' be the code with parity check matrix H'. Then C' and C are equivalent codes.

(v) 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 $\phi_n : (\mathsf{GF}(2^m))^n \longrightarrow (\mathsf{GF}(2))^{mn}$ defined by

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

and $\phi : \mathsf{GF}(2^m) \longrightarrow (\mathsf{GF}(2))^m$ is given by

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

Here $\widehat{H}$ is an $(n-k)m \times nm$ matrix over $\mathsf{GF}(2)$ by Lemma 4 (ii).

(vi) 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.

Algorithm 2 Error vector derivation

Input: A binary seed vector $\boldsymbol{\sigma}$ of length k, integers n, t.
Output: A binary error vector $\mathbf{e} = (e_0, e_1, \ldots, e_{n-1})$ of length n and weight t.

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

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

(i) Select $\mathbf{r} \xleftarrow{U} (\mathsf{GF}(2))^{km}$. Parse $\mathbf{r}$ as $\mathbf{r} = (\boldsymbol{\rho}||\boldsymbol{\sigma})$ where $\boldsymbol{\rho} \in (\mathsf{GF}(2))^{(k-k')m}$, $\boldsymbol{\sigma} \in (\mathsf{GF}(2))^{k'm}$. Set $\boldsymbol{\mu} = (\boldsymbol{\rho}||\mathbf{m}) \in (\mathsf{GF}(2))^{km}$.

(ii) Run Algorithm 2 to generate a unique binary error vector $\mathbf{e}$ of length $(n-k)m$ and weight $w - \mathsf{wt}(\boldsymbol{\mu})$ using $\boldsymbol{\sigma} \in (\mathsf{GF}(2))^{k'm}$ as a seed. Set $\mathbf{e}' = (\mathbf{e}||\boldsymbol{\mu}) \in (\mathsf{GF}(2))^{nm}$ which has weight w.

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

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

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

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

(i) First proceed as follows to decode $\mathbf{c}$ 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 $\mathbf{c}' = \phi_{n-k}^{-1}(\mathbf{c})$ where $\mathbf{c}$ is a column vector of length $(n-k)m$ over $\mathsf{GF}(2)$, $\mathbf{c}'$ 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 11). The $(n-k) \times n$ parity check matrix H is used to decode $\mathbf{c}$ by first computing the syndrome $S = H(\mathbf{c}'||\mathbf{0})^T$ where $\mathbf{0}$ is a vector consisting of k zeros and then by running the decoding algorithm for MDS codes described in Section 2.4 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) Let $\mathbf{e}'' = (\mathbf{e}_0||\boldsymbol{\mu}') \in (\mathsf{GF}(2))^{nm}$ and $\boldsymbol{\mu}' = (\boldsymbol{\rho}'||\mathbf{m}') \in (\mathsf{GF}(2))^{km}$ where $\mathbf{e}_0 \in (\mathsf{GF}(2))^{(n-k)m}$, $\boldsymbol{\rho}' \in (\mathsf{GF}(2))^{(k-k')m}$, $\mathbf{m}' \in (\mathsf{GF}(2))^{k'm}$.

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

Remark 3. Algorithm 2 uses a hash function $\mathsf{Hsh} : \{0,1\}^* \longrightarrow \{0,1\}^k$ in line 6 and $\mathcal{F}$ in line 4. The subroutine $\mathcal{F}(b) \bmod (n-i-1) \longrightarrow j$ 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 n.

Step 2. Convert this s–bytes string to an integer A.

(a) If $A > 2^{8s} - (2^{8s} \bmod (n-i-1))$ then go to Step 1.

(b) else set $j = A \mod (n-i-1)$.

Correctness: While decoding $\mathbf{c}$, we form an $(n-k) \times n$ parity check matrix H over $\mathsf{GF}(q), q = 2^m$ using the secret key $\mathsf{sk} = (z_0, z_1, \ldots, z_{k-1})$ and find the syndrome

$H(\mathbf{c}'||\mathbf{0})^T$ to estimate the error vector $\mathbf{e}'' \in (\mathsf{GF}(2))^{nm}$ with $\mathsf{wt}(\mathbf{e}'') = w$. Note that, the ciphertext component $\mathbf{c} = \widehat{H}(\mathbf{e}')^T$ is the syndrome of $\mathbf{e}'$ where the matrix $\widehat{H}$ is a parity check matrix in the systemetic form over $\mathsf{GF}(2)$ which is indistinguishable from a random matrix over $\mathsf{GF}(2)$. At the time of decoding $\mathbf{c}$, we need a parity check matrix over $\mathsf{GF}(q), q = 2^m$. The matrix $\widehat{H}$, a parity check matrix of MDS code in the systemetic form derived from the public key pk, does not help to decode $\mathbf{c}$ 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 4 (iv).

Theorem 6. *If the decisional SD problem (Definition 13 in Subsection 2.5) is hard, the public key matrix $\widehat{H}$ (derived from the public key pk which is generated by running $\mathsf{basicPKE.KeyGen}(\mathsf{pp}_{\mathsf{basicPKE}})$ where $\mathsf{pp}_{\mathsf{basicPKE}} \longleftarrow \mathsf{basicPKE.Setup}(\lambda)$) is indistinguishable (Definition 14 in Subsection 2.5), then the public key encryption scheme $\mathsf{basicPKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ described above is $\mathsf{IND\text{-}CPA}$ secure (Definition 3 in Subsection 2.1).*

Proof. In $\mathsf{basicPKE}$, a ciphertext $\mathbf{c}$ is computed using the $(n-k)m \times nm$ public key matrix $\widehat{H} = [\widehat{M}|I_{(n-k)m}]$ that is generated from the public key pk=$\widehat{M}$.
The ciphertext is computed as $\mathbf{c} = \widehat{H}(\mathbf{e}')^T$ where $\mathbf{e}' = (\mathbf{e}||\boldsymbol{\mu}) = (\mathbf{e}||\boldsymbol{\rho}||\mathbf{m}) = (\mathbf{r}_1||\mathbf{m})$ and $\mathbf{m} \in (\mathsf{GF}(2))^{k'm}$, $\mathbf{r}_1 = (\mathbf{e}||\boldsymbol{\rho}) \in (\mathsf{GF}(2))^{(n-k')m}, \boldsymbol{\rho} \in (\mathsf{GF}(2))^{(k-k')m}$ satisfying $\mathbf{r} = (\boldsymbol{\rho}||\boldsymbol{\sigma})$, $\boldsymbol{\sigma} \in (\mathsf{GF}(2))^{k'm}$. Here we use $\boldsymbol{\sigma}$ as a seed to generate the error vector $\mathbf{e} \in (\mathsf{GF}(2))^{(n-k)m}$ satisfying $\mathsf{wt}(\mathbf{e}) = w - \mathsf{wt}(\boldsymbol{\mu})$. Let $\widehat{H} = [H_1|H_2]$ where H_1 is $(n-k)m \times (n-k')m$ sub-matrix and H_2 is $(n-k)m \times k'm$ sub-matrix of $\widehat{H}$. Hence,

$$\mathbf{c} = \widehat{H}[\mathbf{r}_1||\mathbf{m}]^T = H_1\mathbf{r}_1^T + H_2\mathbf{m}^T.$$

Suppose that there exists a PPT algorithm $\mathcal{D}$ such that

$$\begin{aligned}|\Pr[\mathcal{D}(\widehat{H}, H_1\mathbf{r}_1^T) = 1 \mid \mathbf{r}_1 \xleftarrow{U} (\mathsf{GF}(2))^{(n-k')m}, \widehat{H} = [H_1|H_2]] \\ - \Pr[\mathcal{D}(\widehat{H}, \mathbf{s}) = 1 \mid \mathbf{s} \xleftarrow{U} U_{(n-k)m,1}, \widehat{H} = [H_1|H_2]]| \geq \delta\end{aligned}$$

for a small positive δ and the uniform distribution $U_{c,d}$ over $c \times d$ random binary matrices. Let suc be the event that $\mathcal{D}(\widehat{H}, H_1\mathbf{r}_1^T) = 1$ where $\widehat{H}$ is the public key matrix and $\mathbf{r}_1 \xleftarrow{U} (\mathsf{GF}(2))^{(n-k')m}$. Then we construct an adversary $\mathcal{D}'$ (see Figure 3) which distinguishes a random matrix from $\widehat{H}$. The adversary $\mathcal{D}'$ takes as input a matrix R. Let $\mathsf{E}_{\mathsf{ran}}$ be the event that the matrix R is chosen randomly from uniform distribution $U_{(n-k)m,nm}$ and E_{re} be the event that R be the public key matrix $\widehat{H}$ constructed as

$\underline{\mathcal{D}'(R)}$

1. $R_{(n-k)m\times nm} = [(R_1)_{(n-k)m\times(n-k')m}|(R_2)_{(n-k)m\times k'm}]$;
2. $p \xleftarrow{U} \{0,1\}$;
3. **if** $p = 1$
4. $\quad \mathbf{s}_1 = R_1\mathbf{r}_1^T$;
5. $\quad p' \longleftarrow \mathcal{D}(R, \mathbf{s}_1)$;
6. **else**
7. $\quad \mathbf{s}_0 \xleftarrow{U} U_{(n-k)m,1}$;
8. $\quad p' \longleftarrow \mathcal{D}(R, \mathbf{s}_0)$;
9. **end if**
10. **if** $p = p'$
11. $\quad$ **return** 1;
12. **else**
13. $\quad$ **return** 0;
14. **end if**

Fig. 3. Adversary $\mathcal{D}'$ in the proof of Theorem 6

stated above using the public key pk generated by $\mathsf{basicPKE.KeyGen}(\mathsf{pp}_{\mathsf{basicPKE}})$. Then

$$\begin{aligned}
&|\Pr[p = p'|\mathsf{E}_{\mathsf{re}}] - \Pr[p = p'|\mathsf{E}_{\mathsf{ran}}]| \\
=&|\Pr[\mathcal{D}'(\widehat{H}) = 1|\text{public key matrix } \widehat{H} = [\widehat{M}|I_{(n-k)m}] \in (\mathsf{GF}(2))^{(n-k)m\times nm}] \\
&\qquad - \Pr[\mathcal{D}'(R) = 1|R \xleftarrow{U} U_{(n-k)m,nm}]| \\
=&\mathsf{Adv}^{\mathsf{IND}}_{\mathcal{D}',\widehat{H}}(\lambda) \leq \mathsf{Adv}^{\mathsf{IND}}_{\widehat{H}}(\lambda) \text{ (see Definition 14).}
\end{aligned}$$

When the event E_{re} occurs, we have $R = \widehat{H}$. Since $\mathcal{D}'$ outputs 1 if and only if $\mathcal{D}$ succeeds, we have $\Pr[p = p'|\mathsf{E}_{\mathsf{re}}] = \Pr[\mathsf{suc}]$. When $\mathsf{E}_{\mathsf{ran}}$ occurs, the matrix R was chosen randomly from uniform distribution $U_{(n-k)m,nm}$. Therefore,

$$\begin{aligned}
&|\Pr[p = p'|\mathsf{E}_{\mathsf{ran}}] - 1/2| \\
=&|\Pr[\mathcal{D}'(R) = 1|R \xleftarrow{U} U_{(n-k)m,nm}] - 1/2| \\
=&|\Pr[\mathcal{D}(R, \mathbf{s}_1) = 1|R \xleftarrow{U} U_{(n-k)m,nm}] - \Pr[\mathcal{D}(R, \mathbf{s}_0) = 1|R \xleftarrow{U} U_{(n-k)m,nm}]| \\
=&\mathsf{Adv}^{\mathsf{decision}}_{\mathcal{D},\mathsf{SD}}(\lambda) \leq \mathsf{Adv}^{\mathsf{decision}}_{\mathsf{SD}}(\lambda)
\end{aligned}$$

as $\Pr[\mathcal{D}(R, \mathbf{s}_0) = 1|R \xleftarrow{U} U_{(n-k)m,nm}] = 1/2$. Note that when the algorithm $\mathcal{D}$ takes uniformly distributed inputs, it outputs 1 with probability 1/2, i.e. $\Pr[\mathcal{D}(\widehat{H}, \mathbf{s}) = 1 \mid \mathbf{s} \xleftarrow{U} U_{(n-k)m,1}, \widehat{H} = [H_1|H_2]] = 1/2$. Combining all the probabilities together,

we have

$$\begin{aligned}
\delta &\leq |\Pr[\mathcal{D}(\widehat{H}, H_1\mathbf{r}_1^T) = 1 \mid \mathbf{r}_1 \xleftarrow{U} (\mathsf{GF}(2))^{(n-k')m}, \widehat{H} = [H_1|H_2]] \\
&\qquad - \Pr[\mathcal{D}(\widehat{H}, \mathbf{s}) = 1 \mid \mathbf{s} \xleftarrow{U} U_{(n-k)m,1}, \widehat{H} = [H_1|H_2]]| \\
&= |\Pr[\mathsf{suc}] - 1/2| \\
&= |\Pr[p = p'|\mathsf{E_{re}}] - 1/2| \\
&= |\Pr[p = p'|\mathsf{E_{re}}] - \Pr[p = p'|\mathsf{E_{ran}}] + \Pr[p = p'|\mathsf{E_{ran}}] - 1/2| \\
&\leq \mathsf{Adv}^{\mathsf{IND}}_{\widehat{H}}(\lambda) + |\Pr[p = p'|\mathsf{E_{ran}}] - 1/2| \\
&\leq \mathsf{Adv}^{\mathsf{decision}}_{\mathsf{SD}}(\lambda) + \mathsf{Adv}^{\mathsf{IND}}_{\widehat{H}}(\lambda).
\end{aligned}$$

This yields a contradiction since decisional SD problem is hard and $\widehat{H}$ is indistinguishable. Hence,

$$\begin{aligned}
&|\Pr[\mathcal{D}(\widehat{H}, H_1\mathbf{r}_1^T) = 1 \mid \mathbf{r}_1 \xleftarrow{U} (\mathsf{GF}(2))^{(n-k')m}, \widehat{H} = [H_1|H_2]] \\
&\qquad - \Pr[\mathcal{D}(\widehat{H}, \mathbf{s}) = 1 \mid \mathbf{s} \xleftarrow{U} U_{(n-k)m,1}, \widehat{H} = [H_1|H_2]]| \leq \delta \qquad (1)
\end{aligned}$$

Now, we construct a distinguisher $\mathcal{B}$ from an IND-CPA adversary $\mathcal{A}$ against the scheme $\mathsf{basicPKE}$ as in Figure 4 where $\mathcal{B}$ distinguishes $\tilde{\mathbf{s}}_1 = H_1\mathbf{r}_1^T$ from the same length random value $\tilde{\mathbf{s}}_0$ where $\mathbf{r}_1 \xleftarrow{U} (\mathsf{GF}(2))^{(n-k')m}$. In Figure 4, H_2 is extracted

$\underline{\mathcal{B}(\mathsf{pp}_{\mathsf{basicPKE}}, \mathsf{pk}, \tilde{\mathbf{s}})}$

1. $(\mathbf{m}_0, \mathbf{m}_1) \longleftarrow \mathcal{A}(\mathsf{pp}_{\mathsf{basicPKE}}, \mathsf{pk})$;
2. $b \xleftarrow{U} \{0, 1\}$;
3. $\mathbf{c} = \tilde{\mathbf{s}} + H_2\mathbf{m}_b^T$;
4. $b' \longleftarrow \mathcal{A}(\mathbf{c})$;
5. **if** $b = b'$
6. **return** 1;
7. **else**
8. **return** 0;
9. **end if**

Fig. 4. A distinguisher $\mathcal{B}$ from the IND-CPA adversary $\mathcal{A}$ in the proof of Theorem 6

by $\mathcal{B}$ from $\widehat{H} = [H_1|H_2]$ which is derived from $\mathsf{pk} = \widehat{M}$. Let RAN be the event that $\tilde{\mathbf{s}}$ $(= \tilde{\mathbf{s}}_0)$ was chosen from the uniform distribution $U_{(n-k)m,1}$ while RE be the event that $\tilde{\mathbf{s}}$ $(= \tilde{\mathbf{s}}_1)$ is $H_1\mathbf{r}_1^T$. We will say that $\mathcal{A}$ succeeds if $b = b'$ when the event RE occurs and we call this event WIN. We get

$$|\Pr[\mathcal{B}(\mathsf{pp}_{\mathsf{basicPKE}}, \mathsf{pk}, \tilde{\mathbf{s}}) = 1|\mathsf{RE}] - \Pr[\mathcal{B}(\mathsf{pp}_{\mathsf{basicPKE}}, \mathsf{pk}, \tilde{\mathbf{s}}) = 1|\mathsf{RAN}]| \leq \mathsf{Adv}^{\mathsf{IND}}_{H_1\mathbf{r}_1^T}(\lambda). \qquad (2)$$

Note that when event RE occurs, we have $\tilde{\mathbf{s}} = \tilde{\mathbf{s}}_1 = H_1\mathbf{r}_1^T$ which is distributed exactly as in the real execution as $\mathbf{c} = \tilde{\mathbf{s}} + H_2\mathbf{m}_b^T = H_1\mathbf{r}_1^T + H_2\mathbf{m}_b^T = \widehat{H}[\mathbf{r}_1||\mathbf{m}_b]^T$. Since $\mathcal{B}$ outputs 1 if and only if $\mathcal{A}$ succeeds, we have $\Pr[\mathcal{B}(\mathsf{pp}_{\mathsf{basicPKE}}, \mathsf{pk}, \tilde{\mathbf{s}}) = 1|\mathsf{RE}] = \Pr[\mathsf{WIN}]$. On the other hand, when event RAN occurs, $\tilde{\mathbf{s}}$ is distributed uniformly. Therefore, $\tilde{\mathbf{s}} + H_2\mathbf{m}_b^T$ given to $\mathcal{A}$ is uniformly distributed as well. This means that $\mathcal{A}$ obtains no information related to b. Since $\mathcal{B}$ outputs 1 if and only if $\mathcal{A}$ succeeds, we can conclude that $\Pr[\mathcal{B}(\mathsf{pp}_{\mathsf{basicPKE}}, \mathsf{pk}, \tilde{\mathbf{s}}) = 1|\mathsf{RAN}] = 1/2$. combining these results, we obtain

$$\begin{aligned} &|\Pr[\mathcal{B}(\mathsf{pp}_{\mathsf{basicPKE}}, \mathsf{pk}, \tilde{\mathbf{s}}) = 1|\mathsf{RE}] - \Pr[\mathcal{B}(\mathsf{pp}_{\mathsf{basicPKE}}, \mathsf{pk}, \tilde{\mathbf{s}}) = 1|\mathsf{RAN}]| \\ =&|\Pr[\mathsf{WIN}] - 1/2| = \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{basicPKE}}(\mathcal{A}). \end{aligned} \tag{3}$$

From equations (2) and (3), we can say that the distinguisher $\mathcal{B}$ distinguishes $\tilde{\mathbf{s}}_1 = H_1 r_1^T$ from the random value $\tilde{\mathbf{s}}_0$ of the same length with non-negligible probability if the adversary $\mathcal{A}$ breaks the IND-CPA security with non-negligible probability. More specifically, if $\mathsf{Adv}^{\mathsf{IND}}_{H_1\mathbf{r}_1^T}(\lambda) \leq \delta$, then $\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{basicPKE}}(\mathcal{A}) \leq \delta$. From Equation (1), we can conclude that the scheme $\mathsf{basicPKE}$ is IND-CPA secure if the decisional SD problem is hard and the public key matrix $\widehat{H}$ is indistinguishable. ■

4 fullPKE: an OW-PCVA secure public key encryption

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

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

(i) Compute $\mathbf{r} = \mathcal{H}(\mathbf{m}) \in (\mathsf{GF}(2))^{km}$, $\mathbf{d} = \mathcal{H}_1(\mathbf{m}) \in (\mathsf{GF}(2))^{k'm}$.
(ii) Parse $\mathbf{r} = (\boldsymbol{\rho}||\boldsymbol{\sigma})$ where $\boldsymbol{\rho} \in (\mathsf{GF}(2))^{(k-k')m}$, $\boldsymbol{\sigma} \in (\mathsf{GF}(2))^{k'm}$.
(iii) Set $\boldsymbol{\mu} = (\boldsymbol{\rho}||\mathbf{m}) \in (\mathsf{GF}(2))^{km}$.
(iv) Run Algorithm 2 using $\boldsymbol{\sigma}$ as a seed to obtain an error vector $\mathbf{e}$ of length $(n-k)m$ and weight $w - \mathsf{wt}(\boldsymbol{\mu})$ and set $\mathbf{e}' = (\mathbf{e}||\boldsymbol{\mu}) \in (\mathsf{GF}(2))^{nm}$.
(v) Use the public key $\mathsf{pk} = \widehat{M}$ to construct the matrix $\widehat{H} = (\widehat{M}|I_{(n-k)m})$.
(vi) Compute $\mathbf{c} = \widehat{H}(\mathbf{e}')^T$.
(vii) Return the ciphertext $\mathsf{CT} = (\mathbf{c}, \mathbf{d}) \in \mathcal{C} = (\mathsf{GF}(2))^{(k+k')m}$.

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

(i) Use the secret key $\mathsf{sk} = (z_0, z_1, \ldots, z_{k-1})$ to form a parity check matrix H and then find error vector $\mathbf{e}''$ of weight w and length nm as in the procedure $\mathsf{basicPKE.Dec}$.
(ii) Parse $\mathbf{e}'' = (\mathbf{e}_0||\boldsymbol{\mu}') \in (\mathsf{GF}(2))^{nm}$ and $\boldsymbol{\mu}'=(\boldsymbol{\rho}'||\mathbf{m}') \in (\mathsf{GF}(2))^{km}$ where $\mathbf{e_0} \in (\mathsf{GF}(2))^{(n-k)m}$, $\boldsymbol{\rho}' \in (\mathsf{GF}(2))^{(k-k')m}$, $\mathbf{m}' \in (\mathsf{GF}(2))^{k'm}$.
(iii) Compute $\mathbf{r}' = \mathcal{H}(\mathbf{m}') \in (\mathsf{GF}(2))^{km}$ and $\mathbf{d}' = \mathcal{H}_1(\mathbf{m}') \in (\mathsf{GF}(2))^{k'm}$.
(iv) Parse $\mathbf{r}' = (\boldsymbol{\rho}''||\boldsymbol{\sigma}')$ where $\boldsymbol{\rho}'' \in (\mathsf{GF}(2))^{(k-k')m}$, $\boldsymbol{\sigma}' \in (\mathsf{GF}(2))^{k'm}$.
(v) Generate error vector $\mathbf{e}'_0$ of length $(n-k)m$ and weight $w - \mathsf{wt}(\boldsymbol{\mu}')$ by running Algorithm 2 with $\boldsymbol{\sigma}'$ as seed.
(vi) If $(\mathbf{e}_0 \neq \mathbf{e}'_0) \vee (\boldsymbol{\rho}' \neq \boldsymbol{\rho}'') \vee (\mathbf{d} \neq \mathbf{d}')$, output $\perp$ that indicates decryption failure. Otherwise, return $\mathbf{m}'$.

Correctness. The decoding algorithm in $\mathsf{fullPKE.Dec}$ 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 $\mathsf{fullPKE.Enc}$ satisfies $\mathsf{wt}(\mathbf{e}') = w \leq k/2$. Consequently, the decoding procedure will recover the correct $\mathbf{e}'$ as Lemma 4 (iv) holds. We regenerate $\mathbf{e}'_0$ and $\boldsymbol{\rho}''$ and compare it with $\mathbf{e}_0$ and $\boldsymbol{\rho}'$ obtained after decoding. Since the error vector generation uses a deterministic function to get a fixed low weight error vector, $\mathbf{e}_0 = \mathbf{e}'_0$ and $\boldsymbol{\rho}' = \boldsymbol{\rho}''$ occurs.

Theorem 7. *If the public key encryption scheme* $\mathsf{basicPKE}$ =(Setup, KeyGen, Enc, Dec) *as described in Section 3 is* $\mathsf{IND\text{-}CPA}$ *secure (Definition 3 in Subsection 2.1), then the public key encryption scheme* $\mathsf{fullPKE}$ = ($\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}$) *as described above provides* $\mathsf{OW\text{-}PCVA}$ *security (Definition 4 in Subsection 2.1) when the hash function* $\mathcal{H}$ *is modeled as a random oracle.*

Proof. Let $\mathcal{A}$ be a PPT adversary against the $\mathsf{OW\text{-}PCVA}$ security of the encryption scheme $\mathsf{fullPKE}$ with at most $n_{\mathcal{H}}$ queries to the hash oracle $\mathcal{H}$, n_P queries to the oracle PCO (Figure 1) and n_V queries to oracle CVO (Figure 1). We show that a PPT adversary against the IND-CPA security of the scheme $\mathsf{basicPKE}$ can be constructed. First we define the sequence of games $\mathbf{G}_j, j = 0, 1, 2, 3, 4$ in Figure 5 and Figure 6. The view of the PPT adversary $\mathcal{A}$ is shown to be computationally indistinguishable in any of the consecutive games. Let E_j be the event that $\mathbf{m}' = \mathbf{m}^*$ in game $\mathbf{G}_j$, $j = 0, 1, 2, 3, 4$.

Game $\mathbf{G}_0$: Game $\mathbf{G}_0$ (Figure 5) is the standard $\mathsf{OW\text{-}PCVA}$ game (Definition 4). So we have

$$\Pr[E_0] = \mathsf{Adv}^{\mathsf{OW\text{-}PCVA}}_{\mathsf{fullPKE}}(\mathcal{A}).$$

Game $\mathbf{G}_1$: In the game $\mathbf{G}_1$, $\mathbf{c}^*$ is computed by running $\mathsf{basicPKE.Enc}(\mathsf{pp}_{\mathsf{basicPKE}}, \mathsf{pk}, \mathbf{m}^*; \mathbf{r}^*)$ for the message $\mathbf{m}^*$ and the queries to the oracles PCO and CVO are answered as in Figure 7. When a query $(\mathbf{m}, \mathbf{c})$ is submitted to the oracle PCO, the

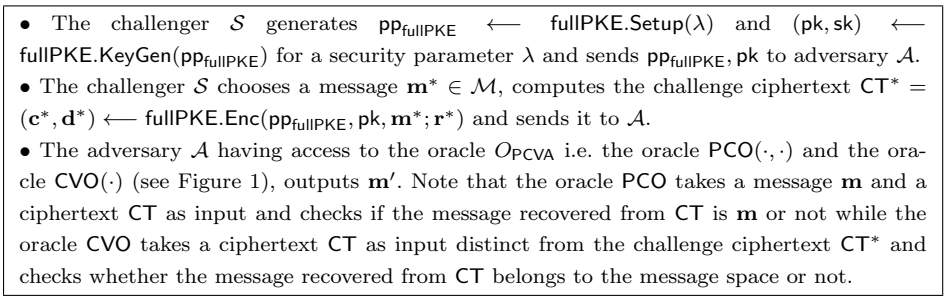


Fig. 5. Game $\mathbf{G}_0$ in the proof of the Theorem 7

challenger $\mathcal{S}$ decrypts $\mathbf{c}$ to recover $\mathbf{m}' \longleftarrow \mathsf{basicPKE.Dec}(\mathsf{pp}_{\mathsf{basicPKE}}, \mathsf{sk}, \mathbf{c})$ and returns 1 if $\mathbf{m}' = \mathbf{m}$ with $\mathsf{basicPKE.Enc}(\mathsf{pp}_{\mathsf{basicPKE}}, \mathsf{pk}, \mathbf{m}'; \mathcal{H}(\mathbf{m}')) \longrightarrow \mathbf{c}$. The condition $\mathsf{basicPKE.Enc}(\mathsf{pp}_{\mathsf{basicPKE}}, \mathsf{pk}, \mathbf{m}'; \mathcal{H}(\mathbf{m}')) \longrightarrow \mathbf{c}$ actually addresses the checking steps (ii)-(iv) (described in Section 4) in $\mathsf{fullPKE.Dec}(\mathsf{pp}_{\mathsf{fullPKE}}, \mathsf{sk}, \mathsf{CT} = (\mathbf{c}, \mathbf{d}))$. On query $\mathbf{c} \neq \mathbf{c}^*$ to the oracle CVO, the challenger computes $\mathbf{m}' \longleftarrow \mathsf{basicPKE.Dec}(\mathsf{pp}_{\mathsf{basicPKE}}, \mathsf{sk}, \mathbf{c})$ and returns 1 if $\mathsf{basicPKE.Enc}(\mathsf{pp}_{\mathsf{basicPKE}}, \mathsf{pk}, \mathbf{m}'; \mathcal{H}(\mathbf{m}')) \longrightarrow \mathbf{c}$ with $\mathbf{m}' \in \mathcal{M}$. The challenger $\mathcal{S}$ maintains hash list $Q_{\mathcal{H}}$ (initially empty) to record random oracle queries made to the hash oracle $\mathcal{H}$ with the convention that $\mathbf{r} = \mathcal{H}(\mathbf{m})$ if and only if $(\mathbf{m}, \mathbf{r}) \in Q_{\mathcal{H}}$. Note that both the games $\mathbf{G}_0$ and $\mathbf{G}_1$ proceed identically. Therefore, we get

$$\Pr[E_0] = \Pr[E_1].$$

Game $\mathbf{G}_2$: In game $\mathbf{G}_2$, a query $\mathbf{c} \neq \mathbf{c}^*$ to the CVO oracle is responded by first

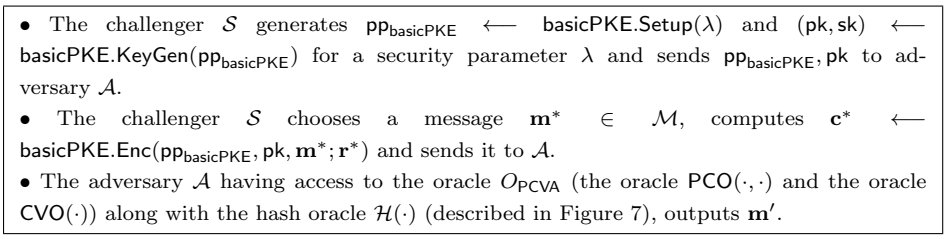


Fig. 6. Sequence of games $\mathbf{G}_j, j = 1, 2, 3, 4$ in the proof of the Theorem 7

decrypting $\mathbf{c}$ as with one which computes $\mathbf{m}' \longleftarrow \mathsf{basicPKE.Dec}(\mathsf{pp}_{\mathsf{basicPKE}}, \mathsf{sk}, \mathbf{c})$ and returning 1 if there exists a previous record $(\mathbf{m}, \mathbf{r}) \in Q_{\mathcal{H}}$ with $\mathsf{basicPKE.Enc}$ $(\mathsf{pp}_{\mathsf{basicPKE}}, \mathsf{pk}, \mathbf{m}; \mathbf{r}) \longrightarrow \mathbf{c}$ and $\mathbf{m} = \mathbf{m}'$.

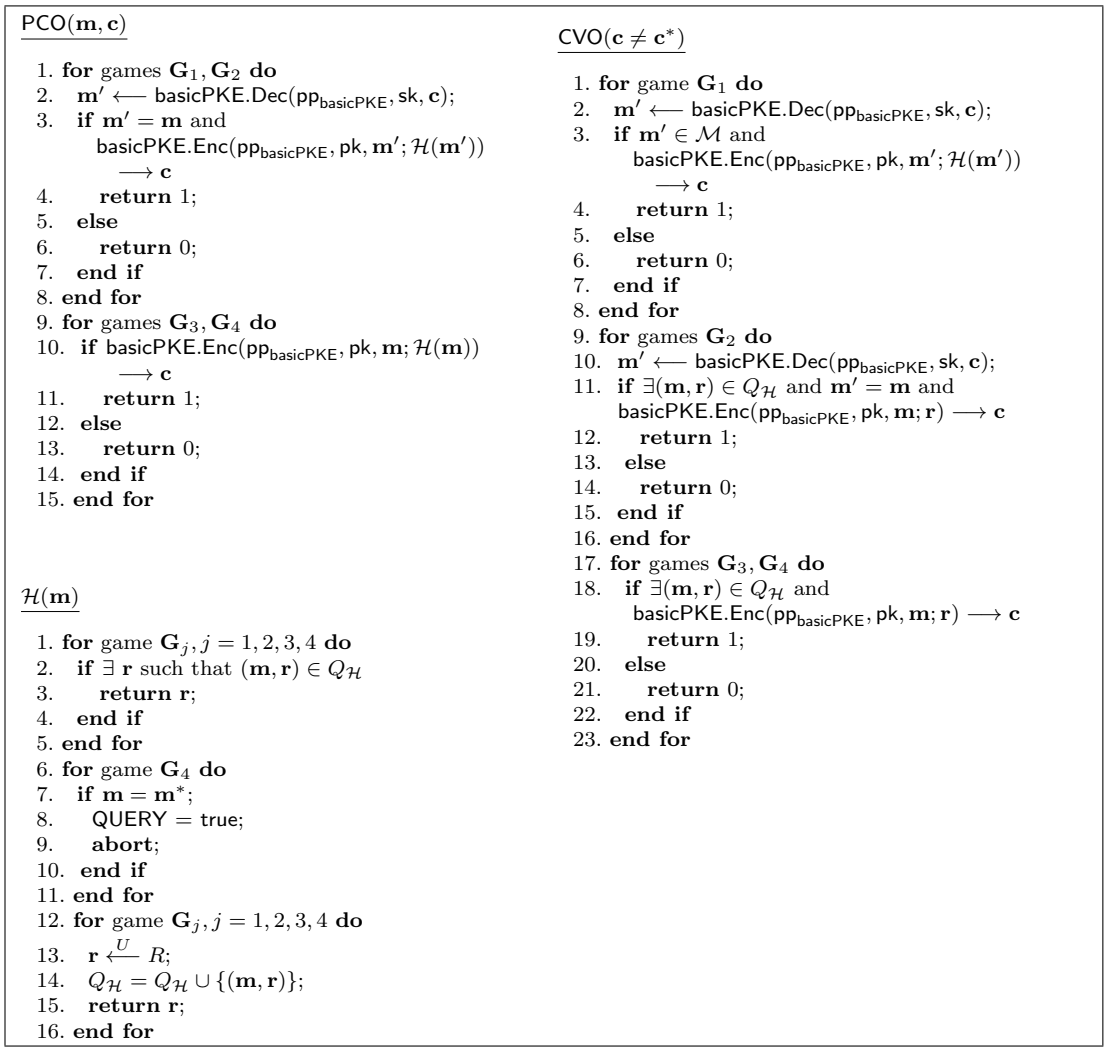


Fig. 7. The Plaintext Checking Oracle $\mathsf{PCO}(\cdot,\cdot)$, Ciphertext Validity Oracle $\mathsf{CVO}(\cdot,\cdot)$ and hash oracle $\mathcal{H}(\cdot)$ for games $\mathbf{G}_j, j = 1, 2, 3, 4$

Now we prove that the scheme basicPKE is γ-uniform.

Lemma 5. *The scheme* basicPKE *is γ-uniform (Definition 2) with* $\gamma = \dfrac{2^{-(k-k')m}}{\binom{(n-k)m}{w-\mathsf{wt}(\boldsymbol{\mu})}}$.

Proof of Lemma 5. Let $\mathbf{c}$ be a generic vector of the basicPKE ciphertext space $(\mathsf{GF}(2))^{(n-k)m}$. Then either $\mathbf{c}$ is a word at distance w from the code, or it is not. If it is not, the probability of $\mathbf{c}$ being a valid ciphertext is exactly 0. On the other hand, suppose $\mathbf{c}$ is at distance w from the code. Then there is only one choice of $\boldsymbol{\rho}$ with probability $1/2^{(k-k')m}$ and one choice of $\mathbf{e}$ with probability $1/\binom{(n-k)m}{w-\mathsf{wt}(\boldsymbol{\mu})}$ that satisfy the equation (see steps (i) and (ii) in procedure basicPKE.Enc in Section 3),

i.e. the probability of $\mathbf{c}$ being a valid $\mathsf{basicPKE}$ ciphertext is exactly $\gamma = (1/2^{(k-k')m}) \cdot (1/\binom{(n-k)m}{w-\mathsf{wt}(\boldsymbol{\mu})})$. Therefore, $\Pr_{\mathbf{r} \leftarrow\!\!\!- \mathcal{R}}[\mathbf{c} = \mathsf{basicPKE.Enc}(\mathsf{pp}_{\mathsf{basicPKE}}, \mathsf{pk}, \mathbf{m}; \mathbf{r})] \leq \gamma$ for any $\mathbf{c} \in (\mathsf{GF}(2))^{(n-k)m}$ which completes the proof. ■ (of Lemma 5)

Now consider a query $\mathsf{CVO}(\mathbf{c})$. Let $\mathbf{m}' \longleftarrow\!\!\!- \mathsf{basicPKE.Dec}(\mathsf{pp}_{\mathsf{basicPKE}}, \mathsf{sk}, \mathbf{c})$. If $\mathsf{CVO}(\mathbf{c}) \longrightarrow 1$ in game $\mathbf{G}_2$, then $\exists\ (\mathbf{m}, \mathbf{r}) \in Q_{\mathcal{H}}$ with $\mathbf{m}' = \mathbf{m}$ and $\mathsf{basicPKE.Enc}(\mathsf{pp}_{\mathsf{basicPKE}}, \mathsf{pk}, \mathbf{m}; \mathbf{r}) \longrightarrow \mathbf{c}$ (see line 11 in oracle CVO in Figure 7). Hence $\mathcal{H}(\mathbf{m}') = \mathcal{H}(\mathbf{m}) = \mathbf{r}$ and $\mathsf{basicPKE.Enc}(\mathsf{pp}_{\mathsf{basicPKE}}, \mathsf{pk}, \mathbf{m}'; \mathcal{H}(\mathbf{m}')) \longrightarrow \mathbf{c}$. This means $\mathsf{CVO}(\mathbf{c}) \longrightarrow 1$ in game $\mathbf{G}_1$. If $\mathsf{CVO}(\mathbf{c}) \longrightarrow 1$ in game $\mathbf{G}_1$, then we can have $\mathsf{CVO}(\mathbf{c}) \longrightarrow 0$ in game $\mathbf{G}_2$ if $\mathcal{H}(\mathbf{m}')$ was not queried before in game $\mathbf{G}_2$. Let L be the event that $\mathcal{H}(\mathbf{m}')$ was not queried before the CVO oracle query. Games $\mathbf{G}_1$ and $\mathbf{G}_2$ are exactly the same unless L occurs. Suppose that $\mathbf{c}$ is a valid ciphertext with respect to $\mathsf{pp}_{\mathsf{basicPKE}}$, pk i.e., there exists $\mathbf{m}$ and $\mathbf{r}$ such that $\mathsf{basicPKE.Enc}(\mathsf{pp}_{\mathsf{basicPKE}}, \mathsf{pk}, \mathbf{m}; \mathbf{r}) \longrightarrow \mathbf{c}$. Then the probability of the event L for a single query is $2^{-\gamma}$ where γ is the parameter defined in Lemma 5. On the contrary, if $\mathbf{c}$ is an invalid ciphertext with respect to $\mathsf{pp}_{\mathsf{basicPKE}}$ and pk, the event L does not occur. As the adversary $\mathcal{A}$ can make at most n_V queries to the oracle CVO, we obtain

$$|\Pr[E_1] - \Pr[E_2]| \leq n_V \cdot 2^{-\gamma}.$$

Game $\mathbf{G}_3$: In game $\mathbf{G}_3$, the oracles $\mathsf{PCO}(\mathbf{m}, \mathbf{c})$ and $\mathsf{CVO}(\mathbf{c})$ are simulated by the challenger $\mathcal{S}$ without checking $\mathbf{m} = \mathbf{m}'$ where $\mathbf{m}' \longleftarrow\!\!\!- \mathsf{basicPKE.Dec}(\mathsf{pp}_{\mathsf{basicPKE}}, \mathsf{sk}, \mathbf{c})$ (see line 10 of PCO oracle and line 18 of CVO oracle in Figure 7). In games $\mathbf{G}_2$ and $\mathbf{G}_3$, it can be noted that the adversary makes at most $n_{\mathcal{H}}$ distinct queries $\mathcal{H}(\boldsymbol{m_1})$, $\mathcal{H}(\boldsymbol{m_2}), \ldots, \mathcal{H}(\boldsymbol{m_{n_{\mathcal{H}}}})$ to the hash oracle $\mathcal{H}$. We consider such a query $\mathcal{H}(\boldsymbol{m_i})$ problematic if and only if it exhibits a correctness error in the scheme $\mathsf{basicPKE}$. As there is no correctness error in $\mathsf{basicPKE}$, no query $\mathcal{H}(\boldsymbol{m_i})$ is problematic. Consequently, games $\mathbf{G}_2$ and $\mathbf{G}_3$ are identical. Indeed, games $\mathbf{G}_2$ and $\mathbf{G}_3$ differ if the adversary $\mathcal{A}$ submits a PCO oracle query on $(\mathbf{m}, \mathbf{c})$ or a CVO oracle query on $\mathbf{c}$ together with a $\mathcal{H}$ oracle query on $\mathbf{m}$ such that $\mathcal{H}(\mathbf{m})$ is problematic and $\mathsf{basicPKE.Enc}(\mathsf{pp}_{\mathsf{basicPKE}}, \mathsf{pk}, \mathbf{m}; \mathcal{H}(\mathbf{m})) \longrightarrow \mathbf{c}$. In this case, the challenger $\mathcal{S}$ will answer the query with 0 in game $\mathbf{G}_2$ as $\mathbf{m}' \neq \mathbf{m}$, while $\mathcal{S}$ will response the query with 1 in game $\mathbf{G}_3$. Therefore, we have

$$\Pr[E_3] = \Pr[E_2].$$

Game $\mathbf{G}_4$: In game $\mathbf{G}_4$, the challenger $\mathcal{S}$ sets a flag QUERY=true and aborts (with uniform random output), when the adversary $\mathcal{A}$ submits query to the hash oracle $\mathcal{H}$ on $\mathbf{m}^*$. Thus, games $\mathbf{G}_3$ and $\mathbf{G}_4$ differ if the flag QUERY=true is raised, meaning that $\mathcal{A}$ made a query $\mathcal{H}$ on $\mathbf{m}^*$, or, equivalently, $(\mathbf{m}^*, \cdot) \in Q_{\mathcal{H}}$. Hence, the games $\mathbf{G}_3$ and $\mathbf{G}_4$ behave identically unless $\mathsf{QUERY} = \mathsf{true}$ occurs. So,

$$|\Pr[E_3] - \Pr[E_4]| \leq \Pr[\mathsf{QUERY} = \mathrm{true}].$$

To get the bound for $\Pr[E_4]$, we construct an adversary $\mathcal{A}'$ against the OW-CPA security of the scheme basicPKE simulating game $\mathbf{G}_4$ for $\mathcal{A}$ (see Figure 8). The adversary $\mathcal{A}'$ takes $\mathsf{pp}_{\mathsf{basicPKE}}, \mathsf{pk}, \mathbf{c}^*$ as input, perfectly simulates game $\mathbf{G}_4$ for $\mathcal{A}$ and finally outputs $\mathbf{m}' = \mathbf{m}^*$ if $\mathcal{A}$ wins in game $\mathbf{G}_4$. Here, $\mathcal{A}$ uses the same $\mathsf{PCO}, \mathsf{CVO}$ oracles for game $\mathbf{G}_4$ in Figure 7 with the same hash oracle $\mathcal{H}$ for game $\mathbf{G}_3$ in Figure 7. Therefore, we get

$$\Pr[E_4] = \mathsf{Adv}^{\mathsf{OW\text{-}CPA}}_{\mathsf{basicPKE}}(\mathcal{A}').$$

$\underline{\mathcal{A}'(\mathsf{pp}_{\mathsf{basicPKE}}, \mathsf{pk}, \mathbf{c}^*)}$

1. $\mathbf{m}' \longleftarrow \mathcal{A}^{\mathcal{H}(\cdot),\mathsf{PCO}(\cdot,\cdot),\mathsf{CVO}(\cdot)}(\mathsf{pp}_{\mathsf{basicPKE}}, \mathsf{pk}, \mathbf{c}^*)$;
2. **return** $\mathbf{m}'$;

Fig. 8. Adversary $\mathcal{A}'$ against OW-CPA security of basicPKE

Combining all the probabilities, we have

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{OW\text{-}PCVA}}_{\mathsf{fullPKE}}(\mathcal{A}) = |\Pr[E_0]| &= |\Pr[E_1]| \\
&= |\Pr[E_1] - \Pr[E_2] + \Pr[E_2]| \\
&\leq |\Pr[E_1] - \Pr[E_2]| + |\Pr[E_2]| \\
&\leq n_V \cdot 2^{-\gamma} + |\Pr[E_3]| \\
&= n_V \cdot 2^{-\gamma} + |\Pr[E_3] - \Pr[E_4] + \Pr[E_4]| \\
&\leq n_V \cdot 2^{-\gamma} + |\Pr[E_3] - \Pr[E_4]| + |\Pr[E_4]| \\
&\leq n_V \cdot 2^{-\gamma} + \Pr[\mathsf{QUERY} = \mathsf{true}] + \mathsf{Adv}^{\mathsf{OW\text{-}CPA}}_{\mathsf{basicPKE}}(\mathcal{A}')
\end{aligned}$$

Now we consider the following relation between OW-CPA security and IND-CPA security of a public key encryption scheme (see Remark 1).

Lemma 6. *[27] Let* PKE *be a public key encryption scheme. Then, for any* OW-CPA *adversary* $\mathcal{B}$*, there exists an* IND-CPA *adversary* $\mathcal{A}$ *with the same running time as that of* $\mathcal{B}$ *such that*

$$\mathsf{Adv}^{\mathsf{OW\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{B}) \leq \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{A}) + 1/|\mathcal{M}|$$

where $\mathcal{M}$ *is the message space.*

As $\mathcal{M} = (\mathsf{GF}(2))^{k'm}$ in basicPKE, we have by Lemma 6, we have

$$\mathsf{Adv}^{\mathsf{OW\text{-}PCVA}}_{\mathsf{fullPKE}}(\mathcal{A}) \leq n_V \cdot 2^{-\gamma} + \Pr[\mathsf{QUERY} = \mathsf{true}] + \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{basicPKE}}(\mathcal{A}'') + \frac{1}{2^{k'm}}$$

for an IND-CPA adversary $\mathcal{A}''$. Now we construct another adversary $\mathcal{D}$ (Figure 9)

$\underline{\mathcal{D}(\mathsf{pp}_{\mathsf{basicPKE}}, \mathsf{pk})}$

1. $(\mathbf{m}_0^*, \mathbf{m}_1^*) \xleftarrow{U} \mathcal{M} \times \mathcal{M}$;
2. $\mathbf{m}' \longleftarrow \mathcal{A}^{\mathcal{H}(\cdot), \mathsf{PCO}(\cdot,\cdot), \mathsf{CVO}(\cdot)}(\mathsf{pp}_{\mathsf{basicPKE}}, \mathsf{pk}, \mathbf{c}^* \longleftarrow \mathsf{basicPKE.Enc}(\mathsf{pp}_{\mathsf{basicPKE}}, \mathsf{pk}, \mathbf{m}_b^*; \mathbf{r}_b^*))$;
3. $b' = \begin{cases} 0 & \text{for } |Q_{\mathcal{H}}(\mathbf{m}_0^*)| = 1, |Q_{\mathcal{H}}(\mathbf{m}_1^*)| = 0; \\ 1 & \text{for } |Q_{\mathcal{H}}(\mathbf{m}_0^*)| = 0, |Q_{\mathcal{H}}(\mathbf{m}_1^*)| = 1; \\ \xleftarrow{U} \{0,1\} & \text{for } |Q_{\mathcal{H}}(\mathbf{m}_0^*)| = |Q_{\mathcal{H}}(\mathbf{m}_1^*)|; \end{cases}$
4. **return** b';

Fig. 9. Adversary $\mathcal{D}$ against IND-CPA security of basicPKE

against the IND-CPA security of the scheme basicPKE which wins when the flag QUERY=true is set in game $\mathbf{G}_4$. The adversary $\mathcal{D}$ picks two random messages $\mathbf{m}_0^*, \mathbf{m}_1^*$ and runs $\mathcal{A}$ on $(\mathsf{pp}_{\mathsf{basicPKE}}, \mathsf{pk}, \mathbf{c}^*)$ where $\mathbf{c}^* \longleftarrow \mathsf{basicPKE.Enc}(\mathsf{pp}_{\mathsf{basicPKE}}, \mathsf{pk}, \mathbf{m}_b^*; \mathbf{r}_b^*)$, $b \xleftarrow{U} \{0,1\}$, is generated and sent by the IND-CPA challenger $\mathcal{C}_h$ in the IND-CPA security game between $\mathcal{C}_h$ and $\mathcal{D}$. Now consider the IND-CPA security game for the adversary $\mathcal{D}$ with random challenge bit b. Let Z be the event that $\mathcal{A}$ queries random oracle $\mathcal{H}$ on $\mathbf{m}_{1-b}^*$. Since the message $\mathbf{m}_{1-b}^*$ is taken uniformly from $\mathcal{M}$ and independent from $\mathcal{A}$'s view, we have $\Pr[Z] \leq \frac{n_{\mathcal{H}}}{2^{k'm}}$. Now let us assume the event that Z did not happen which leads $|Q_{\mathcal{H}}(\mathbf{m}_{1-b}^*)| = 0$. Here $|Q_{\mathcal{H}}(\mathbf{m})|$ denotes the number of all $(\mathbf{m}, \mathbf{r}) \in Q_{\mathcal{H}}$ for a fixed $\mathbf{m} \in \mathcal{M}$. Here, $|Q_{\mathcal{H}}(\mathbf{m})|$ is either 1 or 0 for a message $\mathbf{m}$. When QUERY=true occurs, the adversary $\mathcal{A}$ queries the random oracle $\mathcal{H}$ on $\mathbf{m}_b^*$. Thus, $|Q_{\mathcal{H}}(\mathbf{m}_b^*)| = 1, |Q_{\mathcal{H}}(\mathbf{m}_{1-b}^*)| = 0$ (as Z did not happen) and therefore $b = b'$. When QUERY=true does not occur, adversary $\mathcal{A}$ did not query the hash oracle $\mathcal{H}$ on $\mathbf{m}_b^*$. So, $|Q_{\mathcal{H}}(\mathbf{m}_b^*)| = Q_{\mathcal{H}}(\mathbf{m}_{1-b}^*)| = 0$ and $\Pr[b = b'] = 1/2$ as $\mathcal{D}$ chooses a random bit b'. From all of these relations we can write

$$\begin{aligned} |\Pr[b = b'|\overline{Z}] - 1/2| + |\Pr[b = b'|Z]| &\geq |\Pr[b = b'|Z] + \Pr[b = b'|\overline{Z}] - 1/2| \\ &\geq |\Pr[b = b'|\overline{Z}] - 1/2| \end{aligned}$$

which yields

$$\begin{aligned} \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{basicPKE}}(\mathcal{D}) + \frac{n_{\mathcal{H}}}{2^{k'm}} &\geq |\Pr[b = b'|\overline{Z}] - 1/2| \\ &= |\Pr[b = b'|\mathsf{QUERY} = \mathsf{true}] \cdot \Pr[\mathsf{QUERY} = \mathsf{true}] \\ &\quad + \Pr[b = b'|\overline{\mathsf{QUERY} = \mathsf{true}}] \cdot \Pr[\overline{\mathsf{QUERY} = \mathsf{true}}] - 1/2| \\ &= |\Pr[\mathsf{QUERY} = \mathsf{true}] + 1/2 \Pr[\overline{\mathsf{QUERY} = \mathsf{true}}] - 1/2| \\ &= |\Pr[\mathsf{QUERY} = \mathsf{true}] + 1/2(1 - \Pr[\mathsf{QUERY} = \mathsf{true}]) - 1/2| \\ &= 1/2 \Pr[\mathsf{QUERY} = \mathsf{true}]. \end{aligned}$$

From the above relation and combining two IND-CPA adversaries $\mathcal{A}''$ and $\mathcal{D}$ to a new IND-CPA adversary $\mathcal{B}$ we have

$$\mathsf{Adv}^{\mathsf{OW\text{-}PCVA}}_{\mathsf{fullPKE}}(\mathcal{A}) \leq n_V \cdot 2^{-\gamma} + (2n_{\mathcal{H}} + 1)/2^{k'm} + 3 \cdot \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{basicPKE}}(\mathcal{B})$$

which completes the proof. ■

The OW-PCVA security for a PKE scheme (Definition 4 in Subsection 2.1) trivially implies the OW-VA security of the scheme considering zero queries to the $\mathsf{PCO}(\cdot,\cdot)$ oracle. Therefore, the following corollary is an immediate consequence of Theorem 7.

Corollary 1. *If the scheme* basicPKE =(Setup, KeyGen, Enc, Dec) *as described in Section 3 is* IND-CPA *secure (Definition 3 in Subsection 2.1), then the public key encryption scheme* fullPKE = (Setup, KeyGen, Enc, Dec) *as described in Section 4 provides* OW-VA *security (Definition 4 in Subsection 2.1) considering the hash function* $\mathcal{H}$ *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) following the specifications of Definition 5.

- $\mathsf{fullKEM.Setup}(\lambda) \longrightarrow \mathsf{pp_{fullKEM}}$: A trusted authority runs $\mathsf{fullPKE.Setup}(\lambda)$, chooses another cryptographic hash function $\mathcal{H}_2 : \{0,1\}^* \longrightarrow \{0,1\}^r$ and sets public parameters $\mathsf{pp_{fullKEM}} = (k, k', w, r, q, m, \gamma, \mathcal{H}, \mathcal{H}_1, \mathcal{H}_2)$ taking security parameter λ as input.
- $\mathsf{fullKEM.KeyGen}(\mathsf{pp_{fullKEM}}) \longrightarrow (\mathsf{pk}, \mathsf{sk})$: A user generates public-secret key pair $(\mathsf{pk}, \mathsf{sk})$ by running $\mathsf{fullPKE.KeyGen}(\mathsf{pp_{fullKEM}})$ where $\mathsf{pk} = \widehat{M}$ and $\mathsf{sk} = (z_0, z_1, \ldots, z_{k-1})$.
- $\mathsf{fullKEM.Encaps}(\mathsf{pp_{fullKEM}}, \mathsf{pk}) \longrightarrow (\mathsf{CT}, K)$: Given system parameters $\mathsf{pp_{fullKEM}} = (k, k', w, r, q, m, \gamma, \mathcal{H}, \mathcal{H}_1, \mathcal{H}_2)$ and public key $\mathsf{pk} = \widehat{M}$, an encapsulator proceeds as follows to generate a ciphertext header $\mathsf{CT} \in (\mathsf{GF}(2))^{(k+k')m}$ and an encapsulation key $K \in \{0,1\}^r$.

(i) Sample $\mathbf{m} \xleftarrow{U} (\mathsf{GF}(2))^{k'm}$ and compute $\mathbf{r} = \mathcal{H}(\mathbf{m}) \in (\mathsf{GF}(2))^{km}$, $\mathbf{d} = \mathcal{H}_1(\mathbf{m}) \in (\mathsf{GF}(2))^{k'm}$.

(ii) Parse $\mathbf{r}$ as $\mathbf{r} = (\boldsymbol{\rho}||\boldsymbol{\sigma})$ where $\boldsymbol{\rho} \in (\mathsf{GF}(2))^{(k-k')m}$, $\boldsymbol{\sigma} \in (\mathsf{GF}(2))^{k'm}$. Set $\boldsymbol{\mu} = (\boldsymbol{\rho}||\mathbf{m}) \in (\mathsf{GF}(2))^{km}$.

(iii) Run Algorithm 2 to generate a unique error vector $\mathbf{e}$ of length $(n-k)m$ and weight $w - \mathsf{wt}(\boldsymbol{\mu})$ using $\boldsymbol{\sigma} \in (\mathsf{GF}(2))^{k'm}$ as a seed. Set $\mathbf{e}' = (\mathbf{e}||\boldsymbol{\mu}) \in (\mathsf{GF}(2))^{nm}$.

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

(v) Compute the syndrome $\mathbf{c} = \widehat{H}(\mathbf{e}')^T \in (\mathsf{GF}(2))^{(n-k)m}$ and the encapsulation key $K = \mathcal{H}_2(\mathbf{m}) \in \{0,1\}^r$ where $\mathcal{H}_2$ is the hash function given in $\mathsf{pp_{fullKEM}}$.

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

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

(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 $\mathsf{fullPKE.Dec}$ (i.e as in $\mathsf{basicPKE.Dec}$).

(ii) Let $\mathbf{e}'' = (\mathbf{e}_0 || \boldsymbol{\mu}') \in (\mathsf{GF}(2))^{nm}$ and $\boldsymbol{\mu}' = (\boldsymbol{\rho}' || \mathbf{m}') \in (\mathsf{GF}(2))^{km}$ where $\mathbf{e}_0 \in (\mathsf{GF}(2))^{(n-k)m}$, $\boldsymbol{\rho}' \in (\mathsf{GF}(2))^{(k-k')m}$, $\mathbf{m}' \in (\mathsf{GF}(2))^{k'm}$.

(iii) Compute $\mathbf{r}' = \mathcal{H}(\mathbf{m}') \in (\mathsf{GF}(2))^{km}$ and $\mathbf{d}' = \mathcal{H}_1(\mathbf{m}') \in (\mathsf{GF}(2))^{k'm}$ extracting $\mathcal{H}$ and $\mathcal{H}_1$ from $\mathsf{pp_{fullKEM}}$.

(iv) Parse $\mathbf{r}'$ as $\mathbf{r}' = (\boldsymbol{\rho}'' || \boldsymbol{\sigma}')$ where $\boldsymbol{\rho}'' \in (\mathsf{GF}(2))^{(k-k')m}$, $\boldsymbol{\sigma}' \in (\mathsf{GF}(2))^{k'm}$.

(v) Run Algorithm 2 to generate an error vector $\mathbf{e}_0' \in (\mathsf{GF}(2))^{(n-k)m}$ deterministically of length $(n-k)m$ and weight $w - \mathsf{wt}(\boldsymbol{\mu}')$ using $\boldsymbol{\sigma}' \in (\mathsf{GF}(2))^{k'm}$ as seed.

(vi) If $(\mathbf{e}_0 \neq \mathbf{e}_0') \vee (\boldsymbol{\rho}' \neq \boldsymbol{\rho}'') \vee (\mathbf{d} \neq \mathbf{d}')$, output $\perp$ indicating decapsulation failure. Otherwise, compute the encapsulation key $K = \mathcal{H}_2(\mathbf{m}')$ where $\mathcal{H}_2$ is the hash function given in $\mathsf{pp_{fullKEM}}$.

Correctness. Correctness of $\mathsf{fullKEM}$ follows from that of $\mathsf{fullPKE}$.

Theorem 8. *If the public key encryption scheme* $\mathsf{fullPKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *as described in Section 4 is* OW-VA *secure (Definition 4 in Subsection 2.1), then the key encapsulation mechanism* $\mathsf{fullKEM} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ *as described above provides* IND-CCA *security (Definition 6 in Subsection 2.2) when the hash function* $\mathcal{H}_2$ *is modeled as a random oracle.*

- The challenger $\mathcal{S}$ generates $\mathsf{pp_{fullKEM}} \longleftarrow \mathsf{fullKEM.Setup}(\lambda)$ and $(\mathsf{pk}, \mathsf{sk}) \longleftarrow \mathsf{fullKEM.KeyGen}(\mathsf{pp_{fullKEM}})$ for a security parameter λ and sends $\mathsf{pp_{fullKEM}}, \mathsf{pk}$ to the adversary $\mathcal{B}$.
- The PPT adversary $\mathcal{B}$ has access to the decapsulation oracle $\mathsf{fullKEM.Decaps}$ to which $\mathcal{B}$ can make polynomially many ciphertext queries CT_i and receives the corresponding key $K_i \in \mathcal{K} = \{0,1\}^r$ from $\mathcal{S}$.
- The challenger $\mathcal{S}$ chooses a random bit b from $\{0,1\}$, runs $\mathsf{fullKEM.Encaps}(\mathsf{pp_{fullKEM}}, \mathsf{pk})$ to generate a ciphertext-key pair (CT^*, K_0^*) with $\mathsf{CT}^* \neq \mathsf{CT}_i$, selects randomly $K_1^* \in \mathcal{K}$ and sends the pair (CT^*, K_b^*) to $\mathcal{B}$.
- The adversary $\mathcal{B}$ having the pair (CT^*, K_b^*) keeps submitting polynomially many decapsulation queries on $\mathsf{CT}_i \neq \mathsf{CT}^*$ and finally outputs b'.

Fig. 10. Game $\mathbf{G}_0$ in the proof of Theorem 8

Proof. Let $\mathcal{B}$ be a PPT adversary against the IND-CCA security of the $\mathsf{fullKEM}$ providing at most n_D queries to the oracle $\mathsf{fullKEM.Decaps}$ and at most $n_{\mathcal{H}_2}$ queries to

the hash oracle $\mathcal{H}_2$. We show that there exists a PPT adversary $\mathcal{A}$ against the OW-VA security of fullPKE. We begin with a sequence of games and the view of the adversary $\mathcal{B}$ is shown to be computationally indistinguishable in any of the consecutive games. Finally, we finish up in a game that statistically hides the challenge bit as needed. The games are described in Figure 10 and Figure 11. Let E_j be the event that $b = b'$ in game $\mathbf{G}_j$, $j = 0, 1, 2, 3$.

Game $\mathbf{G}_0$: As usual, game $\mathbf{G}_0$ (Figure 10) is the standard IND-CCA security game for the fullKEM as in Definition 6 in Subsection 2.2 and we have

$$|\Pr[E_0] - 1/2| = \mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{fullKEM}}(\mathcal{B}).$$

Game $\mathbf{G}_1$: In game $\mathbf{G}_1$, the challenger $\mathcal{S}$ chooses a message $\mathbf{m}^*$ randomly and

- The challenger $\mathcal{S}$ generates $\mathsf{pp}_{\mathsf{fullPKE}} \longleftarrow \mathsf{fullPKE.Setup}(\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \longleftarrow \mathsf{fullPKE.KeyGen}(\mathsf{pp}_{\mathsf{fullPKE}})$ for a security parameter λ and sends $\mathsf{pp}_{\mathsf{fullPKE}}, \mathsf{pk}$ to $\mathcal{B}$.
- The PPT adversary $\mathcal{B}$ has access to the decapsulation oracle Decaps (see Figure 12) to which $\mathcal{B}$ can make polynomially many ciphertext queries CT_i and gets the corresponding key $K_i \in \mathcal{K}$ from $\mathcal{S}$.
- The challenger $\mathcal{S}$ chooses a random bit b from $\{0, 1\}$, chooses a message $\mathbf{m}^* \xleftarrow{U} \mathcal{M}$, runs $\mathsf{fullPKE.Enc}(\mathsf{pp}_{\mathsf{fullPKE}}, \mathsf{pk}, \mathbf{m}^*; \mathbf{r}^*)$ to generate a ciphertext CT^*, computes $K_0^* = \mathcal{H}_2(\mathbf{m}^*)$, selects randomly $K_1^* \in \mathcal{K}$ and sends the pair (CT^*, K_b^*) to $\mathcal{B}$.
- The adversary $\mathcal{B}$ having the pair (CT^*, K_b^*) keeps submitting polynomially many decapsulation queries on $\mathsf{CT}_i \neq \mathsf{CT}^*$ to Decaps oracle and hash queries on $\mathbf{m}_i$ to hash oracle $\mathcal{H}_2$ and finally outputs b' (see Figure 12 for hash oracle $\mathcal{H}_2$ and decapsulation oracle Decaps).

Fig. 11. Sequence of games $\mathbf{G}_j, j = 1, 2, 3$ in the proof of Theorem 8

computes the ciphertext CT^* by running $\mathsf{fullPKE.Enc}(\mathsf{pp}_{\mathsf{fullPKE}}, \mathsf{pk}, \mathbf{m}^*; \mathbf{r}^*)$. The challenger also maintains a hash list $Q_{\mathcal{H}_2}$ (initially empty) and records all entries of the form $(\mathbf{m}, K)$ where hash oracle $\mathcal{H}_2$ is queried on some message $\mathbf{m} \in \mathcal{M}$. Here both games $\mathbf{G}_0$ and $\mathbf{G}_1$ proceed identically and we get

$$\Pr[E_0] = \Pr[E_1].$$

Game $\mathbf{G}_2$: In game $\mathbf{G}_2$, the hash oracle $\mathcal{H}_2$ and the decapsulation oracle Decaps are answered in such a way that they no longer use the secret key sk except for testing whether $\mathsf{fullPKE.Dec}(\mathsf{pp}_{\mathsf{fullPKE}}, \mathsf{sk}, \mathsf{CT}) \in \mathcal{M}$ for a given ciphertext CT (see line 12 of Decaps oracle in Figure 12). The hash list $Q_{\mathcal{H}_2}$ stores all entries of the form $(\mathbf{m}, K)$ where the hash oracle $\mathcal{H}_2$ is queried on some message $\mathbf{m} \in \mathcal{M}$. Another list Q_D records entries of the form (CT, K) where either Decaps oracle is queried on some ciphertext CT or the hash oracle $\mathcal{H}_2$ is queried on some message $\mathbf{m} \in \mathcal{M}$ satisfying $\mathsf{CT} \longleftarrow \mathsf{fullPKE.Enc}(\mathsf{pp}_{\mathsf{fullPKE}}, \mathsf{pk}, \mathbf{m}; \mathbf{r})$ with $\mathsf{fullPKE.Dec}(\mathsf{pp}_{\mathsf{fullPKE}}, \mathsf{sk}, \mathsf{CT}) \longrightarrow \mathbf{m}$.
Suppose ER denotes the event that a correctness error has occurred in the scheme

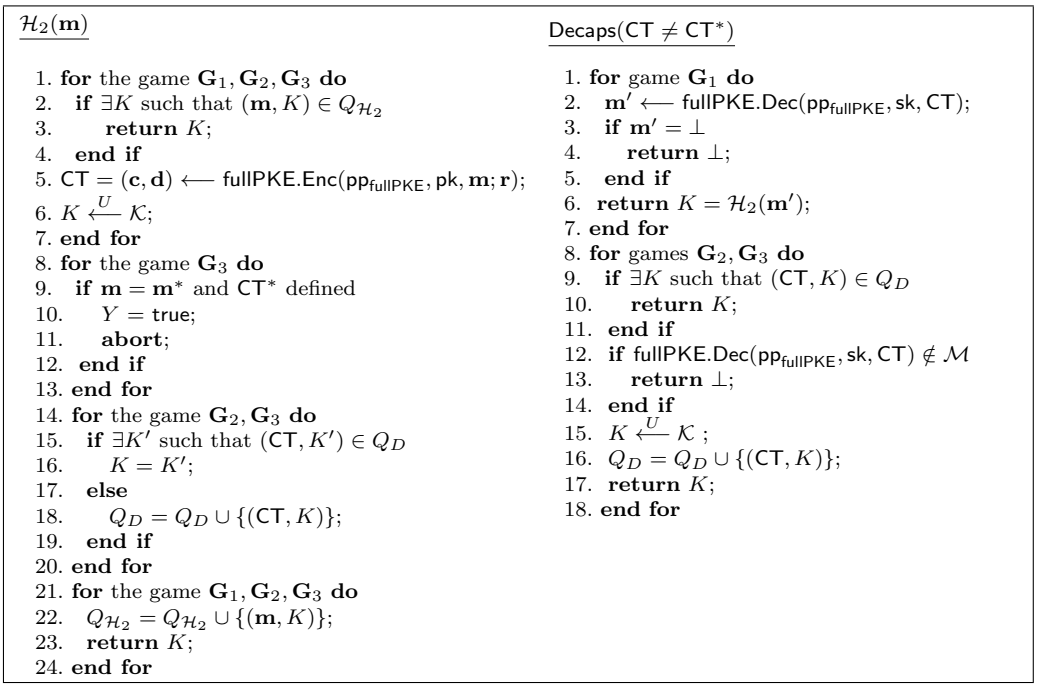


Fig. 12. The hash oracles $\mathcal{H}_2$ and the decapsulation oracle Decaps for games $\mathbf{G}_j, j = 1, 2, 3$ in the proof of Theorem 8

$\mathsf{fullPKE}$. More precisely, ER is the event where either the list $Q_{\mathcal{H}_2}$ contains an entry $(\mathbf{m}, K)$ satisfying the condition $\mathsf{fullPKE.Dec}(\mathsf{pp}_{\mathsf{fullPKE}}, \mathsf{sk}, \mathsf{fullPKE.Enc}(\mathsf{pp}_{\mathsf{fullPKE}}, \mathsf{pk}, \mathbf{m}; \mathbf{r}))$ $\neq \mathbf{m}$ or the list Q_D contains an entry (CT, K) satisfying the condition $\mathsf{fullPKE.Enc}$ $(\mathsf{pp}_{\mathsf{fullPKE}}, \mathsf{pk}, \mathsf{fullPKE.Dec}(\mathsf{pp}_{\mathsf{fullPKE}}, \mathsf{sk}, \mathsf{CT}); \mathbf{r}) \neq \mathsf{CT}$ or both.

Claim : In games $\mathbf{G}_1$ and $\mathbf{G}_2$, the view of $\mathcal{B}$ is identical unless the event ER occurs.

Proof of claim. To prove the claim, consider a fixed $\mathsf{fullPKE}$ ciphertext CT (placed as a Decaps query) with $\mathbf{m} \longleftarrow \mathsf{fullPKE.Dec}(\mathsf{pp}_{\mathsf{fullPKE}}, \mathsf{sk}, \mathsf{CT})$. Note that when $\mathbf{m} \notin \mathcal{M}$, the oracle $\mathsf{Decaps}(\mathsf{CT})$ returns $\perp$ in both games $\mathbf{G}_1$ and $\mathbf{G}_2$. Let $\mathbf{m} \in \mathcal{M}$. Now we prove that in game $\mathbf{G}_2$, $\mathsf{Decaps}(\mathsf{CT}) \longrightarrow \mathcal{H}_2(\mathbf{m})$ for the $\mathsf{fullPKE}$ ciphertext CT of a message $\mathbf{m} \in \mathcal{M}$ with $\mathsf{fullPKE.Enc}(\mathsf{pp}_{\mathsf{fullPKE}}, \mathsf{pk}, \mathbf{m}; \mathbf{r}) \longrightarrow \mathsf{CT}$. We distinguish two cases – $\mathcal{B}$ queries hash oracle $\mathcal{H}_2$ on $\mathbf{m}$ before querying the Decaps oracle on CT, or the other way round.

Case 1: Let the oracle $\mathcal{H}_2$ be queried on $\mathbf{m}$ first by $\mathcal{B}$ before decapsulation query on $\mathsf{fullPKE}$ ciphertext CT. Since Decaps oracle was not yet queried on CT, no entry of the form (CT, K) exist in the current list Q_D yet. Hence, besides adding $(\mathbf{m}, K \xleftarrow{U} \mathcal{K})$ to the list $Q_{\mathcal{H}_2}$ (see line 22 of $\mathcal{H}_2$ oracle in Figure 12), the challenger $\mathcal{S}$ also adds (CT, K) to the list Q_D (see line 18 of $\mathcal{H}_2$ oracle in Figure 12), thereby defining $\mathsf{Decaps}(\mathsf{CT}) \longrightarrow K = \mathcal{H}_2(\mathbf{m})$.

Case 2: Let the oracle Decaps be queried on $\mathsf{fullPKE}$ ciphertext CT before the hash oracle $\mathcal{H}_2$ is queried on message $\mathbf{m}$. Therefore no entry of the form (CT, K) exists in

Q_D yet. Otherwise, $\mathcal{H}_2$ already was queried on a message $\mathbf{m}'' \neq \mathbf{m}$ (as Decaps oracle is assumed to be queried first on CT and the oracle $\mathcal{H}_2$ was not yet queried on $\mathbf{m}$) satisfying $\mathsf{fullPKE.Enc}(\mathsf{pp}_{\mathsf{fullPKE}}, \mathsf{pk}, \mathbf{m}''; \mathbf{r}'') \longrightarrow \mathsf{CT}$ with $\mathsf{fullPKE.Dec}(\mathsf{pp}_{\mathsf{fullPKE}}, \mathsf{sk}, \mathsf{CT}) \longrightarrow \mathbf{m}''$. So, a contradiction arises to the fact that the same $\mathsf{fullPKE}$ ciphertext CT is obtained for two different messages $\mathbf{m}'', \mathbf{m}$ using randomness $\mathbf{r}, \mathbf{r}''$ respectively where $\mathbf{r} = \mathcal{H}(\mathbf{m}) \neq \mathcal{H}(\mathbf{m}'') = \mathbf{r}''$ for a cryptographically secure hash function $\mathcal{H}$. The randomness $\mathbf{r} = \mathcal{H}(\mathbf{m})$ used in the encryption algorithm of the $\mathsf{fullPKE}$ is generated determinitically using the message $\mathbf{m}$ and the hash function $\mathcal{H}$. Consequently, two different messages ($\mathbf{m}' \neq \mathbf{m}$) can not yield the same ciphertext CT. Therefore, Decaps oracle adds $(\mathsf{CT}, K \xleftarrow{U} \mathcal{K})$ to the list Q_D, thereby defining $\mathsf{Decaps}(\mathsf{CT}) \longrightarrow K$. When queried on $\mathbf{m}$ afterwards for hash oracle $\mathcal{H}_2$, an entry of the form (CT, K) already exists in the list Q_D (see line 15 of $\mathcal{H}_2$ oracle in Figure 12). By adding $(\mathbf{m}, K)$ to the list $Q_{\mathcal{H}_2}$ and returning K, the hash oracle $\mathcal{H}_2$ defines $\mathcal{H}_2(\mathbf{m}) = K \longleftarrow \mathsf{Decaps}(\mathsf{CT})$.
Hence, $\mathcal{B}$'s view is identical in games $\mathbf{G}_1$ and $\mathbf{G}_2$ unless ER occurs. ■ (of Claim)

As $\Pr[\mathsf{ER}] = 0$ for our $\mathsf{fullKEM}$, we have

$$\Pr[E_1] = \Pr[E_2].$$

Game $\mathbf{G}_3$: In game $\mathbf{G}_3$, the challenger $\mathcal{S}$ sets a flag $Y = \mathsf{true}$ and aborts immediately with uniformly random output when $\mathcal{B}$ queries the oracle $\mathcal{H}_2$ on $\mathbf{m}^*$. Hence, we get

$$|\Pr[E_2] - \Pr[E_3]| \leq \Pr[Y = \mathsf{true}].$$

In game $\mathbf{G}_3$, $\mathcal{H}_2(\mathbf{m}^*)$ will never be given to $\mathcal{B}$ neither through a query on the hash oracle $\mathcal{H}_2$ nor through a query on the decapsulation oracle Decaps, hence bit b is independent from $\mathcal{B}$'s view. Therefore, $\Pr[E_3] = 1/2$. Now it remains to achieve the bound $\Pr[Y = \mathsf{true}]$. To get this, we construct an adversary $\mathcal{A}$ against the $\mathsf{OW\text{-}VA}$ security of the scheme $\mathsf{fullPKE}$ simulating the game $\mathbf{G}_3$ for the adversary $\mathcal{B}$ as in Figure 13. Here $\mathcal{B}$ uses Decaps oracle as in Figure 13 with the same hash oracle

$\underline{\mathcal{A}^{\mathsf{CVO}(\cdot)}(\mathsf{pp}_{\mathsf{fullPKE}}, \mathsf{pk}, \mathsf{CT}^*)}$

1. $K^* \xleftarrow{U} \mathcal{K}$;
2. $b' \longleftarrow \mathcal{B}^{\mathsf{Decaps}(\cdot), \mathcal{H}_2(\cdot)}(\mathsf{pp}_{\mathsf{fullPKE}}, \mathsf{pk}, \mathsf{CT}^*, K^*)$;
3. **if** $\exists(\mathbf{m}', K') \in Q_{\mathcal{H}_2}$ such that $\mathsf{fullPKE.Enc}(\mathsf{pp}_{\mathsf{fullPKE}}, \mathsf{pk}, \mathbf{m}'; \mathbf{r}') \longrightarrow \mathsf{CT}^*$
4. **return** $\mathbf{m}'$;
5. **else**
6. **abort**;
7. **end if**

$\underline{\mathsf{Decaps}(\mathsf{CT} \neq \mathsf{CT}^*)}$

1. **if** $\exists K$ such that $(\mathsf{CT}, K) \in Q_D$
2. **return** K;
3. **end if**
4. **if** $\mathsf{CVO}(\mathsf{CT}) = 0$
5. **return** $\perp$;
6. **end if**
7. $K \xleftarrow{U} \mathcal{K}$;
8. $Q_D = Q_D \cup \{(\mathsf{CT}, K)\}$;
9. **return** K;

Fig. 13. Adversary $\mathcal{A}$ against $\mathsf{OW\text{-}VA}$ security of $\mathsf{fullPKE}$

$\mathcal{H}_2$ for the game $\mathbf{G}_2$ as in Figure 12. The ciphertext validity oracle $\mathsf{CVO}(\cdot)$ used in Figure 13 is shown in Figure 1 which does the same work as in the Decaps oracle in Figure 11 for game $\mathbf{G}_3$. Consequently, the simulation is perfect until the event $Y = \mathsf{true}$ happens. Moreover, $Y = \mathsf{true}$ ensures that $\mathcal{B}$ has queried $\mathcal{H}_2(\mathbf{m}^*)$, which implies that $(\mathbf{m}^*, K') \in Q_{\mathcal{H}_2}$ for some $K' \in \mathcal{K}$ where the list $Q_{\mathcal{H}_2}$ is maintained by $\mathcal{A}$ simulating G_3 for $\mathcal{B}$. In this case, we have $\mathsf{fullPKE.Enc}(\mathsf{pp}_{\mathsf{fullPKE}}, \mathsf{pk}, \mathbf{m}^*; \mathbf{r}^*) \longrightarrow \mathsf{CT}^*$ and therefore the adversary $\mathcal{A}$ returns $\mathbf{m}^*$. Therefore,

$$\Pr[Y = \mathsf{true}] = \mathsf{Adv}^{\mathsf{OW\text{-}VA}}_{\mathsf{fullPKE}}(\mathcal{A})$$

Combining all the probabilities, we get

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{fullKEM}}(\mathcal{B}) &= |\Pr[E_0] - 1/2| \\
&= |\Pr[E_1] - 1/2| \\
&= |\Pr[E_2] - 1/2| \\
&= |\Pr[E_2] - \Pr[E_3]| \\
&\leq \Pr[Y = \text{true}] \\
&= \mathsf{Adv}^{\mathsf{OW\text{-}VA}}_{\mathsf{fullPKE}}(\mathcal{A})
\end{aligned}$$

which completes the proof. ■

Theorem 9. *Assuming the hardness of decisional* SD *problem (Definition 13 in Subsection 2.5) and indistinguishability of the public key matrix* $\widehat{H}$ *(derived from the public key* pk *by running* $\mathsf{fullKEM.KeyGen}(\mathsf{pp}_{\mathsf{fullKEM}})$ *where* $\mathsf{pp}_{\mathsf{fullKEM}} \longleftarrow \mathsf{fullKEM.Setup}(\lambda)$*, λ being the security parameter), the scheme* $\mathsf{fullKEM} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ *described in Section 5 provides* IND-CCA *security (Definition 6 in Subsection 2.2) when the hash functions $\mathcal{H}$ and $\mathcal{H}_2$ are modeled as random oracles.*

Proof. The proof of the above theorem is the immediate consequence of Theorem 6, Corollary 1 and Theorem 8.

Remark 4. 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. We consider the security games in the quantum random oracle model along with the classical random oracle model. The adversaries equipped with a quantum computer are provided quantum access to the random oracles and classical access to some other oracles like plaintext checking oracles, ciphertext validity checking oracles, decapsulation oracles, etc. The scheme basicPKE provides IND-CPA security as the decisional SD problem is hard and the public key matrix is indistinguishable (see Theorem 6). Note that IND-CPA security always implies OW-CPA security. We can show that OW-CPA security of the encryption scheme basicPKE indicates OW-PCA security of fullPKE considering $\mathcal{H}$ as a quantum random oracle which follows from Theorem 10 stated below.

Theorem 10. *If the public key encryption scheme* basicPKE =(Setup, KeyGen, Enc, Dec) *described in Section 3 is* OW-CPA *secure (Definition 4 in Subsection 2.1), then the public key encryption scheme* fullPKE = (Setup, KeyGen, Enc, Dec) *as described in Section 4 provides* OW-PCA *security (Definition 4 in Subsection 2.1) when the hash function* $\mathcal{H}$ *is modeled as a quantum random oracle.*

Then, we can prove that OW-PCA security of fullPKE implies the IND-CCA security of the KEM modeling $\mathcal{H}_1, \mathcal{H}_2$ as quantum random oracles by Theorem 11.

Theorem 11. *If the public key encryption scheme* fullPKE = (Setup, KeyGen, Enc, Dec) *described in Section 4 is* OW-PCA *secure (Definition 4 in Subsection 2.1), then the scheme* fullKEM = (Setup, KeyGen, Encaps, Decaps) *as described in Section 5 achieves* IND-CCA *security (Definition 6 in Subsection 2.2) when the hash functions* $\mathcal{H}_1$ *and* $\mathcal{H}_2$ *are modeled as quantum random oracles.*

Therefore, combining Theorem 6, Remark 1, Theorem 10 and Theorem 11, Theorem 12 can be obtained.

Theorem 12. *Depending on the hardness of decisional* SD *problem (Definition 13 in Subsection 2.5) and indistinguishability of the public key matrix* $\widehat{H}$ *(derived from the public key* pk *by running* $\mathsf{fullKEM.KeyGen}(\mathsf{pp}_{\mathsf{fullKEM}})$ *where* $\mathsf{pp}_{\mathsf{fullKEM}} \longleftarrow \mathsf{fullKEM.Setup}(\lambda)$*,* λ *being the security parameter), our scheme* fullKEM = (Setup, KeyGen, Encaps, Decaps) *described in Section 5 provides* IND-CCA *security (Definition 6 in Subsection 2.2) when the hash functions* $\mathcal{H}, \mathcal{H}_1$ *and* $\mathcal{H}_2$ *are modeled as quantum random oracles.*

6 Conclusion

In this work, we give a proposal to design a key encapsulation mechanism based on MDS codes. We have shown that our KEM protocol provides IND-CCA security in the random oracle model and quantum random oracle model. In terms of storage, our work seems well with other protocols based on coding theory as shown in Table 1. More specifically, we are able to reduce the secret key size. The ciphertext size in our case is also shorter than some schemes for suitable parameters. In our proposal, we exploits MDS codes that does not involve a correctness error like some lattice-based schemes. From all of these aspects, we believe that our proposal to design a KEM will offer a promising alternative in the area of post-quantum cryptographic primitives.

References

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

2. Albrecht, M., Cid, C., Paterson, K.G., Tjhai, C.J., Tomlinson, M.: Nts-kem. NIST submissions (2019)
3. Aragon, N., Barreto, P., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.C., Gaborit, P., Gueron, S., Guneysu, T., Melchor, C.A., et al.: Bike: Bit flipping key encapsulation. NIST submissions (2017)
4. Aragon, N., Barreto, P., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.C., Gaborit, P., Gueron, S., Guneysu, T., Melchor, C.A., et al.: Bike: Bit flipping key encapsulation. NIST submissions (2019)
5. Aragon, N., Blazy, O., Deneuville, J.C., Gaborit, P., Hauteville, A., Ruatta, O., Tillich, J.P., Zémor, G.: Lake-low rank parity check codes key exchange (2017)
6. Aragon, N., Blazy, O., Deneuville, J.C., Gaborit, P., Hauteville, A., Ruatta, O., Tillich, J.P., Zémor, G.: Locker-low rank parity check codes encryption (2017)
7. Baldi, M., Barenghi, A., Chiaraluce, F., Pelosi, G., Santini, P.: Ledakem: a post-quantum key encapsulation mechanism based on qc-ldpc codes. In: International Conference on Post-Quantum Cryptography. pp. 3–24. Springer (2018)
8. 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.: Dags: Key encapsulation using dyadic gs codes. Journal of Mathematical Cryptology 12(4), 221–239 (2018)
9. Banegas, G., Barreto, P., Boidje, B.O., Cayrel, P.L., Dione, G.N., Gaj, K., Gueye, C.T., Haeussler, R., Klamti, J.B., N'diaye, O., et al.: Dags: Key encapsulation using dyadic gs codes. IACR Cryptology ePrint Archive 2017(1037) (2017)
10. Bardet, M., Barelli, E., Blazy, O., Canto-Torres, R., Couvreur, A., Gaborit, P., Otmani, A., Sendrier, N., Tillich, J.P.: Big quake. NIST submissions (2017)
11. Barg, A.: Complexity issues in coding theory. Electronic Colloquium on Computational Complexity (ECCC) 4(46) (1997)
12. Barreto, P.S., Gueron, S., Gueneysu, T., Misoczki, R., Persichetti, E., Sendrier, N., Tillich, J.P.: Cake: Code-based algorithm for key encapsulation. In: IMA International Conference on Cryptography and Coding. pp. 207–226. Springer (2017)
13. Berlekamp, E., McEliece, R., Van Tilborg, H.: On the inherent intractability of certain coding problems (corresp.). IEEE Transactions on Information Theory 24(3), 384–386 (1978)
14. Bernstein, D.J., Chou, T., Lange, T., von Maurich, I., Misoczki, R., Niederhagen, R., Persichetti, E., Peters, C., Schwabe, P., Sendrier, N., et al.: Classic mceliece: conservative code-based cryptography. NIST submissions (2017)
15. Bernstein, D.J., Chou, T., Schwabe, P.: Mcbits: fast constant-time code-based cryptography. In: International Workshop on Cryptographic Hardware and Embedded Systems. pp. 250–272. Springer (2013)
16. Blaum, M., Roth, R.M.: On lowest density mds codes. IEEE Transactions on Information Theory 45(1), 46–59 (1999)
17. Deneuville, J.C., Gaborit, P., Zémor, G.: Ouroboros: A simple, secure and efficient key exchange protocol based on coding theory. In: International Workshop on Post-Quantum Cryptography. pp. 18–34. Springer (2017)
18. Dey, J., Dutta, R.: Secure key encapsulation mechanism with compact ciphertext and public key from generalized srivastava code. In: International Conference on Information Security and Cryptology. pp. 175–193. Springer (2019)
19. Fabšič, T., Hromada, V., Stankovski, P., Zajac, P., Guo, Q., Johansson, T.: A reaction attack on the qc-ldpc mceliece cryptosystem. In: International Workshop on Post-Quantum Cryptography. pp. 51–68. Springer (2017)
20. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Annual International Cryptology Conference. pp. 537–554. Springer (1999)
21. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of computer and system sciences 28(2), 270–299 (1984)
22. Guo, Q., Johansson, T., Stankovski, P.: A key recovery attack on mdpc with cca security using decoding errors. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 789–815. Springer (2016)
23. Gupta, K.C., Pandey, S.K., Venkateswarlu, A.: On the direct construction of recursive mds matrices. Designs, Codes and Cryptography 82(1-2), 77–94 (2017)
24. Gupta, K.C., Pandey, S.K., Venkateswarlu, A.: Towards a general construction of recursive mds diffusion layers. Designs, Codes and Cryptography 82(1-2), 179–195 (2017)

25. Gupta, K.C., Pandey, S.K., Venkateswarlu, A.: Almost involutory recursive mds diffusion layers. Designs, Codes and Cryptography 87(2-3), 609–626 (2019)
26. Gupta, K.C., Ray, I.G.: On constructions of mds matrices from companion matrices for lightweight cryptography. In: International Conference on Availability, Reliability, and Security. pp. 29–43. Springer (2013)
27. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the fujisaki-okamoto transformation. In: Theory of Cryptography Conference. pp. 341–371. Springer (2017)
28. Kesarwani, A., Sarkar, S., Venkateswarlu, A.: Exhaustive search for various types of mds matrices. IACR Transactions on Symmetric Cryptology pp. 231–256 (2019)
29. Khathuria, K., Rosenthal, J., Weger, V.: Encryption scheme based on expanded reed-solomon codes. arXiv preprint arXiv:1906.00745 (2019)
30. Kim, J.L., Kim, Y.S., Galvez, L., Kim, M.J., Lee, N.: Mcnie: A code-based public-key cryptosystem. arXiv preprint arXiv:1812.05008 (2018)
31. MacWilliams, F.J., Sloane, N.J.A.: The theory of error-correcting codes, vol. 16. Elsevier (1977)
32. Massey, J.: Shift-register synthesis and bch decoding. IEEE transactions on Information Theory 15(1), 122–127 (1969)
33. McEliece, R.J.: A public-key cryptosystem based on algebraic coding theory. Coding Thv 4244, 114–116 (1978)
34. Melchor, C.A., Aragon, N., Bardet, M., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.C.: Rollo-rank-ouroboros, lake & locker (2019)
35. Melchor, C.A., Aragon, N., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.C., Gaborit, P., Hauteville, A., Zémor, G., Bourges, I.C.: Ouroboros-r (2017)
36. Niederreiter, H.: Knapsack-type cryptosystems and algebraic coding theory. Prob. Control and Inf. Theory 15(2), 159–166 (1986)
37. Nojima, R., Imai, H., Kobara, K., Morozov, K.: Semantic security for the mceliece cryptosystem without random oracles. Designs, Codes and Cryptography 49(1-3), 289–305 (2008)
38. Rackoff, C., Simon, D.R.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: Annual International Cryptology Conference. pp. 433–444. Springer (1991)
39. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: Proceedings 35th annual symposium on foundations of computer science. pp. 124–134. Ieee (1994)
40. Szepieniec, A.: Ramstake. Tech. rep., Technical report, National Institute of Standards and Technology (2017)
41. Wang, Y.: Rlcekey encapsulation mechanism (rlce-kem) specifcation. NIST Submission (2017)
42. Yamada, A., Eaton, E., Kalach, K., Lafrance, P., Parent, A.: Qc-mdpc kem: A key encapsulation mechanism based on the qc-mdpc mceliece encryption scheme. NIST Submission (2017)