Compact Fully Asynchronous Updatable Public Key Encryption Scheme from Hamming Quasi-Cyclic Cryptosystem

Sanajit Patra[1], Ratna Dutta[1], and Jayashree Dey[2]

[1] Indian Institute of Technology Kharagpur, India
sanajitpatra@kgpian.iitkgp.ac.in, ratna@maths.iitkgp.ac.in
[2] Indian Institute of Technology Madras, India
jayashreedey92@gmail.com

Abstract. In this work, we propose the *first* code-based uPKE from the *hamming quasi-cyclic public key encryption scheme* (hqcPKE) of Gaborit *et al.* by integrating an efficient key-update mechanism utilizing a carefully designed deterministic sampling algorithm. Our sampling algorithm exploits a structured permutation set preserving Hamming weight and satisfying specific algebraic properties that is of independent interest. More positively, our protocol allows unbounded key updates free from the cumulative error issues inheriant to lattice based designs and supports asynchronous key updates, authorizing senders to update the public key independently. We formally establish the security of the proposed construction against *indistinguishability under chosen-randomness and chosen-plaintext attack* (IND-CR-CPA) in standard model considering *decisional quasi-cyclic syndrome decoding with parity* (DQCSDP) assumption. We emphasize that in comparison with existing post-quantum secure asynchronous schemes supporting unbounded updates, our construction in the standard model instead of random oracle model achieves significantly improved storage and communication efficiency, particularly in terms of public key size, ciphertext size and update ciphertext size. From a computational perspective, our design enables a more efficient public key update procedure and exhibits comparable performance for key generation, encryption, decryption and secret key update.

Keywords: Updatable public key encryption · Code based cryptography · Forward security.

1 Introduction

The problem of enabling secure communication over untrusted channels lies at the core of cryptographic research. The security of encrypted data such as emails, group messages and financial transactions relies on the confidentiality of the secret key. Once this key is compromised, the protection afforded to all past communications collapses, allowing an adversary to recover sensitive information they may have collected over months or even years. *Forward security* (FS)

mitigates the issue by ensuring that the disclosure of future keys does not compromise the secrecy of past messages. Achieving FS in symmetric-key cryptography is comparatively straightforward and well understood. In a symmetric-key setting, given an initial key s_0, one can update the key as $(\mathsf{s}_i, \mathsf{k}_i) \longleftarrow \mathsf{PRG}(\mathsf{s}_{i-1})$ using a pseudo-random generator PRG where s_i is used for further key derivation and k_i is the one-time symmetric key. Obtaining forward security is particularly challenging in *public-key encryption* (PKE) scenarios. In such schemes, time is typically partitioned into discrete intervals. The receiver updates the secret key for the next interval using the secret key of the the current interval, keeping the public key unchanged. The foundational work of Canetti *et al.* [8] introduced the concept of *forward-secure public key encryption* (FS-PKE). In a FS-PKE system, the keys evolve over time and create two synchronized chains − one a public key chain $\mathsf{pk}_0 \rightarrow \mathsf{pk}_1 \rightarrow \mathsf{pk}_2 \rightarrow \cdots$ and a secret key chain $\mathsf{sk}_0 \rightarrow \mathsf{sk}_1 \rightarrow \mathsf{sk}_2 \rightarrow \cdots$, where pk_0 and sk_0 are respectively the initial public and secret keys. The key-update mechanism allows multiple senders to securely encrypt messages for a receiver who autonomously evolves their secret key across time periods. More precisely, the system is designed in such a way that a message encrypted with a public key pk_i can only be decrypted by its corresponding secret key sk_i. Most of the FS-PKE constructions employ *hierarchical identity-based encryption* (HIBE) techniques ([16,20] etc.). However, these designs continue to suffer from increased complexity and lower efficiency.

Updatable public key encryption (uPKE). To bridge this efficiency/complexity gap, the notion of uPKE was introduced by Jost *et al.* [21] and later advanced by Alwen *et al.* [3]. Beyond the functionality of standard public key encryption, uPKE schemes support asynchronous key updates in which both encryption and decryption keys are refreshed using newly generated entropy. This update mechanism enables the system to recover from key exposure by re-establishing security guarantees. A key distinction between uPKE and FS-PKE lies in who performs the update. In uPKE, updated keys may be generated at any time by any sender whereas it is restricted to the recipient in FS-PKE. More specifically, to update the public-secret key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ of a recipient u at epoch i, a sender can generate and transmit a public update $(\mathsf{pk}_{i+1}, \mathsf{up}_{i+1}$ where pk_{i+1} denotes the updated public key for next epoch $i+1$ formed using a randomness $\boldsymbol{\mu}_i$ and up_{i+1} refers to an update ciphertext that is derived by encrypting the randomness $\boldsymbol{\mu}_i$ under the public key pk_i corresponding to epoch i. Upon receiving this update, the recipient uses its secret key sk_i to decrypt up_{i+1} and recovers $\boldsymbol{\mu}_i$ and subsequently derives the updated secret key sk_{i+1} corresponding to the new public key pk_{i+1}.

Unlike standard PKE where a compromised secret key leads to a complete system reset, uPKE considers an environment where an adversary may gain access to secret key information at a epoch. The *indistinguishability under chosen-randomness and chosen-plaintext attack* (IND-CR-CPA) security allows an adversary to adaptively enforce updates on the target user's public key while controlling the randomness used in the update algorithm. More interestingly, it ensures that if an adversary gets the information regarding the secret key

sk_t at some epoch t, it could not decrypt any ciphertexts CT_i from previous epochs $i < t$. This guarantee remains valid provided that at least one key update in the interval between epochs i and t is carried out by an honest user, i.e., the update is generated using fresh private randomness that is unknown to the adversary. However, a more stronger security notion *indistinguishability under chosen-randomness and chosen-ciphertext attack* (IND-CR-CCA) requires that the adversary gets access to the decryption oracle corresponding to the secret key of the current epoch.

Related Work. Several uPKE constructions have been proposed in the past few years. Jost *et al.* [21] and Alwen *et al.* [3] proposed uPKE constructions following the framework of the hashed ElGamal public-key encryption scheme whose security relies on the *computational Diffie-Hellman* (CDH) assumption in the *random oracle model* (ROM). Later, Eaton *et al.* [13] introduced the first post-quantum secure uPKE design techniques depending on the *supersingular isogeny Diffie–Hellman* (SIDH) and *commutative supersingular isogeny Diffie-Hellman* (CSIDH) assumptions. Subsequently, Dodis *et al.* [11] eliminated the use of random oracles and presented two concrete uPKE constructions in the standard model based on the *decisional Diffie–Hellman* (DDH) and *learning with errors* (LWE) assumptions respectively. Haidar *et al.* [1] further proposed a scheme based on the *decisional composite residuosity* (DCR) assumption without random oracles. Later a more efficient LWE-based construction are presented by Haidar *et al.* [19] in the standard-model. They also proposed two generic transforms to achieve stronger security notion IND-CR-CCA refining the construction of Dodis *et al.* Later, Duparc *et al.* [12] proposed *supersingular isogeny lollipop based encryption* (SILBE) that does not rely on isogeny group actions which provides *one-way chosen-randomness chosen plaintext attack* (OW-CR-CPA) security, a weaker security framework. In 2024, Alwen *et al.* [4] proposed an efficient IND-CR-CCA secure uPKE scheme over bilinear groups in the ROM and the *algebraic group model*. In 2025, Asano and Watanabe [5] demonstrated a simple and efficient attack on Dodis *et al.*'s IND-CR-CCA secure scheme [11], suggested how to fix that and proposed a new IND-CR-CCA secure generic construction for a uPKE scheme in ROM. More recently, Albrecht *et al.* [2] achieved unbounded updatable encryption based on LWE and *permutation code equivalence* (PCE); however, their construction requires large parameters, stemming from the use of the leftover hash lemma and intricate security reductions.

1.1 Our Contribution

There is a growing demand for cryptographic schemes whose security is based on problems believed to be resistant to quantum attacks due to Shor [27]. Recent advances in quantum computing [28] have further emphasized the necessity of developing cryptographic primitives that remain secure in the post-quantum era. Lattice-based uPKE schemes ([11,19]) inherently suffer from noise accumulation across successive key updates. This limits the number of supported updates or necessitates expensive noise-management and compression techniques, thereby

reducing efficiency and scalability in long-term communication scenarios. On the other hand, isogeny-based constructions ([13,12]) either require coordination or interaction between senders and receivers during key updates. This prevents fully asynchronous operation or only supports a bounded number of updates. Moreover, some isogeny-based proposals rely on assumptions that have been weakened or broken, such as those underlying SIDH-based constructions. These limitations significantly hinder their suitability for practical, long-lived secure messaging applications.
Code-based cryptography stands out as an interesting option in post-quantum paradigms whose foundations rely primarily on algebraic coding theory over finite fields and the security is rooted in the well-studied hardness of decoding random linear codes. More interestingly, code-based protocols are well-suited for deployment across a wide range of platforms. Supporting efficient and scalable cryptographic functionality over extended periods is a fundamental requirement for secure messaging systems that rely on frequent public-key updates. Although updatable public-key encryption has witnessed notable progress, existing constructions often remain inefficient or impractical for real-world deployment, particularly in the post-quantum context. So far, no prior work considers uPKE in code-based environment, which motivates the exploration of new constructions that enable more efficient and practical post-quantum cryptographic solutions.

In this work, we adopt a coding theory approach and propose the *first* post-quantum secure uPKE construction for long-term secure messaging in code-based setting. Our starting point is the *hamming quasi-cyclic public key cryptosystem* (hqcPKE) of Gaborit *et al.* [15] that couples an efficiently decodable linear code with a random double-circulant code and derives security from the hardness of the *decisional quasi-cyclic syndrome decoding* (DQCSD) problem. To be more precise a parity version of DQCSD problem namely the *decisional quasi-cyclic syndrome decoding with parity* (DQCSDP) problem is assummed to be hard for their security. We removed the truncation mechanism presented in [15] and uses only the parity version assumption of DQCSDP problem. A deterministic sampling algorithm $\mathsf{Samp} : \mathcal{MS} \longrightarrow \mathsf{Per}(\mathcal{R})$ is used a structured permutation set $\mathsf{Per}(\mathcal{R})$ with few specific algebraic properties for our update mechanism where $\mathcal{R} = \mathbb{F}_2[x]/(x^n - 1)$ and $\mathcal{MS} = \mathbb{F}_2^k$ is the message space. We emphasize that our scheme enables fully asynchronous public-key updates without requiring sender-receiver coordination in contrast to the isogeny-based designs ([13,12]) and does not have the cumulative error issues inherent in lattice-based designs ([11,19]). In addition, our candidate supports an unbounded number of key updates and achieves IND-CR-CPA security under the DQCSDP assumption. We outline our main findings below.

- Based on our structured permutation set $\mathsf{Per}(\mathcal{R})$ where each element of $\mathsf{Per}(\mathcal{R})$ satisfies specific algebraic properties stated below, allows the public keys and secret keys to be updated in our uhqcPKE via a transformation while preserving correctness and security. Messages are mapped to elements of $\mathsf{Per}(\mathcal{R})$ by a deterministic sampling algorithm Samp.

Assumption (Informal): Let $(\Pi, \mathsf{F}) \in \mathsf{Per}(\mathcal{R})$. We assume that an element of $\mathsf{Per}(\mathcal{R})$ has the following specific properties:

(i) F and Π preserve Hamming weight.

(ii) $\mathsf{F}(\mathbf{x} + \mathbf{hy}) = \mathsf{F}(\mathbf{x}) + \Pi(\mathbf{h})\mathsf{F}(\mathbf{y})$ for all $\mathbf{h}, \mathbf{x}, \mathbf{y} \in \mathcal{R}$.

(iii) The set $\mathsf{Per}(\mathcal{R})$ have sufficient entropy such that the output of Samp mimics random sampling.

Table 1: Storage and communication cost of asynchronous post-quantum IND-CR-CPA secure uPKE schemes.

Scheme	$\lvert\mathsf{sk}\rvert$	$\lvert\mathsf{pk}\rvert$	$\lvert\mathsf{ct}\rvert$	$\lvert\mathsf{up}\rvert$	Assumption	UB
[11]	m	$(nm+n)\log q$	$(m+1)\log q$	$m(m+1)\log q$	LWE	Bounded
[19]	$n\log q$	$(n^2+n)\log q$	$(n^2+n)\log q$	$(n^2+n)\log q$	LWE	Bounded
[2]	n	$(nk+k)\log q + \mathcal{O}(\lambda)$	$(n+1)\log q$	$\frac{\lambda}{\log p}(n+1)\log q$	Hollow LWE and PCE	Unbounded
uhqcPKE	$2\hat{n}$	$2\hat{n}$	$2\hat{n}$	$2\hat{n}$	DQCSDP	Unbounded

λ is the security parameter, q the modulus in lattice-based schemes, p message modulus, n lattice dimension, $\hat{n}$ code length, k dimension of hollow code, $m(>n)$ is the number of columns of the LWE matrix, UB: update bound, $\lvert\mathsf{sk}\rvert$: secret key size, $\lvert\mathsf{pk}\rvert$: public key size, $\lvert\mathsf{ct}\rvert$: ciphertext size, $\lvert\mathsf{up}\rvert$: update ciphertext size, LWE : learning with errors, PCE: permutation code equivalence, DQCSDP: decisional quasi-cyclic syndrome decoding with parity

– Table 1 provides a comparative analysis of our uhqcPKE with existing asynchronous post-quantum secure uPKE schemes in terms of storage and communication overhead. The isogeny-based uPKE constructions ([13,12]) do not support the asynchronous property unlike our uhqcPKE. Moreover, the uPKE scheme of Duparc *et al.* [12] achieves only OW-CR-CPA security, a comparatively weaker security notion as compared to IND-CR-CPA security of our uhqcPKE. The SIDH based construction of Eaton *et al.* [13] is subsequently broken by attacks ([9,23,26]). In contrast to the lattice-based constructions ([11,19,2]), our candidate relies on DQCSDP assumption based on linear codes. As summarized in Table 1, the existing lattice-based uPKE protocols are more expensive in terms of public key size, ciphertext size as well as updated ciphertext size as compared to our uhqcPKE. When $\hat{n} = n$ for our uhqcPKE the secret key size $\lvert\mathsf{sk}\rvert = 2n$ is comparable to others but in terms of $\lvert\mathsf{pk}\rvert, \lvert\mathsf{ct}\rvert, \lvert\mathsf{up}\rvert$ are linear while the others are more expensive.
The overall computation cost of the uhqcPKE is comparable as compared to the lattice-based uPKE as described in Table 2 in terms of **KeyGen**, **Enc**, **Dec**, **SkUpdate** cost while uhqcPKE performs better in terms of **PkUpdate** cost compared to other lattice-based schemes. In the Table 2, we have not included the decoding cost of uhqcPKE as it varies over codes. More favorably, our scheme uhqcPKE affords unbounded key updates whereas the works in [11,19] primarily focus on only bounded number of updates due to noise management issues after certain number of updates. In contrast,

Table 2: Computation cost of existing asynchronous post-quantum IND-CR-CPA secure uPKE schemes.

Scheme	KeyGen	Enc	Dec	PkUpdate	SkUpdate
[11]	$\mathcal{O}(nm)$	$\mathcal{O}(nm)$	$\mathcal{O}(m)$	$\mathcal{O}(nm^2)$	$\mathcal{O}(m^2)$
[19]	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$
[2]	$\mathcal{O}(nk)$	$\mathcal{O}(nk)$	$\mathcal{O}(n)$	$\mathcal{O}(n^3 + \frac{\lambda}{\log p}nk)$	$\mathcal{O}(n^2)$
Our uhqcPKE	$\mathcal{O}(\hat{n}^2)$	$\mathcal{O}(\hat{n}^2)$	$\mathcal{O}(\hat{n}^2)$	$\mathcal{O}(\hat{n}^2)$	$\mathcal{O}(\hat{n}^2)$

λ is the security parameter, p the message modulus, n the lattice dimension, $m(> n)$ the number of columns of the LWE matrix, k the hollow code dimension, $\hat{n}$ code length. All costs are reported in Big-$\mathcal{O}$ notation.

our candidate employs a structured weight preserving function which allows to update the public key and secret key without any noise accumulation. Albrecht *et al.* [2] addresses the noise-management issues by incorporating a permutation matrix and an invertible matrix into the update mechanism. However the concrete performance of their uPKE remains low for realistic choices of parameters.

– We provide a rigorous security analysis and formally establish the IND-CR-CPA security of our uhqcPKE under the 2-DQCSDP and the 3-DQCSDP assumptions. The IND-CR-CPA security model between a challenger and an adversary captures the forward security which ensures that compromise of secret keys at the current epoch does not endanger the confidentiality of past ciphertexts. The security experiment shows that the scheme will remain secure even if the keys are periodically updated and at some point of time the secret key is exposed. More formally, we prove the following Theorem.

Theorem (Informal) *Our proposed* uhqcPKE *is* IND-CR-CPA *under the hardness of the* 2-DQCSDP *and the* 3-DQCSDP *assumptions.*

1.2 Technical Overview

We construct an IND-CR-CPA secure updatable encryption scheme uhqcPKE based on IND-CPA secure hqcPKE of [15]. Our construction follows the general uPKE framework where key updates can be performed asynchronously by any sender. To enable the update functionality, we use a structured permutation set $\mathsf{Per}(\mathcal{R})$ where $\mathcal{R} = \mathbb{F}_2[x]/(x^n - 1)$. Each element (Π, F) of $\mathsf{Per}(\mathcal{R})$ satisfies few specific properties as stated in Section 1.1.

We use a deterministic sampling function $\mathsf{Samp} : \mathbb{F}_2^k \longrightarrow \mathsf{Per}(\mathcal{R})$ to sample an element (Π, F) of $\mathsf{Per}(\mathcal{R})$ using a vector $\boldsymbol{\mu}_i$ from the message space $\mathbb{F}_2^k$, use it to update the public key at epoch i and encrypt $\boldsymbol{\mu}_i$ to form an update ciphertext up_{i+1}. To update a secret key at epoch i the update ciphertext up_{i+1} is de-

crypted using the secret key sk_i at epoch i to recover the randomness $\boldsymbol{\mu}_i$ and use $\mathsf{Samp}(\boldsymbol{\mu}_i) \in \mathsf{Per}(\mathcal{R})$ to update the secret key at epoch $i+1$. More explicitly, a user samples its initial secret key $\mathsf{sk}_0 = (\mathbf{x}_0, \mathbf{y}_0)$ by selecting $\mathbf{x}_0, \mathbf{y}_0 \xleftarrow{\$} \mathcal{S}_w^n(\mathbb{F}_2)$ and computes the corresponding initial public key $\mathsf{pk}_0 = (\mathbf{h}_0, \mathbf{s}_0 = \mathbf{x}_0 + \mathbf{h}_0\mathbf{y}_0) \in \mathcal{R}\times\mathcal{R}$ at epoch 0 where $\mathcal{S}_w^n(\mathbb{F}_2) = \{\mathbf{v} \in \mathbb{F}_2^n \mid \mathsf{wt}(\mathbf{v}) = w\}$ and $\mathbf{h}_0 \xleftarrow{\$} \mathcal{R}$ is a uniformly random public polynomial. To encrypt a message $\mathbf{m} \in \mathbb{F}^k$ under the public key $\mathsf{pk}_i = (\mathbf{h}_i, \mathbf{s}_i)$ at epoch i, the sender computes the $\mathsf{uhqcPKE}$ ciphertext $\mathsf{CT}_{\mathbf{m}} = \big(\mathbf{u}_{\mathbf{m}} = \mathbf{r}_1 + \mathbf{h}_i\mathbf{r}_2,\ \mathbf{v}_{\mathbf{m}} = \mathbf{mG} + \mathbf{s}_i\mathbf{r}_2 + \mathbf{e}\big)$ for some random $(\mathbf{r}_1, \mathbf{r}_2) \in \mathcal{S}_{w_r}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2)$, $\mathbf{e} \in \mathcal{S}_{w_e}^n(\mathbb{F}_2)$ where $\mathbf{G}$ denotes the public generator matrix of the $[n, k]$ linear code, w, w_r, w_e are fixed weights. To update the public key pk_i at epoch i, a sender samples a randomness $\boldsymbol{\mu}_i \xleftarrow{\$} \mathbb{F}_2^k$ and encrypts it under $\mathsf{pk}_i = (\mathbf{h}_i, \mathbf{s}_i)$ to generate the update ciphertext $\mathsf{up}_{i+1} = \mathsf{CT}_{\boldsymbol{\mu}_i} = \big(\mathbf{u}_{\boldsymbol{\mu}_i} = \mathbf{r}_1' + \mathbf{h}_i\mathbf{r}_2',\ \mathbf{v}_{\boldsymbol{\mu}_i} = \boldsymbol{\mu}_i\mathbf{G} + \mathbf{s}_i\mathbf{r}_2' + \mathbf{e}'\big)$ for randomly chosen $(\mathbf{r}_1', \mathbf{r}_2') \in \mathcal{S}_{w_r}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2)$, $\mathbf{e}' \in \mathcal{S}_{w_e}^n(\mathbb{F}_2)$. The updated public key is then computed as $\mathsf{pk}_{i+1} = (\mathbf{h}_{i+1}, \mathbf{s}_{i+1})$ by setting $\mathbf{h}_{i+1} = \Pi_{\boldsymbol{\mu}_i}(\mathbf{h}_i)$, $\mathbf{s}_{i+1} = \mathsf{F}_{\boldsymbol{\mu}_i}(\mathbf{s}_i)$ and generating $\mathsf{Samp}(\boldsymbol{\mu}_i) = (\Pi_{\boldsymbol{\mu}_i}, \mathsf{F}_{\boldsymbol{\mu}_i}) \in \mathsf{Per}(\mathcal{R})$. Upon receiving $(\mathsf{pk}_{i+1}, \mathsf{up}_{i+1})$, the receiver decrypting up_{i+1}, recovers the update randomness $\boldsymbol{\mu}_i$, generates $\mathsf{Samp}(\boldsymbol{\mu}_i) = (\Pi_{\boldsymbol{\mu}_i}, \mathsf{F}_{\boldsymbol{\mu}_i}) \in \mathsf{Per}(\mathcal{R})$ and updates the old secret key $\mathsf{sk}_i = (\mathbf{x}_i, \mathbf{y}_i)$ by new secret key $\mathsf{sk}_{i+1} = (\mathbf{x}_{i+1}, \mathbf{y}_{i+1})$ by setting $\mathbf{x}_{i+1} = \mathsf{F}_{\boldsymbol{\mu}_i}(\mathbf{x}_i)$, $\mathbf{y}_{i+1} = \mathsf{F}_{\boldsymbol{\mu}_i}(\mathbf{y}_i)$. The weight preserving property of Π, F and the property $\mathsf{F}(\mathbf{x} + \mathbf{hy}) = \mathsf{F}(\mathbf{x}) + \Pi(\mathbf{h})\mathsf{F}(\mathbf{y})$ for all $\mathbf{x}, \mathbf{y}, \mathbf{h} \in \mathcal{R}$ keeps the distribution of the public-secret key pair unaltered.

We prove that our $\mathsf{uhqcPKE}$ is $\mathsf{IND\text{-}CR\text{-}CPA}$ secure via a game-based reduction under the hardness assumptions of the 2-DQCSDP and the 3-DQCSDP. The proof proceeds through a sequence of hybrid games in which the adversary's view is modified in a gradual manner. For every pair of consecutive games, we show that no efficient adversary can distinguish them. We argue that any adversary capable of distinguishing two adjacent games can be transformed into a distinguisher that breaks either the 2-DQCSDP assumption or the 3-DQCSDP assumption. In particular, if we consider a 2-DQCSDP instance $(\mathbf{H}, \mathbf{s})$ where $\mathbf{H} = (\mathbf{I}_n\ \mathsf{rot}(\mathbf{h}))$ is an $n \times 2n$ matrix and $\mathbf{s}$ is a vector of length n. From this instance, we derive the public key $\mathsf{pk}_0 = (\mathbf{h}, \mathbf{s}) = (\mathbf{h}_0, \mathbf{s}_0)$ where $\mathbf{h}, \mathbf{s} \in \mathcal{R}$ and simulate an adversary that attempts to distinguish two consecutive games. If such an adversary succeeds, then we obtain a distinguisher that breaks the 2-DQCSDP assumption. Consequently, the hardness of 2-DQCSDP implies that the corresponding games are computationally indistinguishable. Similarly, we consider a 3-DQCSDP instance $(\mathbf{H}, (\mathbf{a}_1, \mathbf{a}_2))$ where $\mathbf{H}$ is a $2n \times 3n$ matrix of the form $\mathbf{H} = \begin{pmatrix} \mathbf{I}_n & \mathbf{0} & \mathsf{rot}(\mathbf{h}) \\ \mathbf{0} & \mathbf{I}_n & \mathsf{rot}(\mathbf{s}) \end{pmatrix}$ and $(\mathbf{a}_1, \mathbf{a}_2)$ is a vector of length $2n$ with $\mathbf{a}_1 = \mathbf{r}_1 + \mathbf{hr}_2 \in \mathcal{R}$ and $\mathbf{a}_2 = \mathbf{sr}_2 + \mathbf{e} \in \mathcal{R}$ for some unknown $\mathbf{r}_1, \mathbf{r}_2 \in \mathcal{R}$ and $\mathbf{e} \in \mathcal{R}$ satisfying a fixed parity condition. For a message $\mathbf{m}$, we define the ciphertext as $\mathsf{CT} = (\mathbf{a}_1, \mathbf{mG} + \mathbf{a}_2) = (\mathbf{r}_1 + \mathbf{hr}_2,\ \mathbf{mG} + \mathbf{sr}_2 + \mathbf{e})$. The resulting distribution of CT is identical to that of an honestly generated ciphertext. We exploit this property to simulate the challenge ciphertext in the security experiment. We show that if there exists an adversary capable of distinguishing two

adjacent games in this hybrid sequence, then one can construct a distinguisher that breaks the 3-DQCSDP assumption. Hence, the hardness of 3-DQCSDP guarantees that the corresponding games are computationally indistinguishable. In the final game, the challenge ciphertext is statistically independent of the challenge bit b and therefore combining all the arguments, we conclude that our uhqcPKE is IND-CR-CPA secure.

2 Preliminaries

Notations. Let $\mathbb{Z}$ be the set of integers, $\mathbb{F}$ be finite field and $\mathbb{F}_2$ be the finite binary field. The Hamming weight of a vector, denoted by $\mathsf{wt}(\cdot)$, refers to the number of non-zero entries in that vector. We define the set of all binary vectors of length n with Hamming weight exactly w as $\mathcal{S}_w^n(\mathbb{F}_2) = \{\mathbf{v} \in \mathbb{F}_2^n \mid \mathsf{wt}(\mathbf{v}) = w\}$. The notation $x \xleftarrow{\$} \mathcal{S}$ indicates that x is sampled uniformly at random from the set $\mathcal{S}$. The transpose of a vector $\mathbf{u}$ is denoted by $\mathbf{u}^T$. Also for any two functions f, g, we denote the composition by $f \circ g = f(g)$. Let $\mathcal{R} = \mathbb{F}_2[x]/(x^n-1)$ denote the quotient ring of polynomials where $n \in \mathbb{Z}^+$. We use bold lowercase or uppercase letters to denote vectors, polynomials, or matrices. A prime n is primitive if $(x^n - 1)/(x - 1)$ is irreducible in $\mathcal{R}$. An element $\mathbf{u} \in \mathcal{R}$ is often denoted as the vector $\mathbf{u} = (u_0, u_1, \ldots, u_{n-1}) \in \mathbb{F}_2^n$ where u_i denotes the coefficient of x^i for $i = 0, 1, \ldots, n-1$. For any $\mathbf{u}, \mathbf{v} \in \mathcal{R}$, where $\mathbf{u} = (u_0, u_1, \ldots, u_{n-1}) \in \mathbb{F}_2^n$ and $\mathbf{v} = (v_0, v_1, \ldots, v_{n-1}) \in \mathbb{F}_2^n$ we define their product $\mathbf{uv} = \mathbf{w} \in \mathcal{R}$, where $\mathbf{w} = (w_0, w_1, \ldots, w_n)$ is given component-wise as $w_k = \sum_{i+j \equiv k \pmod n} u_i v_j$ for $k \in \{0, 1, \ldots, n-1\}$.

2.1 Basics Related to Coding Theory

Definition 1 (Circulant Matrix [24]). *Let $\mathbf{v} = (v_0, v_1, \ldots, v_{n-1}) \in \mathbb{F}_2^n$. The* circulant matrix *generated by $\mathbf{v}$, denoted $\mathsf{rot}(\mathbf{v})$, is defined as*

$$\mathsf{rot}(\mathbf{v}) = \begin{pmatrix} v_0 & v_{n-1} & \cdots & v_1 \\ v_1 & v_0 & \cdots & v_2 \\ \cdots & \cdots & \cdots & \cdots \\ v_{n-1} & v_{n-2} & \cdots & v_0 \end{pmatrix} \in \mathbb{F}_2^{n \times n}.$$

An important consequence of this construction is that the product of any two vectors $\mathbf{u}, \mathbf{v} \in \mathcal{R}$ can be represented as a standard matrix–vector product involving the circulant form. That is, $\mathbf{uv} = \mathbf{u} \times \mathsf{rot}(\mathbf{v})^T = \left(\mathsf{rot}(\mathbf{u}) \times \mathbf{v}^T\right)^T = \mathbf{v} \times \mathsf{rot}(\mathbf{u})^T = \mathbf{vu}$. Here, '$\times$' denotes the usual matrix multiplication. This equivalence reflects the commutativity of polynomial multiplication in $\mathcal{R}$ and allows efficient implementation via matrix operations.

Definition 2 (Linear Code [22]). *A linear $[n, k]$ code C of length n and dimension k is a subspace of the vector space $\mathbb{F}^n$ of dimension k over the field $\mathbb{F}$. The elements of C are referred to as codewords. An $[n, k]$ linear code is said to be a binary linear code if $\mathbb{F} = \mathbb{F}_2$.*

Definition 3 (Generator Matrix [22]). *A matrix* $\mathbf{G} \in \mathbb{F}_2^{k\times n}$ *is said to be a generator matrix for the* $[n,k]$ *binary linear code* C *if* $\mathsf{C} = \left\{\mathbf{mG} \;\middle|\; \mathbf{m} \in \mathbb{F}_2^k\right\}$.

Definition 4 (Parity-Check Matrix [22]). *Let* C *be an* $[n,k]$ *linear code over* $\mathbb{F}_2$. *A matrix* $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$ *is said to be a parity-check matrix of* C *if* C *is the kernel of* $\mathbf{H}$ *i.e.,* $\mathsf{C} = \left\{\mathbf{v} \in \mathbb{F}_2^n \;\middle|\; \mathbf{Hv}^T = \mathbf{0}\right\}$. *Equivalently,* $\mathbf{H}$ *is a generator matrix for the dual code* $\mathsf{C}^\perp$ *where* $\mathsf{C}^\perp = \left\{\mathbf{uH} \;\middle|\; \mathbf{u} \in \mathbb{F}_2^{n-k}\right\}$.

Definition 5 (Syndrome [22]). *Let* C *be an* $[n,k]$ *linear code over* $\mathbb{F}_2$ *and* $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$ *be a parity-check matrix of* C. *For any vector* $\mathbf{v} \in \mathbb{F}_2^n$, *the* syndrome *of* $\mathbf{v}$ *is defined as* $\mathbf{s} = \mathbf{Hv}^T$. *A vector* $\mathbf{v}$ *is a codeword in* C *if and only if its syndrome is zero, i.e.,* $\mathbf{v} \in \mathsf{C} \iff \mathbf{Hv}^T = \mathbf{0}$.

Definition 6 (Minimum Distance [22]). *Let* C *be an* $[n,k]$ *linear code over* $\mathcal{R} = \mathbb{F}_2[x]/(x^n-1)$ *and* $\mathsf{wt}(\cdot)$ *denote a weight function (typically the Hamming weight) on* $\mathcal{R}$. *The* minimum distance *of* C *is defined as*

$$d = \min_{\substack{\mathbf{u},\mathbf{v}\in\mathsf{C}\\ \mathbf{u}\neq\mathbf{v}}} \mathsf{wt}(\mathbf{u}-\mathbf{v}).$$

A linear code of length n, dimension k and minimum distance d is denoted by $[n,k,d]$. Such a code can uniquely correct any error pattern affecting up to $\tau = \left\lfloor \frac{d-1}{2} \right\rfloor$ positions.

Definition 7 (Quasi-Cyclic Codes [25]). *Let* $\mathbf{c} = (\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_{s-1}) \in \mathbb{F}_2^{sn}$, *where each block* $\mathbf{c}_i \in \mathbb{F}_2^n$ *for* $0 \le i < s$. *A linear code* $\mathsf{C} \subseteq \mathbb{F}_2^{sn}$ *of length* sn, *dimension* k *and minimum distance* d *denoted* $[sn,k,d]$ *is said to be a quasi-cyclic (QC) code of index* s *if for every codeword* $\mathbf{c} \in \mathsf{C}$ *the vector obtained by applying a simultaneous cyclic shift to each block* $\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_{s-1}$ *also belongs to* C.

Alternatively, viewing each block $\mathbf{c}_i$ *as a polynomial in the ring* $\mathcal{R} = \mathbb{F}_2[x]/(x^n-1)$, *the code* C *is quasi-cyclic of index* s *if for any codeword* $\mathbf{c} = (\mathbf{c}_0, \ldots, \mathbf{c}_{s-1}) \in \mathsf{C}$, *the tuple* $(x\mathbf{c}_0, \ldots, x\mathbf{c}_{s-1})$ *also lies in* C.

Definition 8 (Systematic Quasi-Cyclic Codes [18]). *A systematic quasi-cyclic(*$[sn,n]$*) code of index* s *and rate* $\frac{1}{s}$ *is a quasi-cyclic code with an* $(s-1)n \times sn$ *parity-check matrix of the form:*

$$\begin{bmatrix} \mathbf{I}_n & 0 & \cdots & 0 & \mathbf{A}_0 \\ 0 & \mathbf{I}_n & \cdots & 0 & \mathbf{A}_1 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \mathbf{I}_n & \mathbf{A}_{s-2} \end{bmatrix}$$

where, $\mathbf{A}_i$ *for* $i \in \{0,1,\ldots,s-2\}$ *are circulant matrix.*

Remark 1 *The definition of systematic quasi-cyclic codes of index* s *can be generalized to all rates* ℓ/s *for* $\ell = 1,2,\ldots,s-1$.

2.2 Hardness Assumptions

This section establishes the computational foundations for the security of the cryptographic protocols.

Definition 9 (Syndrome Decoding (SD) **Distribution [7]).** *The* $\mathsf{SD}(n,k,w)$ *distribution parameterized by positive integers* n*,* k *and* w *is defined as the distribution over pairs* $(\mathbf{H},\mathbf{y})$ *where* $\mathbf{H} \leftarrow \mathbb{F}_2^{(n-k)\times n}$ *is sampled randomly and the syndrome* $\mathbf{y}$ *satisfies* $\mathbf{y}^T = \mathbf{H}\mathbf{x}^T$ *which is computed by sampling a random vector* $\mathbf{x} \xleftarrow{\$} \mathbb{F}_2^n$ *such that* $\mathsf{wt}(\mathbf{x}) = w$*.*

Definition 10 (Computational $\mathsf{SD}(\mathsf{CSD})$ **Problem [14]).** *Let* n,k,w *be positive integers. The* $\mathsf{CSD}(n,k,w)$ *problem parametrized by* n,k,w *states that given a pair* $(\mathbf{H},\mathbf{y}) \in \mathbb{F}_2^{(n-k)\times n} \times \mathbb{F}_2^{n-k}$ *sampled according to the distribution* $\mathsf{SD}(n,k,w)$ *defined in definition 9, find a vector* $\mathbf{x} \in \mathbb{F}_2^n$ *such that* $\mathbf{H}\mathbf{x}^T = \mathbf{y}^T$ *and* $\mathsf{wt}(\mathbf{x}) = w$*.*

The computational version of the SD problem is known to be NP-complete in the Hamming metric, as shown by Berlekamp, McEliece and van Tilborg [6].

Definition 11 (Decisional SD (DSD) **Problem [14]).** *The decisional syndrome decoding* $\mathsf{DSD}(n,k,w)$ *problem parametrized by* n,k,w *is defined as the problem of deciding, whether a pair* $(\mathbf{H},\mathbf{y}) \in \mathbb{F}_2^{(n-k)\times n} \times \mathbb{F}_2^{n-k}$ *is sampled from the distribution* $\mathsf{SD}(n,k,w)$ *or from the uniform distribution over* $\mathbb{F}_2^{(n-k)\times n} \times \mathbb{F}_2^{n-k}$ *with non-negligible advantage.*

Proposition 1. *Let* $\mathbf{f}(x) = \sum_{i=0}^{n-1} f_i x^i \in \mathcal{R}$ *be a polynomial where* $\mathcal{R} = \mathbb{F}_2[x]/(x^n - 1)$*. We define its parity as* $\mathrm{par}(\mathbf{f}) = \mathbf{f}\cdot\mathbf{1} = \sum_{i=0}^{n-1} f_i \bmod 2$*. Then, for all* $\mathbf{h} = \mathbf{h}(x), \mathbf{y} = \mathbf{y}(x) \in \mathcal{R}$*, we have* $\mathrm{par}(\mathbf{hy}) = \mathrm{par}(\mathbf{h}) \cdot \mathrm{par}(\mathbf{y}) \bmod 2$*. Moreover, if* $\mathbf{x},\mathbf{y},\mathbf{h} \in \mathcal{R}$ *satisfy* $\mathrm{wt}(x) = \mathrm{wt}(y) = w$ *and* $\mathrm{par}(\mathbf{h}) = b_1$ *and if* $\mathbf{s} = \mathbf{x} + \mathbf{hy}$*, then* $\mathrm{par}(\mathbf{s}) = w + b_1 \times w \bmod 2$*.*

Proof. Let $\mathbf{h}(x) = \sum_{i=0}^{n-1} h_i x^i \in \mathcal{R}$ and $\mathbf{y}(x) = \sum_{j=0}^{n-1} y_j x^j \in \mathcal{R}$ where $h_i, y_j \in \mathbb{F}_2$ and $\mathbf{hy} = \sum_{k=0}^{n-1} c_k x^k \in \mathcal{R}$ with

$$c_k = \sum_{i+j \equiv k \pmod n} h_i y_j \in \mathbb{F}_2.$$

Now,

$$\mathrm{par}(\mathbf{hy}) = \sum_{k=0}^{n-1} c_k = \sum_{k=0}^{n-1} \sum_{i+j\equiv k} h_i y_j \bmod 2.$$

Since each ordered pair $(i,j) \in \{0,\ldots,n-1\}^2$ appears exactly once in the above sum, this simplifies to

$$\mathrm{par}(\mathbf{hy}) = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} h_i y_j \bmod 2$$

which factors as

$$(\sum_{i=0}^{n-1} h_i)(\sum_{j=0}^{n-1} y_j) \bmod 2.$$

Hence $\mathrm{par}(\mathbf{hy}) = \mathrm{par}(\mathbf{h}) \cdot \mathrm{par}(\mathbf{y}) \bmod 2$.

For the second claim, parity is linear over $\mathbb{F}_2$ as $\mathrm{par}(\mathbf{x}+\mathbf{y}) = (\mathbf{x}+\mathbf{y})\cdot\mathbf{1} = \mathbf{x}\cdot\mathbf{1} + \mathbf{y}\cdot\mathbf{1} = \mathrm{par}(\mathbf{x}) + \mathrm{par}(\mathbf{y})$ and thus $\mathrm{par}(\mathbf{s}) = \mathrm{par}(\mathbf{x}) + \mathrm{par}(\mathbf{hy}) \bmod 2$. Since $\mathrm{par}(\mathbf{x}) = \mathrm{wt}(\mathbf{x}) \equiv w \bmod 2$ and $\mathrm{par}(\mathbf{hy}) = \mathrm{par}(\mathbf{h}) \cdot \mathrm{par}(\mathbf{y}) = b_1 w \bmod 2$, we conclude $\mathrm{par}(\mathbf{s}) = w + b_1 w = w(1+b_1) \bmod 2$.

We use the following notations

- $\mathbb{F}_{2,b_1}^n = \{\mathbf{h} \in \mathbb{F}_2^n \mid \mathbf{h}\cdot\mathbf{1} = b_1 \mod 2\}$ $b_1 \in \{0,1\}$.
- $\mathbb{F}_{2,b_1}^{n\times 2n} = \left\{\mathbf{H} = [\mathbf{I}_n \mid \mathsf{rot}(\mathbf{h})] \in \mathbb{F}_2^{n\times 2n} \;\middle|\; \mathbf{h} \in \mathbb{F}_{2,b_1}^n\right\}$ for $b_1 \in \{0,1\}$.
- $\mathbb{F}_{2,b_1,b_2}^{2n\times 3n} = \left\{\mathbf{H} = \begin{bmatrix}\mathbf{I}_n & \mathbf{0} & \mathsf{rot}(\mathbf{h}_1)\\ \mathbf{0} & \mathbf{I}_n & \mathsf{rot}(\mathbf{h}_2)\end{bmatrix} \in \mathbb{F}_2^{2n\times 3n} \;\middle|\; \mathbf{h}_1 \in \mathbb{F}_{2,b_1}^n,\ \mathbf{h}_2 \in \mathbb{F}_{2,b_2}^n\right\}$ for $b_1, b_2 \in \{0,1\}$.

The first denotes a set of binary vectors of length n whose entries sum to a parity $b_1 \in \{0,1\}$. Here, $\mathbf{1}$ denotes the vector whose all entries are 1 and of length n and $\mathsf{rot}(\mathbf{h})$, $\mathsf{rot}(\mathbf{h}_1)$, $\mathsf{rot}(\mathbf{h}_2)$ denote the circulant matrices generated by the vector $\mathbf{h}$, $\mathbf{h}_1$, $\mathbf{h}_2$ respectively.

Definition 12 (2-Quasi-Cyclic Syndrome Decoding with Parity (2-QCSDP) **Distribution [24]).** *Let n, w be positive integers and let $b_1 \in \{0,1\}$ be a parity bit. Define $b_2 = (w + b_1 w) \bmod 2$. The* 2-$\mathsf{QCSDP}(n, w, b_1, b_2)$ *problem is defined as the distribution over pairs $(\mathbf{H}, \mathbf{y})$ where, $\mathbf{H} \leftarrow \mathbb{F}_{2,b_1}^{n\times 2n}$ is sampled randomly and the syndrome $\mathbf{y}$ satisfies $\mathbf{y}^T = \mathbf{H}\mathbf{x}^T$ which is computed by sampling $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2) \in \mathbb{F}_2^{2n}$ such that $\mathsf{wt}(\mathbf{x}_1) = \mathsf{wt}(\mathbf{x}_2) = w$.*

Definition 13 (2-Decisional Quasi-Cyclic Syndrome Decoding with Parity (2-DQCSDP) **Problem [24]).** *Let n, w be positive integers, and let $b_1 \in \{0,1\}$. Define $b_2 = (w + b_1 w) \bmod 2$. The* 2-$\mathsf{DQCSDP}(n, w, b_1, b_2)$ *problem states that given a pair $(\mathbf{H}, \mathbf{y}) \in \mathbb{F}_{2,b_1}^{n\times 2n} \times \mathbb{F}_{2,b_2}^n$, decide with non-negligible advantage whether $(\mathbf{H}, \mathbf{y})$ is sampled from the distribution* 2-$\mathsf{QCSDP}(n, w, b_1, b_2)$ *or from the uniform distribution over $\mathbb{F}_{2,b_1}^{n\times 2n} \times \mathbb{F}_{2,b_2}^n$.*

More formally, the 2-DQCSDP problem is said to be hard if for any PPT distinguisher $\mathcal{D}$, the advantage $\mathsf{Adv}_{\mathcal{D}}^{\text{2-DQCSDP}}(1^\lambda)$ in solving the 2-DQCSDP problem is negligible, i.e.,

$\mathsf{Adv}_{\mathcal{D}}^{\text{2-DQCSDP}}(1^\lambda) = \mid \Pr[\mathcal{D}(\mathbf{H}, \mathbf{y} = \mathbf{x}\mathbf{H}^T) = 1 \mid \mathbf{H} \in \mathbb{F}_{2,b_1}^{n\times 2n}, \mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2) \in \mathcal{S}_w^n(\mathbb{F}_2) \times \mathcal{S}_w^n(\mathbb{F}_2)] - \Pr[\mathcal{D}(\mathbf{H}, \mathbf{y}) = 1 \mid \mathbf{H} \xleftarrow{\$} \mathbb{F}_{2,b_1}^{n\times 2n}, \mathbf{y} \xleftarrow{\$} \mathbb{F}_{2,b_2}^n] \mid < \varepsilon$ for some negligible function ε.

Definition 14 (3-Quasi Cyclic Syndrome Decoding with Parity (3-QCSDP) Distribution [24]). *Let n, w, b_1, b_2 be positive integers and define $b_3 = (w + b_1 w) \bmod 2$. The 3-$\mathsf{QCSDP}(n, w, b_1, b_2)$ distribution is defined as the distribution over pairs $(\mathbf{H}, (\mathbf{y}_1, \mathbf{y}_2))$ where $\mathbf{H} \xleftarrow{\$} \mathbb{F}_{2,b_1,b_2}^{2n \times 3n}$ is sampled randomly and the syndrome $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2)$ with $(\mathbf{y}_1, \mathbf{y}_2) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$ satisfies $\mathbf{y}^T = \mathbf{H}\mathbf{x}^T$ which is computed by sampling a random vector $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \xleftarrow{\$} \mathbb{F}_2^{3n}$ such that $\mathsf{wt}(\mathbf{x}_1) = \mathsf{wt}(\mathbf{x}_2) = \mathsf{wt}(\mathbf{x}_3) = w$.*

Definition 15 (3-Decisional Quasi-Cyclic Syndrome Decoding problem with Parity (3-DQCSDP) Problem [24]). *Let n, w, b_1, b_2 be positive integers and define $b_3 = (w + b_1 w) \bmod 2$ and $b_3' = (w + b_2 w) \bmod 2$. The 3-$\mathsf{DQCSDP}(n, w, b_1, b_2)$ problem states that given a pair $(\mathbf{H}, (\mathbf{y}_1, \mathbf{y}_2)) \in \mathbb{F}_{2,b_1,b_2}^{2n \times 3n} \times \left(\mathbb{F}_{2,b_3}^n \times \mathbb{F}_{2,b_3'}^n\right)$, decide with non-negligible advantage whether the pair is sampled from the 3-$\mathsf{QCSDP}(n, w, b_1, b_2)$ distribution or from the uniform distribution over $\mathbb{F}_{2,b_1,b_2}^{2n \times 3n} \times \left(\mathbb{F}_{2,b_3}^n \times \mathbb{F}_{2,b_3'}^n\right)$.*

More formally, the 3-DQCSDP problem is said to be hard if for any PPT distinguisher $\mathcal{D}$, the advantage $\mathsf{Adv}_{\mathcal{D}}^{\text{3-DQCSDP}}(1^\lambda)$ in solving the 3-DQCSDP problem is negligible, i.e.,

$\mathsf{Adv}_{\mathcal{D}}^{\text{3-DQCSDP}}(1^\lambda) = \mid \Pr[\mathcal{D}(\mathbf{H}, \mathbf{y} = \mathbf{x}\mathbf{H}^T) = 1 \mid \mathbf{H} \in \mathbb{F}_{2,b_1,b_2}^{2n \times 3n}, \mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \in \mathcal{S}_w^n(\mathbb{F}_2) \times \mathcal{S}_w^n(\mathbb{F}_2) \times \mathcal{S}_w^n(\mathbb{F}_2)] - \Pr[\mathcal{D}(\mathbf{H}, \mathbf{y}) = 1 \mid \mathbf{H} \xleftarrow{\$} \mathbb{F}_{2,b_1,b_2}^{2n \times 3n}, \mathbf{y} \xleftarrow{\$} \mathbb{F}_{2,b_2}^{2n}] \mid < \varepsilon$ for some negligible function ε.

2.3 Public Key Encryption

Definition 16 (Public Key Encryption [10]). *A public key encryption (PKE) scheme consists of the following four PPT algorithms $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ associated with a public key space $\mathcal{PKS}$, secret key space $\mathcal{SKS}$, message space $\mathcal{MS}$ and a ciphertext space $\mathcal{CS}$ and randomness space $\mathcal{RS}$.*

- $\mathsf{PKE.Setup}(1^\lambda) \longrightarrow \mathsf{pp}_{\mathsf{PKE}}$*: On input of the security parameter λ, a trusted third party runs this algorithm and publishes public parameters $\mathsf{pp}_{\mathsf{PKE}}$.*
- $\mathsf{PKE.KeyGen}(\mathsf{pp}_{\mathsf{PKE}}) \longrightarrow (\mathsf{pk}_{\mathsf{PKE}}, \mathsf{sk}_{\mathsf{PKE}})$*: A user runs this algorithm on input of the public parameter $\mathsf{pp}_{\mathsf{PKE}}$ and outputs a pair of secret key and public key $(\mathsf{sk}_{\mathsf{PKE}}, \mathsf{pk}_{\mathsf{PKE}}) \in \mathcal{SKS} \times \mathcal{PKS}$. It publishes the public key $\mathsf{pk}_{\mathsf{PKE}}$ while keeping the secret key $\mathsf{sk}_{\mathsf{PKE}}$ secret.*
- $\mathsf{PKE.Enc}(\mathsf{pp}_{\mathsf{PKE}}, \mathsf{pk}_{\mathsf{PKE}}, \mathbf{m}; \theta) \longrightarrow \mathsf{CT}_{\mathbf{m}}$ *: An encryptor runs this PPT algorithm by taking as input the public parameter $\mathsf{pp}_{\mathsf{PKE}}$, public key $\mathsf{pk}_{\mathsf{PKE}}$ and a message $\mathbf{m} \in \mathcal{MS}$ and returns a ciphertext $\mathsf{CT}_{\mathbf{m}} \in \mathcal{CS}$ which is sent through an insecure channel. Here $\theta \in \mathcal{RS}$ is an encryption randomness choosen by the algorithm during encryption.*

– $\mathsf{PKE.Dec}(\mathsf{pp_{PKE}}, \mathsf{sk_{PKE}}, \mathsf{CT_m}) \longrightarrow \mathbf{m}/\perp$: *On input of the public parameter* $\mathsf{pp_{PKE}}$*, the secret key* $\mathsf{sk_{PKE}}$ *and the ciphertet* $\mathsf{CT_m}$*, the decryptor runs this algorithm and recovers either the message* $\mathbf{m}$ *or returns* $\perp$*, a designated symbol indicating failure.*

Correctness: We say that a public key encryption scheme PKE is correct if for any security parameter λ,

$$\Pr\left[\mathsf{PKE.Dec}(\mathsf{pp_{PKE}}, \mathsf{sk_{PKE}}, \mathsf{CT_m}) = \mathbf{m} \;\middle|\; \begin{array}{l} \mathsf{pp_{PKE}} \longleftarrow \mathsf{PKE.Setup}(1^\lambda), \\ (\mathsf{pk_{PKE}}, \mathsf{sk_{PKE}}) \longleftarrow \mathsf{PKE.KeyGen}(\mathsf{pp_{PKE}}), \\ \mathsf{CT_m} \longleftarrow \mathsf{PKE.Enc}(\mathsf{pp_{PKE}}, \mathsf{pk_{PKE}}, \mathbf{m}; \theta) \end{array}\right] = 1.$$

Definition 17 (Indistinguishability under Chosen Plaintext Attack (IND-CPA) **[17]).** *A public key encryption scheme (*PKE*) is said to be IND-CPA secure if for any* PPT *adversary* $\mathcal{A}$*, the advantage*

$$\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{PKE,\mathcal{A}}(1^\lambda) = \left|\Pr\left[\mathsf{EXP}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE},\mathcal{A}}(1^\lambda) = 1\right] - \frac{1}{2}\right| = \left|\Pr\left[b = b'\right] - \frac{1}{2}\right|$$

is negligible, where the IND-CPA security experiment $\mathsf{EXP}^{\mathsf{IND\text{-}CPA}}_{PKE,\mathcal{A}}(1^\lambda)$ *between a challenger* $\mathcal{C}$ *and the adversary* $\mathcal{A}$ *is described in Figure 1.*

Setup Phase: The challenger $\mathcal{C}$ computes a public parameter $\mathsf{pp_{PKE}} \leftarrow \mathsf{PKE.Setup}(1^\lambda)$ and a pair of public-secret key $(\mathsf{pk_{PKE}}, \mathsf{sk_{PKE}}) \leftarrow \mathsf{PKE.KeyGen}(\mathsf{pp_{PKE}})$. It forwards the public parameter $\mathsf{pp_{PKE}}$ and the public key $\mathsf{pk_{PKE}}$ to the adversary $\mathcal{A}$ while keeping the secret key $\mathsf{sk_{PKE}}$ to itself.

Challenge Phase: After receiving $(\mathbf{m}_0^*, \mathbf{m}_1^*) \in \mathcal{MS} \times \mathcal{MS}$ a message pair from the adversary $\mathcal{A}$, the challenger $\mathcal{C}$ uniformly samples $b \xleftarrow{\$} \{0,1\}$ and sends $\mathsf{CT}^*_{\mathbf{m}_b} \leftarrow \mathsf{PKE.Enc}(\mathsf{pp_{PKE}}, \mathsf{pk_{PKE}}, \mathbf{m}_b^*; \theta)$ to $\mathcal{A}$ where $\theta \in \mathcal{RS}$ is an encryption randomness chosen by the algorithm during encryption.

Guess Phase: The adversary $\mathcal{A}$ eventually returns a bit b'. If $b = b'$, the experiment outputs 1, otherwise outputs 0.

Fig. 1: $\mathsf{EXP}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE},\mathcal{A}}(1^\lambda)$: IND-CPA security experiment between a challenger and an adversary

2.4 Hamming Quasi-Cyclic Public Key Encryption (hqcPKE) [24]

The public key encryption scheme $\mathsf{hqcPKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ associated a message space $\mathcal{MS} = \mathbb{F}_2^k$, a ciphertext space $\mathcal{CS} = \mathcal{R} \times \mathcal{R}$, randomness space $\mathcal{RS} = \mathbb{F}_2^k$ and key space (both public key and secret key) $\mathcal{KS} =$

$\mathcal{R} \times \mathcal{R}$. It employs two types of codes—a decodable linear $[n, k]$ code C over $\mathbb{F}_2$ having a generator matrix $\mathbf{G} \in \mathbb{F}_2^{k \times n}$ and error-correcting capacity τ and a random double-circulant $[2n, n]$ code having a parity check matrix in the form $(\mathbf{1}, \mathbf{h})$ consists of four PPT algorithms satisfying the following requirements where $\mathcal{R} = \mathbb{F}_2[x]/(x^n - 1)$ and we use the notation $\mathcal{S}_w^n(\mathbb{F}_2)$ defined as $\mathcal{S}_w^n(\mathbb{F}_2) = \{\mathbf{v} \in \mathbb{F}_2^n \mid \mathsf{wt}(\mathbf{v}) = w\}$.

- $\mathsf{hqcPKE.Setup}(1^\lambda) \longrightarrow \mathsf{pp}_{\mathsf{hqcPKE}}$: Given a security parameter λ, a trusted authority executes this PPT algorithm to generate the public parameters $\mathsf{pp}_{\mathsf{hqcPKE}}$ as follows.
  1. Chooses non-negative integers $n, k, \alpha, \tau, w, w_r, w_e$ where n is the length and k is the dimension of the code. Here τ denotes the error-correcting capability α denotes the length of the encryption randomness and w, w_r, w_e denote the weight of the secret key components and error vectors.
  2. Selects a generator matrix $\mathbf{G} \in \mathbb{F}_2^{k \times n}$ of a decodable $[n, k]$ linear code C with error correcting capability τ and a randomness generator $\mathsf{P} : \{0,1\}^* \longrightarrow \mathcal{S}_{w_e}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2)$.
  3. Publishes the public parameter $\mathsf{pp}_{\mathsf{hqcPKE}} = (n, k, \alpha, \tau, w, w_r, w_e, \mathbf{G}, \mathsf{P})$.

- $\mathsf{hqcPKE.KeyGen}(\mathsf{pp}_{\mathsf{hqcPKE}}) \longrightarrow (\mathsf{pk}_{\mathsf{hqcPKE}}, \mathsf{sk}_{\mathsf{hqcPKE}})$: On input the public parameters $\mathsf{pp}_{\mathsf{hqcPKE}}$, a user executes the key generation algorithm to generate public-secret key pair $\mathsf{pk}_{\mathsf{hqcPKE}}, \mathsf{sk}_{\mathsf{hqcPKE}}$ by performing the following steps.
  1. Samples $\mathbf{h} \xleftarrow{\$} \mathcal{R}$ and $(\mathbf{x}, \mathbf{y}) \xleftarrow{\$} \mathcal{S}_w^n(\mathbb{F}_2) \times \mathcal{S}_w^n(\mathbb{F}_2)$. Note that, $\mathbf{x} \in \mathbb{F}_2^n, \mathbf{y} \in \mathbb{F}_2^n$ with $\mathsf{wt}(\mathbf{x}) = w$ and $\mathsf{wt}(\mathbf{y}) = w$.
  2. Computes $\mathbf{s} = \mathbf{x} + \mathbf{h}\mathbf{y} \in \mathcal{R}$.
  3. Sets $\mathsf{pk}_{\mathsf{hqcPKE}} = (\mathbf{h}, \mathbf{s})$ and $\mathsf{sk}_{\mathsf{hqcPKE}} = (\mathbf{x}, \mathbf{y})$.
  4. Publishes the public key $\mathsf{pk}_{\mathsf{hqcPKE}} = (\mathbf{h}, \mathbf{s})$ and keep the secret $\mathsf{sk}_{\mathsf{hqcPKE}} = (\mathbf{x}, \mathbf{y})$ secret to itself.

- $\mathsf{hqcPKE.Enc}(\mathsf{pp}_{\mathsf{hqcPKE}}, \mathsf{pk}_{\mathsf{hqcPKE}}, \mathbf{m}; \theta) \longrightarrow \mathsf{CT}_{\mathbf{m}}$: An encryptor runs this procedure with input of the public parameter $\mathsf{pp}_{\mathsf{hqcPKE}}$, the public key $\mathsf{pk}_{\mathsf{hqcPKE}}$ and a message $\mathbf{m} \in \mathbb{F}_2^k$ and returns a ciphertext $\mathsf{CT}_{\mathbf{m}}$ by carrying out the following steps.
  1. Samples a random α-bit string $\theta \xleftarrow{\$} \mathbb{F}_2^\alpha$ and generates $\mathsf{P}(\theta) = (\mathbf{e}, \mathbf{r}_1, \mathbf{r}_2)$ where $\mathbf{r}_1, \mathbf{r}_2 \xleftarrow{\$} \mathcal{S}_{w_r}^n(\mathbb{F}_2)$, $\mathbf{e} \xleftarrow{\$} \mathcal{S}_{w_e}^n(\mathbb{F}_2)$.
  2. Computes $\mathbf{u}_{\mathbf{m}} = \mathbf{r}_1 + \mathbf{h}\mathbf{r}_2 \in \mathcal{R}$ and $\mathbf{v}_{\mathbf{m}} = \mathbf{m}\mathbf{G} + \mathbf{s}\mathbf{r}_2 + \mathbf{e} \in \mathcal{R}$.
  3. Returns the ciphertext $\mathsf{CT}_{\mathbf{m}} = (\mathbf{u}_{\mathbf{m}}, \mathbf{v}_{\mathbf{m}})$.

- $\mathsf{hqcPKE.Dec}(\mathsf{pp}_{\mathsf{hqcPKE}}, \mathsf{sk}_{\mathsf{hqcPKE}}, \mathsf{CT}_{\mathbf{m}}) \longrightarrow \mathbf{m}/\perp$: The decryptor, having the public parameter $\mathsf{pp}_{\mathsf{hqcPKE}}$ and secret key $\mathsf{sk}_{\mathsf{hqcPKE}} = (\mathbf{x}, \mathbf{y})$, runs this algorithm to decrypt the ciphertext $\mathsf{CT}_{\mathbf{m}} = (\mathbf{u}_{\mathbf{m}}, \mathbf{v}_{\mathbf{m}})$ with $\mathbf{u}_{\mathbf{m}} = \mathbf{r}_1 + \mathbf{h}\mathbf{r}_2 \in \mathcal{R}$ and $\mathbf{v}_{\mathbf{m}} = \mathbf{m}\mathbf{G} + \mathbf{s}\mathbf{r}_2 + \mathbf{e} \in \mathcal{R}$.
  1. Performs $\mathsf{C.decode}(\mathbf{v}_{\mathbf{m}} - \mathbf{u}_{\mathbf{m}}\mathbf{y}) \longrightarrow \mathbf{m}/\perp$ where the procedure $\mathsf{C.decode}()$ is the decoding algorithm of the $[n, k]$ linear code C with generator matrix $\mathbf{G} \in \mathbb{F}_2^{k \times n}$. We provided an instantiation of the encoding-decoding algorithm for Reed-Solomon code in Subsection 2.5.

Correctness. The decryption of hqcPKE depends on the decoding technique of the $[n, k]$ binary linear code C and the decryption occurs when the decoding algorithm C.decode correctly decodes $\mathbf{v_m} - \mathbf{u_m y}$ as,

$$\begin{aligned}\mathbf{v_m} - \mathbf{u_m y} = \mathbf{mG} + \mathbf{sr}_2 + \mathbf{e} - (\mathbf{r}_1 + \mathbf{hr}_2)\mathbf{y} &= \mathbf{mG} + (\mathbf{x} + \mathbf{hy})\mathbf{r}_2 + \mathbf{e} - (\mathbf{r}_1 + \mathbf{hr}_2)\mathbf{y} \\ &= \mathbf{mG} + \mathbf{xr}_2 + \mathbf{e} - \mathbf{r}_1\mathbf{y}.\end{aligned}$$

The decoding correctly executes if $\mathsf{wt}(\mathbf{xr}_2 + \mathbf{e} - \mathbf{r}_1\mathbf{y}) \leq \tau$, τ being the error-correcting capability of the code C.

Theorem 1 ([24]). *Let* $\mathbf{x} = (x_0, x_1, \ldots, x_{n-1}) \xleftarrow{\$} \mathcal{S}_w^n(\mathbb{F}_2)$, $\mathbf{r} = (r_0, r_1, \ldots, r_{n-1}) \xleftarrow{\$} \mathcal{S}_{w_r}^n(\mathbb{F}_2)$ *be mutually independent and* $\mathbf{z} = \mathbf{xr} = (z_0, z_1, \ldots, z_{n-1})$. *Then for every* $k \in \{0, 1, \ldots, n-1\}$, *the k-th coordinate* z_k *of* $\mathbf{z}$ *is Bernoulli distributed with parameter* $\tilde{p} = \Pr[z_k = 1]$ *where*

$$\tilde{p} = \frac{T}{\binom{n}{w}\binom{n}{w_r}}, \quad T = \sum_{\substack{1 \leq \ell \leq \min(w, w_r) \\ \ell \ odd}} C_\ell, \quad C_\ell = \binom{n}{\ell}\binom{n-\ell}{w-\ell}\binom{n-w}{w_r-\ell}.$$

Theorem 2 ([24]). *Let* $\mathbf{x}, \mathbf{y} \xleftarrow{\$} \mathcal{S}_w^n(\mathbb{F}_2)$, $\mathbf{r}_1, \mathbf{r}_2 \xleftarrow{\$} \mathcal{S}_{w_r}^n(\mathbb{F}_2)$, $\mathbf{e} \xleftarrow{\$} \mathcal{S}_{w_e}^n(\mathbb{F}_2)$ *be mutually independent random vectors and* $\mathbf{e}' = \mathbf{xr}_2 - \mathbf{r}_1\mathbf{y} + \mathbf{e} = (e'_0, e'_1, \ldots, e'_{n-1})$. *Then for every* $k = \{0, 1, \ldots, n-1\}$, *the k th coordinate* e'_k *of* $\mathbf{e}'$ *satisfies the following distribution.*

$$\begin{aligned}&\Pr[e'_k = 1] = 2\tilde{p}(1-\tilde{p})(1 - \frac{w_e}{n}) + ((1-\tilde{p})^2 + \tilde{p}^2)\frac{w_e}{n}, \\ &\Pr[e'_k = 0] = ((1-\tilde{p})^2 + \tilde{p}^2)(1 - \frac{w_e}{n}) + 2\tilde{p}(1-\tilde{p})\frac{w_e}{n}, \\ &\textit{with } \tilde{p} = \frac{T}{\binom{n}{w}\binom{n}{w_r}}, \ T = \sum_{\substack{1 \leq \ell \leq \min(w, w_r) \\ \ell \ odd}} C_\ell, \ C_\ell = \binom{n}{\ell}\binom{n-\ell}{w-\ell}\binom{n-w}{w_r-\ell}.\end{aligned}$$

Theorem 3 ([24]). *The encryption scheme* hqcPKE *presented above is* IND-CPA *secure as presented in Definition17, assuming the hardness of* 2-DQCSDP *and* 3-DQCSDP *problems defined in Definitions 13 and 15.*

2.5 Encoding-Decoding of Reed-Solomon Code

The scheme hqcPKE can be instantiated by BCH code, Reed-Solomon code, Reed-Muller code [24]. We discuss the encoding-decoding algorithm for Reed-Solomon code [15].

A Reed-Solomon code $\mathsf{RS}[n, k, d_{\mathsf{min}}]$ over $\mathbb{F}_{2^m}$ is a linear code with block length $n = 2^m - 1$, the number of information bits k (dimension), error-correcting capacity τ and minimum distance $d_{\mathsf{min}} = 2\tau + 1$, number of parity check digits $n - k = 2\tau$. The generator polynomial for $\mathsf{RS}[n, k, d_{\mathsf{min}}]$ code is $g(x) =$

$(x+\alpha)(x+\alpha^2)\cdots(x+\alpha^{2\tau})$ where α is a primitive element in $\mathbb{F}_{2^m}$. A message $\mathbf{u}=(u_0,u_1,\ldots,u_{k-1})\in\mathbb{F}_2^k$ which is represented as a polynomial $u(x)=u_0+u_1x+\cdots+u_{k-1}x^{k-1}\in\mathbb{F}_2[x]$. The encoding procedure consists of the three steps. The decoding procedure for $\mathsf{RS}[n,k,d_{\mathsf{min}}]$ code is presented in Fig-

1. $u(x)=u_0+u_1x+\cdots+u_{k-1}x^{k-1}\in\mathbb{F}_2[x]$ will be multiplied by x^{n-k}.
2. Divide $x^{n-k}u(x)$ by the generator polynomial $g(x)$ and obtain the remainder $b(x)$ with $deg(b(x))\leq n-k=2\tau$.
3. The code polynomial is $c(x)=b(x)+x^{n-k}u(x)\in\mathbb{F}_2^n$.

Fig. 2: Encoding procedure for the Reed-Solomon Code [15]

ure 3. Suppose the codeword $c(x)=c_0+c_1x+\cdots+c_{n-1}x^{n-1}\in\mathbb{F}_2^n[x]$ and the received codeword be $r(x)=r_0+r_1x+r_2x^2+\cdots+r_{n-1}x^{n-1}\in\mathbb{F}_2^n[x]$. Let $r(x)=c(x)+e(x)$ when $e(x)=e_0+e_1x+\cdots+e_{n-1}x^{n-1}\in\mathbb{F}_2^n[x]$ be the error polynomial.

1. As the first step, the syndrome values $S_1,S_2,\ldots,S_{2\tau}$ of the received code $r(x)$ are calculated where τ is the error-correcting capacity of the code. Note that 2τ number of syndromes of the received codeword is taken so that we can construct the error location polynomial using Berlekamp's algorithm [7].

2. Given the polynomial $r(x)$, compute $S_i=r(\alpha^i)$ for $i=\{1,2,\ldots,2\tau\}$ where $\alpha\in\mathbb{F}_{2^m}$ is a primitive element.

3. Obtain $S_i=c(\alpha^i)+e(\alpha^i)=0+e(\alpha^i)=e(\alpha^i)$as every valid codeword $c(x)$ satisfies, $c(\alpha^i)=0$ for $i=1,2,\ldots,2\tau$. Thus, the syndromes depend only on the error polynomial $e(x)$.

4. We assume the error polynomial has $\delta(\leq\tau)$ errors $e_{j_1},e_{j_2},\ldots,e_{j_\delta}$ at locations $j_1,j_2,\ldots,j_\delta$. These syndromes form a system of 2τ nonlinear equations.

$$\begin{cases} S_1=e_{j_1}\alpha^{j_1}+e_{j_2}\alpha^{j_2}+\cdots+e_{j_\delta}\alpha^{j_\delta},\\ S_2=e_{j_1}(\alpha^{j_1})^2+e_{j_2}(\alpha^{j_2})^2+\cdots+e_{j_\delta}(\alpha^{j_\delta})^2,\\ S_3=e_{j_1}(\alpha^{j_1})^3+e_{j_2}(\alpha^{j_2})^3+\cdots+e_{j_\delta}(\alpha^{j_\delta})^3,\\ \qquad\qquad\vdots\\ S_{2\tau}=e_{j_1}(\alpha^{j_1})^{2\tau}+e_{j_2}(\alpha^{j_2})^{2\tau}+\cdots+e_{j_\delta}(\alpha^{j_\delta})^{2\tau}. \end{cases} \tag{1}$$

5. Defining $\beta_i = \alpha^{j_i} \in \mathbb{F}_{2^m}$ for $i = \{1, 2, \ldots, 2\tau\}$ into the system we get

$$\begin{cases} S_1 = e_{j_1}\beta_1 + e_{j_2}\beta_2 + \cdots + e_{j_\delta}\beta_\delta, \\ S_2 = e_{j_1}\beta_1^2 + e_{j_2}\beta_2^2 + \cdots + e_{j_\delta}\beta_\delta^2, \\ S_3 = e_{j_1}\beta_1^3 + e_{j_2}\beta_2^3 + \cdots + e_{j_\delta}\beta_\delta^3, \\ \qquad\qquad \vdots \\ S_{2\tau} = e_{j_1}\beta_1^{2\tau} + e_{j_2}\beta_2^{2\tau} + \cdots + e_{j_\delta}\beta_\delta^{2\tau}. \end{cases} \tag{2}$$

1. Compute the 2τ syndromes S_i as in the equation 1 using the received polynomial $r(x) \in \mathbb{F}_2^n[x]$ by evaluating $r(\alpha^i)$ for each i for $i \in \{1, 2, \cdots, \tau\}$.
2. Compute the error-location polynomial $\zeta(x) = (1 + \beta_1 x)(1 + \beta_2 x) \cdots (1 + \beta_\delta x) = 1 + \zeta_1 x + \zeta_2 x^2 + \cdots + \zeta_\delta x^\delta$ from the 2τ syndromes from equation 2 using the Berlekamp's algorithm [7].
3. Find the error-location numbers $(\beta_1^{-1}, \beta_2^{-1}, \ldots, \beta_\delta^{-1})$ by calculating the roots of the polynomial $\zeta(x)$ and find their inverses.
4. Compute the error-evaluator polynomial $Z(x) = 1 + (S_1 + \zeta_1)x + (S_2 + \zeta_1 S_1 + \zeta_2)x^2 + \cdots + (S_\delta + \zeta_1 S_{\delta-1} + \cdots + \zeta_\delta)x^\delta$.
5. Compute the error values $e_{j_l} = \frac{Z(\beta_l^{-1})}{\prod_{\substack{i=1 \\ i \neq l}}^{\delta}(1 + \beta_i \beta_l^{-1})}$ at the location β_ℓ.
6. Correct the errors in the received polynomial $r(x)$ by computing $r(x) - e(x)$ where $e(x) = (e_{j_1}, e_{j_2}, \ldots, e_{j_\delta})$.

Fig. 3: Decoding procedure for the Reed-Solomon Code [15]

2.6 Updatable Public Key Encryption

Definition 18. *[11] An updatable public key encryption (*uPKE*) scheme associated with a message space $\mathcal{MS}$, a ciphertext space $\mathcal{CT}$, a public key space $\mathcal{PKS}$, a secret key space $\mathcal{SKS}$ and randomness space $\mathcal{RS}$, $\mathcal{RS}'$ is specified by the following six* PPT *algorithms* $\mathsf{uPKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{PkUpdate}, \mathsf{SkUpdate})$.

- $\mathsf{uPKE.Setup}(1^\lambda) \longrightarrow \mathsf{pp}_{\mathsf{uPKE}}$*: A trusted authority runs the* PPT *algorithm taking the security parameter λ as input and publishes a public parameter* $\mathsf{pp}_{\mathsf{uPKE}}$.

- $\mathsf{uPKE.KeyGen}(\mathsf{pp}_{\mathsf{uPKE}}) \longrightarrow (\mathsf{pk}_0^{(\mathsf{u})}, \mathsf{sk}_0^{(\mathsf{u})})$*: This* PPT *algorithm, run by a user* u*, takes the public parameter* $\mathsf{pp}_{\mathsf{uPKE}}$ *as input and outputs a fresh secret key* $\mathsf{sk}_0^{(\mathsf{u})}$ *and a fresh public key* $\mathsf{pk}_0^{(\mathsf{u})}$ *at epoch 0. It keeps the secret key* $\mathsf{sk}_0^{(\mathsf{u})}$ *to itself and publishes the public key* $\mathsf{pk}_0^{(\mathsf{u})}$.

– $\mathsf{uPKE.Enc}(\mathsf{pp}_{\mathsf{uPKE}}, \mathsf{pk}_i^{(\mathsf{u})}, \mathbf{m}; \theta) \longrightarrow \mathsf{CT}_{\mathbf{m}}$*: The encryptor runs this* PPT *algorithm, taking the public parameter* $\mathsf{pp}_{\mathsf{uPKE}}$*, public key* $\mathsf{pk}_i^{(\mathsf{u})}$ *at epoch* i *and a message* $\mathbf{m} \in \mathcal{MS}$ *as input and outputs a ciphertext* $\mathsf{CT}_{\mathbf{m}} \in \mathcal{CS}$ *where* $\theta \in \mathcal{RS}'$ *is an encryption randomness choosen by the algorithm during encryption.*

– $\mathsf{uPKE.Dec}(\mathsf{pp}_{\mathsf{uPKE}}, \mathsf{sk}_i^{(\mathsf{u})}, \mathsf{CT}_{\mathbf{m}}) \longrightarrow \mathbf{m}/\perp$*: The decryptor runs this* PPT *algorithm, which takes the public parameter* $\mathsf{pp}_{\mathsf{uPKE}}$*, the secret key* $\mathsf{sk}_i^{(\mathsf{u})}$ *at epoch* i *and the ciphertext* $\mathsf{CT}_{\mathbf{m}}$ *as input and returns either the message* $\mathbf{m}$ *or* $\perp$ *which indicates the decryption failure.*

– $\mathsf{uPKE.PkUpdate}(\mathsf{pp}_{\mathsf{uPKE}}, \mathsf{pk}_i^{(\mathsf{u})}; \boldsymbol{\mu}_i) \longrightarrow (\mathsf{up}_{i+1}^{(\mathsf{u})}, \mathsf{pk}_{i+1}^{(\mathsf{u})})$*: Any user who wants to update the public key can run this* PPT *algorithm which takes the public parameter* $\mathsf{pp}_{\mathsf{uPKE}}$*, the public key* $\mathsf{pk}_i^{(\mathsf{u})}$ *at epoch* i *as input and outputs an update ciphertext* $\mathsf{up}_{i+1}^{(\mathsf{u})}$ *and a new public key* $\mathsf{pk}_{i+1}^{(\mathsf{u})}$ *at epoch* $i+1$ *where* $\boldsymbol{\mu}_i \in \mathcal{RS}$ *is randomness choosen by the algorithm during public key update.*

– $\mathsf{uPKE.SkUpdate}(\mathsf{pp}_{\mathsf{uPKE}}, \mathsf{up}_{i+1}^{(\mathsf{u})}, \mathsf{sk}_i^{(\mathsf{u})}) \longrightarrow \mathsf{sk}_{i+1}^{(\mathsf{u})}$*: A user* u *runs this* PPT *algorithm on input the public parameter* $\mathsf{pp}_{\mathsf{uPKE}}$*, the update ciphertext* $\mathsf{up}_{i+1}^{(\mathsf{u})}$ *at epoch* $i+1$ *and the secret key* $\mathsf{sk}_i^{(\mathsf{u})}$ *at epoch* i *and outputs an updated secret key* $\mathsf{sk}_{i+1}^{(\mathsf{u})}$ *for epoch* $i+1$*.*

Correctness: For any sequence of randomness $\{\boldsymbol{\mu}_i\}_{i=1}^{q}$, where q is any positive integer representing an arbitrary chain of updates and $\boldsymbol{\mu}_i \in \mathcal{RS}$ the scheme uPKE is said to be correct if for any message $\mathbf{m} \in \mathcal{MS}$ and for any $i \in \{0, 1, 2, \ldots, q\}$,

$$\Pr\left[\mathsf{uPKE.Dec}(\mathsf{pp}_{\mathsf{uPKE}}, \mathsf{sk}_{i+1}^{(\mathsf{u})}, \mathsf{CT}_{\mathbf{m}}) = \mathbf{m} \;\middle|\; \begin{array}{l} \mathsf{pp}_{\mathsf{uPKE}} \leftarrow \mathsf{uPKE.Setup}(1^{\lambda}), \\ (\mathsf{pk}_0^{(\mathsf{u})}, \mathsf{sk}_0^{(\mathsf{u})}) \leftarrow \mathsf{uPKE.KeyGen}(\mathsf{pp}_{\mathsf{uPKE}}), \\ (\mathsf{up}_{i+1}^{(\mathsf{u})}, \mathsf{pk}_{i+1}^{(\mathsf{u})}) \leftarrow \mathsf{uPKE.PkUpdate}(\mathsf{pp}_{\mathsf{uPKE}}, \mathsf{pk}_i^{(\mathsf{u})}; \boldsymbol{\mu}_i), \\ \mathsf{sk}_{i+1}^{(\mathsf{u})} \leftarrow \mathsf{uPKE.SkUpdate}(\mathsf{pp}_{\mathsf{uPKE}}, \mathsf{up}_{i+1}^{(\mathsf{u})}, \mathsf{sk}_i^{(\mathsf{u})}), \\ \mathsf{CT}_{\mathbf{m}} \leftarrow \mathsf{uPKE.Enc}(\mathsf{pp}_{\mathsf{uPKE}}, \mathsf{pk}_{i+1}^{(\mathsf{u})}, \mathbf{m}; \theta) \end{array}\right] \geq 1 - \delta(\lambda)$$

where $\delta(\lambda)$ denotes the decryption failure rate.

Definition 19 (Indistinguishability under Chosen Randomness Chosen Plaintext Attack (IND-CR-CPA)**[11]).** *Consider an updatable public key encryption scheme* $\mathsf{uPKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{PkUpdate}, \mathsf{SkUpdate})$*. The scheme* uPKE *is said to be* IND-CR-CPA *secure if for any* PPT *adversary* $\mathcal{A}$ *the advantage*

$$\mathsf{Adv}_{\mathsf{uPKE},\mathcal{A}}^{\mathsf{IND\text{-}CR\text{-}CPA}}(1^{\lambda}) = \left|\Pr\left[\mathsf{EXP}_{\mathsf{uPKE},\mathcal{A}}^{\mathsf{IND\text{-}CR\text{-}CPA}}(1^{\lambda}) = 1\right] - \frac{1}{2}\right| = \left|\Pr\left[b = b'\right] - \frac{1}{2}\right|$$

is negligible where the $\mathsf{IND\text{-}CR\text{-}CPA}$ *security experiment* $\mathsf{EXP}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{uPKE},\mathcal{A}}(1^\lambda)$ *is the security game between a challenger* $\mathcal{C}$ *and the adversary* $\mathcal{A}$ *as presented in Figure 4.*

Setup: The challenger $\mathcal{C}$ first computes $\mathsf{pp}_{\mathsf{uPKE}} \longleftarrow \mathsf{uPKE.Setup}(1^\lambda)$ and $(\mathsf{pk}_0^{(\mathsf{u})}, \mathsf{sk}_0^{(\mathsf{u})}) \longleftarrow \mathsf{uPKE.KeyGen}(\mathsf{pp}_{\mathsf{uPKE}})$. It keeps the secret key $\mathsf{sk}_0^{(\mathsf{u})}$ to itself and forwards the public key $\mathsf{pk}_0^{(\mathsf{u})}$ along with the public parameter $\mathsf{pp}_{\mathsf{uPKE}}$ to the adversary $\mathcal{A}$. It also sets the current epoch $i = 0$.

Pre-Challenge Query Phase: The adversary submits polynomially many adaptive queries to the following oracle $\mathcal{O}^{\mathsf{uPKE}}_{\mathsf{upd}}(\cdot)$.

– $\mathcal{O}^{\mathsf{uPKE}}_{\mathsf{upd}}(\boldsymbol{\mu}_i)$: On receiving the choice of the randomness $\boldsymbol{\mu}_i \in \mathcal{RS}$ from the adversary, the challenger computes $(\mathsf{up}_{i+1}^{(\mathsf{u})}, \mathsf{pk}_{i+1}^{(\mathsf{u})}) \longleftarrow \mathsf{uPKE.PkUpdate}(\mathsf{pp}_{\mathsf{uPKE}}, \mathsf{pk}_i^{(\mathsf{u})}; \boldsymbol{\mu}_i)$ and $\mathsf{sk}_{i+1}^{(\mathsf{u})} \longleftarrow \mathsf{uPKE.SkUpdate}(\mathsf{pp}_{\mathsf{uPKE}}, \mathsf{up}_{i+1}^{(\mathsf{u})}, \mathsf{sk}_i^{(\mathsf{u})})$. The challenger $\mathcal{C}$ forwards $(\mathsf{up}_{i+1}^{(\mathsf{u})}, \mathsf{pk}_{i+1}^{(\mathsf{u})})$ to the adversary $\mathcal{A}$ and keeps the secret key $\mathsf{sk}_{i+1}^{(\mathsf{u})}$ secret. Also, it increases the epoch i to $i + 1$.

Challenge Phase: After receiving $\mathbf{m}_0^*, \mathbf{m}_1^* \in \mathcal{MS}$ from the adversary, the challenger $\mathcal{C}$ uniformly samples a bit $b \xleftarrow{\$} \{0, 1\}$, computes $\mathsf{CT}_{\mathbf{m}_b^*} \longleftarrow \mathsf{uPKE.Enc}(\mathsf{pp}_{\mathsf{uPKE}}, \mathsf{pk}_t^{(\mathsf{u})}, \mathbf{m}_b^*; \theta)$ where where $\theta \in \mathcal{RS}'$ is an encryption randomness choosen by the algorithm during encryption and sends $\mathsf{CT}_{\mathbf{m}_b^*}$ to the adversary $\mathcal{A}$ where t is the current epoch.

Post-Challenge Query Phase: The adversary $\mathcal{A}$ is allowed to make $\mathcal{O}^{\mathsf{uPKE}}_{\mathsf{upd}}(\cdot)$ queries as in pre-challenge query phase.

Reveal Phase: The challenger $\mathcal{C}$ chooses a new randomness $\boldsymbol{\mu}^* \xleftarrow{\$} \mathcal{RS}$ and generates $(\mathsf{up}^*, \mathsf{pk}^*) \longleftarrow \mathsf{uPKE.PkUpdate}(\mathsf{pp}_{\mathsf{uPKE}}, \mathsf{pk}_{t'}^{(\mathsf{u})}; \boldsymbol{\mu}^*)$ and $\mathsf{sk}^* \longleftarrow \mathsf{uPKE.SkUpdate}(\mathsf{pp}_{\mathsf{uPKE}}, \mathsf{up}^*, \mathsf{sk}_{t'}^{(\mathsf{u})})$. $\mathcal{C}$ then sends $(\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{up}^*)$ to the adversary $\mathcal{A}$ where $t'(\geq t)$ is the current epoch..

Guess Phase: The adversary $\mathcal{A}$ submits a bit b' to the challenger. If $b' = b$, the experiment $\mathsf{EXP}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{uPKE},\mathcal{A}}(1^\lambda)$ outputs 1, else outputs 0.

Fig. 4: $\mathsf{EXP}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{uPKE},\mathcal{A}}(1^\lambda)$: $\mathsf{IND\text{-}CR\text{-}CPA}$ security experiment between a challenger $\mathcal{C}$ and the adversary $\mathcal{A}$

3 uhqcPKE: Updatable Hamming Quasi Cyclic Public Encryption

3.1 Protocol Requirements:

Let $\mathcal{R} = \mathbb{F}_2[x]/(x^n - 1)$ and $\mathsf{Samp} : \mathcal{MS} \longrightarrow \mathsf{Per}(\mathcal{R})$ be a deterministic sampling algorithm where $\mathcal{MS}$ is the message space and the set $\mathsf{Per}(\mathcal{R})$ is defined as the collection of special structured permutation maps $\mathsf{Per}(\mathcal{R}) = \{(\Pi, \mathsf{F})\}$ with the following properties:

– **Linearity/Homomorphism:** For any $\mathbf{x}, \mathbf{y}, \mathbf{h} \in \mathcal{R}$, it holds that $\mathsf{F}(\mathbf{x}+\mathbf{hy}) = \mathsf{F}(\mathbf{x}) + \Pi(\mathbf{h})\mathsf{F}(\mathbf{y})$.

– **Weight-Preservation:** Both Π and F preserve the Hamming weight of their inputs. For any $\mathbf{x}, \mathbf{y} \in \mathcal{R}$, $\mathsf{wt}(\mathbf{x}) = \mathsf{wt}(\Pi(\mathbf{x}))$ and $\mathsf{wt}(\mathbf{y}) = \mathsf{wt}(\mathsf{F}(\mathbf{y}))$.
– **Distributional Equivalence:** The set $\mathsf{Per}(\mathcal{R})$ must have sufficient entropy such that the output of Samp mimics random sampling. Specifically, for $\boldsymbol{\mu} \xleftarrow{\$} \mathbb{F}_2^k$, let $(\Pi_{\boldsymbol{\mu}}, \mathsf{F}_{\boldsymbol{\mu}}) \leftarrow \mathsf{Samp}(\boldsymbol{\mu})$. For uniformly sampled $\mathbf{x} \xleftarrow{\$} \mathcal{S}_w^n(\mathbb{F}_2)$ and $\mathbf{h} \xleftarrow{\$} \mathcal{R}$, the distributions of $\mathsf{F}_{\boldsymbol{\mu}}(\mathbf{x})$ and $\Pi_{\boldsymbol{\mu}}(\mathbf{h})$ must be computationally indistinguishable from vectors sampled uniformly at random from $\mathcal{S}_w^n(\mathbb{F}_2)$ and $\mathcal{R}$, respectively.

3.2 Description of the protocol

Our proposed updatable hamming quasi cyclic public key encryption $\mathsf{uhqcPKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{PkUpdate}, \mathsf{SkUpdate})$ works as follows.

– $\mathsf{uhqcPKE.Setup}(1^\lambda) \longrightarrow \mathsf{pp}_{\mathsf{uhqcPKE}}$: On input the security parameter λ, a trusted authority runs this PPT algorithm and publishes a set of public parameters $\mathsf{pp}_{\mathsf{uhqcPKE}}$ as follows.
  1. Chooses non-negative integers $n, k, \alpha, \tau, w, w_r, w_e$ where n is the length and k is the dimension of a τ error correcting decodable linear code C over $\mathbb{F}_2$ and w, w_r, w_e denotes fixed weight of vectors and α denotes the length of encryption randomness.
  2. Selects a generator matrix $\mathbf{G}$ of order $k \times n$ for the $[n, k]$ binary linear code C.
  3. Selects a $\mathsf{Samp} : \mathbb{F}_2^k \longrightarrow \mathsf{Per}(\mathcal{R})$ where $\mathsf{Per}(\mathcal{R})$ is described as in Subsection 3.1 and Section 5 where $\mathcal{R} = \mathbb{F}_2[x]/x^n - 1$.
  4. Selects a randomness generator $\mathsf{P} : \{0,1\}^* \longrightarrow \mathcal{S}_{w_e}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2)$, we use the notation $\mathcal{S}_w^n(\mathbb{F}_2) = \{\mathbf{v} \in \mathbb{F}_2^n \mid \mathsf{wt}(\mathbf{v}) = w\}$.
  5. Publishes public parameter $\mathsf{pp}_{\mathsf{uhqcPKE}} = (n, k, \alpha, \tau, w, w_r, w_e, \mathsf{Samp}, \mathbf{G}, \mathsf{P})$.

– $\mathsf{uhqcPKE.KeyGen}(\mathsf{pp}_{\mathsf{uhqcPKE}}) \longrightarrow (\mathsf{pk}_0^{(\mathsf{u})}, \mathsf{sk}_0^{(\mathsf{u})})$: A user u runs this PPT algorithm taking $\mathsf{pp}_{\mathsf{uhqcPKE}} = (n, k, \alpha, \tau, w, w_r, w_e, \mathsf{Samp}, \mathbf{G}, \mathsf{P})$, the public parameter, as input and returns a fresh pair of public key and secret key $(\mathsf{pk}_0^{(\mathsf{u})}, \mathsf{sk}_0^{(\mathsf{u})})$ at epoch 0 by computing the following steps.
  1. Samples $\mathbf{h}_0 \xleftarrow{\$} \mathcal{R}$ and $(\mathbf{x}_0, \mathbf{y}_0) \xleftarrow{\$} \mathcal{S}_w^n(\mathbb{F}_2) \times \mathcal{S}_w^n(\mathbb{F}_2)$. Note that $\mathbf{x}_0, \mathbf{y}_0 \in \mathbb{F}_2^n$ with $\mathsf{wt}(\mathbf{x}_0) = w = \mathsf{wt}(\mathbf{y}_0)$.
  2. Computes $\mathbf{s}_0 = \mathbf{x}_0 + \mathbf{h}_0\mathbf{y}_0 \in \mathcal{R}$.
  3. Sets the public key $\mathsf{pk}_0^{(\mathsf{u})} = (\mathbf{h}_0, \mathbf{s}_0)$ and secret key $\mathsf{sk}_0^{(\mathsf{u})} = (\mathbf{x}_0, \mathbf{y}_0)$.
  4. Publishes the public key $\mathsf{pk}_0^{(\mathsf{u})}$ while keeping the secret key $\mathsf{sk}_0^{(\mathsf{u})}$ secret to itself.

– $\mathsf{uhqcPKE.Enc}(\mathsf{pp}_{\mathsf{uhqcPKE}}, \mathsf{pk}_i^{(\mathsf{u})}, \mathbf{m}; \theta) \longrightarrow \mathsf{CT}_{\mathbf{m}}$: The encryptor runs the algorithm, taking public parameter $\mathsf{pp}_{\mathsf{uhqcPKE}} = (n, k, \alpha, \tau, w, w_r, w_e, \mathsf{Samp}, \mathbf{G}, \mathsf{P})$, public key $\mathsf{pk}_i^{(\mathsf{u})} = (\mathbf{h}_i, \mathbf{s}_i)$ at epoch i, a message $\mathbf{m} \in \mathbb{F}_2^k$ as input and outputs $\mathsf{CT}_{\mathbf{m}} = (\mathbf{u}_{\mathbf{m}}, \mathbf{v}_{\mathbf{m}}) \in \mathcal{R} \times \mathcal{R}$ by executing the following steps.

1. Choose a α-bit string $\theta \in \mathbb{F}_2^\alpha$ and compute $\mathsf{P}(\theta) = (\mathbf{e}, \mathbf{r}_1, \mathbf{r}_2)$ where $\mathbf{e} \xleftarrow{\$} \mathcal{S}_{w_e}^n(\mathbb{F}_2)$ and $\mathbf{r} = (\mathbf{r}_1, \mathbf{r}_2) \xleftarrow{\$} \mathcal{S}_{w_r}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2)$. Note that, $\mathsf{wt}(\mathbf{e}) = w_e$ and $\mathsf{wt}(\mathbf{r}_1) = \mathsf{wt}(\mathbf{r}_2) = w_r$.
2. Computes $\mathbf{u_m} = \mathbf{r}_1 + \mathbf{h}_i\mathbf{r}_2 \in \mathcal{R}$ and $\mathbf{v_m} = \mathbf{mG} + \mathbf{s}_i\mathbf{r}_2 + \mathbf{e} \in \mathcal{R}$.
3. Returns $\mathsf{CT}_\mathbf{m} = (\mathbf{u_m}, \mathbf{v_m})$.

– $\mathsf{uhqcPKE.Dec}(\mathsf{pp}_{\mathsf{uhqcPKE}}, \mathsf{sk}_i^{(\mathsf{u})}, \mathsf{CT}_\mathbf{m}) \longrightarrow \mathbf{m}/\perp$: The decryptor runs this algorithm with the public parameter $\mathsf{pp}_{\mathsf{uhqcPKE}} = (n, k, \tau, w, w_r, w_e, \mathsf{Samp}, \mathbf{G}, \mathsf{P})$, its secret key $\mathsf{sk}_i^{(\mathsf{u})} = (\mathbf{x}_i, \mathbf{y}_i)$ at epoch i and the ciphertext $\mathsf{CT}_\mathbf{m} = (\mathbf{u_m}, \mathbf{v_m})$ as input and outputs $\mathbf{m}$ or $\perp$ by running the decoding algorithm $\mathsf{C.decode}(\mathbf{v_m} - \mathbf{u_m}\mathbf{y}_i)$ as in the decryption algorithm (for RS code) of hqcPKE presented in Subsection 2.4.

– $\mathsf{uhqcPKE.PkUpdate}(\mathsf{pp}_{\mathsf{uhqcPKE}}, \mathsf{pk}_i^{(\mathsf{u})}; \boldsymbol{\mu}_i) \longrightarrow (\mathsf{up}_{i+1}^{(\mathsf{u})}, \mathsf{pk}_{i+1}^{(\mathsf{u})})$: Taking as input the public parameter $\mathsf{pp}_{\mathsf{uhqcPKE}} = (n, k, \alpha, \tau, w, w_r, w_e, \mathsf{Samp}, \mathbf{G}, \mathsf{P})$, the public key $\mathsf{pk}_i^{(\mathsf{u})} = (\mathbf{h}_i, \mathbf{s}_i = \mathbf{x}_i + \mathbf{h}_i\mathbf{y}_i)$ at epoch i , a user runs this algorithm and updates the public key by computing the following steps to output a new pair of an update ciphertext and public key $(\mathsf{up}_{i+1}^{(\mathsf{u})}, \mathsf{pk}_{i+1}^{(\mathsf{u})})$.

1. Samples a randomness $\boldsymbol{\mu}_i \xleftarrow{\$} \mathbb{F}_2^k$ and generates $(\Pi_{\boldsymbol{\mu}_i}, \mathsf{F}_{\boldsymbol{\mu}_i}) = \mathsf{Samp}(\boldsymbol{\mu}_i) \in \mathsf{Per}(\mathcal{R})$.
2. Computes $\Pi_{\boldsymbol{\mu}_i}(\mathbf{h}_i) = \mathbf{h}_{i+1} \in \mathcal{R}$ and $\mathbf{s}_{i+1} = \mathsf{F}_{\boldsymbol{\mu}_i}(\mathbf{s}_i)$. Note that $\mathbf{s}_{i+1} = \mathsf{F}_{\boldsymbol{\mu}_i}(\mathbf{s}_i) = \mathsf{F}_{\boldsymbol{\mu}_i}(\mathbf{x}_i + \mathbf{h}_i\mathbf{y}_i) = \mathsf{F}_{\boldsymbol{\mu}_i}(\mathbf{x_i}) + \Pi_{\boldsymbol{\mu}_i}(\mathbf{h}_i)\mathsf{F}_{\boldsymbol{\mu}_i}(\mathbf{y}_i) = \mathbf{x}_{i+1} + \mathbf{h}_{i+1}\mathbf{y}_{i+1} \in \mathcal{R}$ for epoch $i+1$.
3. Encrypts $\boldsymbol{\mu}_i$ by running the $\mathsf{uhqcPKE.Enc}(\mathsf{pp}_{\mathsf{uhqcPKE}}, \mathsf{pk}_i^{(\mathsf{u})}, \boldsymbol{\mu}_i; \theta')$ which outputs $\mathsf{up}_{i+1}^{(\mathsf{u})} = \mathsf{CT}_{\boldsymbol{\mu}_i} = (\mathbf{u}_{\boldsymbol{\mu}_i}, \mathbf{v}_{\boldsymbol{\mu}_i}) = (\hat{\mathbf{r}}_1 + \mathbf{h}_i\hat{\mathbf{r}}_2, \boldsymbol{\mu}_i\mathbf{G} + \mathbf{s}_i\hat{\mathbf{r}}_2 + \hat{\mathbf{e}})$ where $\mathsf{P}(\theta') = (\hat{\mathbf{e}}, \hat{\mathbf{r}}_1, \hat{\mathbf{r}}_2)$ for $\theta' \in \mathbb{F}_2^\alpha$.
4. Updates the public key as $\mathsf{pk}_{i+1}^{(\mathsf{u})} = (\mathbf{h}_{i+1}, \mathbf{s}_{i+1})$.
5. Returns update ciphertext-public key pair $(\mathsf{up}_{i+1}^{(\mathsf{u})}, \mathsf{pk}_{i+1}^{(\mathsf{u})})$.

– $\mathsf{uhqcPKE.SkUpdate}(\mathsf{pp}_{\mathsf{uhqcPKE}}, \mathsf{up}_{i+1}^{(\mathsf{u})}, \mathsf{sk}_i^{(\mathsf{u})}) \longrightarrow \mathsf{sk}_{i+1}^{(\mathsf{u})}$: The user u runs this algorithm on input the public parameter $\mathsf{pp}_{\mathsf{uhqcPKE}} = (n, k, \alpha, , \tau, w, w_r, w_e, \mathsf{Samp}, \mathbf{G}, \mathsf{P})$, its update ciphertext $\mathsf{up}_{i+1}^{(\mathsf{u})}$ at epoch $i+1$, secret key $\mathsf{sk}_i^{(\mathsf{u})}$ at epoch $i+1$ and generates its updated secret key $\mathsf{sk}_{i+1}^{(\mathsf{u})}$ for epoch $i+1$ by executing the following steps.

1. Executes the algorithm $\mathsf{uhqcPKE.Dec}(\mathsf{pp}_{\mathsf{uhqcPKE}}, \mathsf{sk}_i^{(\mathsf{u})}, \mathsf{up}_{i+1}^{(\mathsf{u})})$ returns $\boldsymbol{\mu}_i$ by running the algorithm $\mathsf{C.Decode}(\mathbf{v}_{\boldsymbol{\mu}_i} - \mathbf{u}_{\boldsymbol{\mu}_i}\mathbf{y}_i)$.
2. Computes $(\Pi_{\boldsymbol{\mu}_i}, \mathsf{F}_{\boldsymbol{\mu}_i}) = \mathsf{Samp}(\boldsymbol{\mu}_i) \in \mathsf{Per}(\mathcal{R})$, $\mathbf{x}_{i+1} = \mathsf{F}_{\boldsymbol{\mu}_i}(\mathbf{x}_i) \in \mathcal{R}$ and $\mathbf{y}_{i+1} = \mathsf{F}_{\boldsymbol{\mu}_i}(\mathbf{y}_i) \in \mathcal{R}$. Note that $\mathsf{wt}(\mathbf{x}_{i+1}) = \mathsf{wt}(\mathbf{x}_i) = w$ and $\mathsf{wt}(\mathbf{y}_{i+1}) = \mathsf{wt}(\mathbf{y}_i) = w$.
3. Update the secret key as $\mathsf{sk}_{i+1}^{(\mathsf{u})} = (\mathbf{x}_{i+1}, \mathbf{y}_{i+1})$.

3.3 Correctness

For a given security parameter λ, let $\mathsf{pp}_{\mathsf{uhqcPKE}} \leftarrow \mathsf{uhqcPKE.Setup}(1^\lambda)$. The scheme is correct if, for any non-negative integer q representing an arbitrary chain of updates, the decryption failure rate is bounded by $\delta(1^\lambda)$.

We demonstrate correctness by tracking the evolution of keys from epoch 0 to an arbitrary epoch $i+1$ where $i \in \{0, 1, 2 \ldots, q\}$.

Epoch 0:

$$(\mathsf{sk}_0^{(\mathsf{u})}, \mathsf{pk}_0^{(\mathsf{u})}) \longleftarrow \mathsf{uhqcPKE.KeyGen}(\mathsf{pp}_{\mathsf{uhqcPKE}})$$

where $\mathsf{sk}_0^{(\mathsf{u})} = (\mathbf{x}_0, \mathbf{y}_0) \in \mathcal{S}_w^n(\mathbb{F}_2) \times \mathcal{S}_w^n(\mathbb{F}_2)$ and $\mathsf{pk}_0^{(\mathsf{u})} = (\mathbf{h}_0, \mathbf{s}_0 = \mathbf{x}_0 + \mathbf{h}_0\mathbf{y}_0) \in \mathcal{R} \times \mathcal{R}$.

Epochs 0 *to* i*:* Assume a sequence of $i+1$ updates generated by sampling update messages $\boldsymbol{\mu}_j \xleftarrow{\$} \mathbb{F}_2^k$. For $j \in \{0, 1, \ldots, i\}$, the keys recursively evolve as follows:

$$(\mathsf{up}_{j+1}^{(\mathsf{u})}, \mathsf{pk}_{j+1}^{(\mathsf{u})}) \longleftarrow \mathsf{uhqcPKE.PkUpdate}(\mathsf{pp}_{\mathsf{uhqcPKE}}, \mathsf{pk}_j^{(\mathsf{u})}; \boldsymbol{\mu}_j)$$

$$\mathsf{sk}_{j+1}^{(\mathsf{u})} \longleftarrow \mathsf{uhqcPKE.SkUpdate}(\mathsf{pp}_{\mathsf{uhqcPKE}}, \mathsf{up}_{j+1}^{(\mathsf{u})}, \mathsf{sk}_j^{(\mathsf{u})})$$

where $\mathsf{Samp}(\boldsymbol{\mu}_j) = (\Pi_{\boldsymbol{\mu}_j}, \mathsf{F}_{\boldsymbol{\mu}_j}) \in \mathsf{Per}(\mathcal{R})$. By the properties defined in Section 3.1, the keys at epoch $i+1$ are:

$$\mathsf{sk}_{i+1}^{(\mathsf{u})} = (\mathbf{x}_{i+1}, \mathbf{y}_{i+1}) = (\mathsf{F}_{\boldsymbol{\mu}_i}(\mathbf{x}_i), \mathsf{F}_{\boldsymbol{\mu}_i}(\mathbf{y}_i)) \in \mathcal{S}_w^n(\mathbb{F}_2) \times \mathcal{S}_w^n(\mathbb{F}_2)$$

$$\mathsf{pk}_{i+1}^{(\mathsf{u})} = (\mathbf{h}_{i+1}, \mathbf{s}_{i+1}) = (\Pi_{\boldsymbol{\mu}_i}(\mathbf{h}_i), \mathsf{F}_{\boldsymbol{\mu}_i}(\mathbf{x}_i + \mathbf{h}_i\mathbf{y}_i) = \mathbf{x}_{i+1} + \mathbf{h}_{i+1}\mathbf{y}_{i+1}) \in \mathcal{R} \times \mathcal{R}.$$

Because Samp ensures distributional equivalence, the key pair $(\mathsf{sk}_{i+1}^{(\mathsf{u})}, \mathsf{pk}_{i+1}^{(\mathsf{u})})$ is computationally indistinguishable from a freshly generated key pair from $\mathsf{uhqcPKE.KeyGen}$.

Encryption and Decryption at Epoch $i+1$*:* For any message $\mathbf{m} \in \mathbb{F}_2^k$, let $\mathsf{CT}_{\mathbf{m}}$ be encrypted under the updated public key $\mathsf{pk}_{i+1}^{(\mathsf{u})}$:

$$\mathsf{CT}_{\mathbf{m}} \longleftarrow \mathsf{uhqcPKE.Enc}(\mathsf{pp}_{\mathsf{uhqcPKE}}, \mathsf{pk}_{i+1}^{(\mathsf{u})}, \mathbf{m}; \theta)$$

where $\mathsf{CT}_{\mathbf{m}} = (\mathbf{u}_{\mathbf{m}}, \mathbf{v}_{\mathbf{m}}) = (\mathbf{r}_1 + \mathbf{h}_{i+1}\mathbf{r}_2, \mathbf{mG} + \mathbf{s}_{i+1}\mathbf{r}_2 + \mathbf{e}) \in \mathcal{R} \times \mathcal{R}$ for randomness $(\mathbf{e}, \mathbf{r}_1, \mathbf{r}_2) \leftarrow \mathsf{P}(\theta)$ with appropriate weights w_e and w_r.

During decryption using $\mathsf{sk}_{i+1}^{(\mathsf{u})} = (\mathbf{x}_{i+1}, \mathbf{y}_{i+1})$, the algebraic structure yields:

$$\begin{aligned}
\mathbf{v}_{\mathbf{m}} - \mathbf{u}_{\mathbf{m}}\mathbf{y}_{i+1} &= (\mathbf{mG} + \mathbf{s}_{i+1}\mathbf{r}_2 + \mathbf{e}) - (\mathbf{r}_1 + \mathbf{h}_{i+1}\mathbf{r}_2)\mathbf{y}_{i+1} \\
&= \mathbf{mG} + (\mathbf{x}_{i+1} + \mathbf{h}_{i+1}\mathbf{y}_{i+1})\mathbf{r}_2 + \mathbf{e} - (\mathbf{r}_1 + \mathbf{h}_{i+1}\mathbf{r}_2)\mathbf{y}_{i+1} \\
&= \mathbf{mG} + \mathbf{x}_{i+1}\mathbf{r}_2 + \mathbf{e} - \mathbf{r}_1\mathbf{y}_{i+1}
\end{aligned}$$

The decoding algorithm $\mathsf{C.decode}(\cdot)$ successfully recovers $\mathbf{m}$ provided that the noise term satisfies $\mathsf{wt}(\mathbf{x}_{i+1}\mathbf{r}_2 + \mathbf{e} - \mathbf{r}_1\mathbf{y}_{i+1}) \leq \tau$ where τ is the error-correcting

capability of the code C. Since the distributions of the updated keys match those of the standard hqcPKE, the probability of a decoding failure is identically bounded by the standard hqcPKE decryption failure rate δ.

Therefore,

$$\Pr\left[\mathsf{uhqcPKE.Dec}(\mathsf{pp}_{\mathsf{uhqcPKE}}, \mathsf{sk}_{i+1}^{(\mathsf{u})}, \mathsf{CT}_{\mathbf{m}}) = \mathbf{m}\right] \geq 1 - \delta.$$

4 Security Analysis

Theorem 4. Our proposed updatable public key encryption $\mathsf{uhqcPKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{PkUpdate}, \mathsf{SkUpdate})$ presented above is $\mathsf{IND\text{-}CR\text{-}CPA}$ secure as per Definition 19 under the hardness of the 2-DQCSDP and the 3-DQCSDP assumptions presented in Subsection 2.2.

Proof. To prove the $\mathsf{IND\text{-}CR\text{-}CPA}$ security of $\mathsf{uhqcPKE}$, we consider a sequence of games in which the adversary's view is gradually modified and we show that no efficient adversary can distinguish between any two consecutive games. Across these games, the challenge ciphertext is slowly transformed from the original game to the one where it is independent of the bit b in the $\mathsf{IND\text{-}CR\text{-}CPA}$ security experiment. We argue that any adversary capable of distinguishing one game from the subsequent can be used to construct a distinguisher that breaks either the 2-DQCSDP assumption or the 3-DQCSDP assumption. For each game $\mathcal{G}_i$, $i = 0, 1, \ldots, 5$, let S_i denotes the event that the adversary correctly guesses the challenge bit, i.e., $b' = b$.

- *Game* $\mathcal{G}_0$: This is the original $\mathsf{IND\text{-}CR\text{-}CPA}$ security game $\mathsf{EXP}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{uhqcPKE},\mathcal{A}}(1^\lambda)$ between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ as shown below.

  Setup: The challenger $\mathcal{C}$ computes public parameter

  $$\mathsf{pp}_{\mathsf{uhqcPKE}} \longleftarrow \mathsf{uhqcPKE.Setup}(1^\lambda)$$

  and initial public key-secret key pair

  $$(\mathsf{pk}_0^{(\mathsf{u})}, \mathsf{sk}_0^{(\mathsf{u})}) \longleftarrow \mathsf{uhqcPKE.KeyGen}(\mathsf{pp}_{\mathsf{uhqcPKE}})$$

  where $\mathsf{pp}_{\mathsf{uhqcPKE}} = (n, k, \alpha, \tau, w, w_r, w_e, \mathsf{Samp}, \mathbf{G}, \mathsf{P})$,

  $$\mathsf{pk}_0^{(\mathsf{u})} = (\mathbf{h}_0, \mathbf{s}_0) \in \mathcal{R} \times \mathcal{R},\ \mathsf{sk}_0^{(\mathsf{u})} = (\mathbf{x}_0, \mathbf{y}_0) \in \mathcal{S}_w^n(\mathbb{F}_2) \times \mathcal{S}_w^n(\mathbb{F}_2)$$

  satisfying $\mathbf{s}_0 = \mathbf{x}_0 + \mathbf{h}_0\mathbf{y}_0$. Let $b_1 = \mathbf{h}_0 \cdot \mathbf{1} \pmod 2$ and $b_2 = w + b_1 \times w \bmod 2$. Then $\mathsf{pk}_0^{(\mathsf{u})} = (\mathbf{h}_0, \mathbf{s}_0) \in \mathbb{F}^n_{2,b_1} \times \mathbb{F}^n_{2,b_2}$ as $\mathbf{h}_0$ has parity b_0 and $\mathbf{s}_0$ has parity b_1. It initializes the epoch $i = 0$ and keeps the secret key $\mathsf{sk}_0^{(\mathsf{u})}$ to itself and forwards the public key $\mathsf{pk}_0^{(\mathsf{u})}$ along with the public parameter $\mathsf{pp}_{\mathsf{uhqcPKE}}$ to the adversary $\mathcal{A}$.

Pre-Challenge Query Phase: The adversary submits $t-1$ many adaptive queries to the oracle $\mathcal{O}_{\mathsf{upd}}^{\mathsf{uhqcPKE}}(\cdot)$ which are simulated as follows by $\mathcal{C}$.

– $\mathcal{O}_{\mathsf{upd}}^{\mathsf{uhqcPKE}}(\boldsymbol{\mu}_i)$: Upon receiving the choice of randomness $\boldsymbol{\mu}_i \in \mathbb{F}_2^k$ for $i = \{0, 1, \ldots, t-1\}$ from the adversary $\mathcal{A}$, the challenger $\mathcal{C}$ computes

$$(\mathsf{up}_{i+1}^{(\mathsf{u})}, \mathsf{pk}_{i+1}^{(\mathsf{u})}) \longleftarrow \mathsf{uhqcPKE.PkUpdate}(\mathsf{pp}_{\mathsf{uhqcPKE}}, \mathsf{pk}_i^{(\mathsf{u})}; \boldsymbol{\mu}_i),$$

$$\mathsf{sk}_{i+1}^{(\mathsf{u})} \longleftarrow \mathsf{uhqcPKE.SkUpdate}(\mathsf{pp}_{\mathsf{uhqcPKE}}, \mathsf{up}_{i+1}^{(\mathsf{u})}, \mathsf{sk}_i^{(\mathsf{u})})$$

where $\mathsf{pk}_{i+1}^{(\mathsf{u})} = (\mathbf{h}_{i+1} = \Pi_{\boldsymbol{\mu}_i}(\mathbf{h}_i), \mathbf{s}_{i+1} = \mathsf{F}_{\boldsymbol{\mu}_i}(\mathbf{s}_i)) \in \mathcal{R} \times \mathcal{R}$,

$$\mathsf{up}_{i+1}^{(\mathsf{u})} = \mathsf{CT}_{\boldsymbol{\mu}_i} \longleftarrow \mathsf{uhqcPKE.Enc}(\mathsf{pp}_{\mathsf{uhqcPKE}}, \mathsf{pk}_i^{(\mathsf{u})}, \boldsymbol{\mu}_i; \theta'),$$

where $\mathsf{CT}_{\boldsymbol{\mu}_i} = (\mathbf{u}_{\boldsymbol{\mu}_i} = \mathbf{r}_1' + \mathbf{h}_i\mathbf{r}_2', \mathbf{v}_{\boldsymbol{\mu}_i} = \boldsymbol{\mu}_i\mathbf{G} + \mathbf{s}_i\mathbf{r}_2' + \mathbf{e}')$

$$\mathsf{sk}_{i+1}^{(\mathsf{u})} = (\mathbf{x}_{i+1}, \mathbf{y}_{i+1}) = (\mathsf{F}_{\boldsymbol{\mu}_i}(\mathbf{x}_i), \mathsf{F}_{\boldsymbol{\mu}_i}(\mathbf{y}_i)) \in \mathcal{S}_w^n(\mathbb{F}_2) \times \mathcal{S}_w^n(\mathbb{F}_2)$$

satisfying $\mathbf{s}_{i+1} = \mathbf{x}_{i+1} + \mathbf{h}_{i+1}\mathbf{y}_{i+1}$, $\mathsf{pk}_i^{(\mathsf{u})} = (\mathbf{h}_i, \mathbf{s}_i) \in \mathcal{R} \times \mathcal{R}$, $\mathsf{sk}_i^{(\mathsf{u})} = (\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{S}_w^n(\mathbb{F}_2) \times \mathcal{S}_w^n(\mathbb{F}_2)$ with $\mathbf{s}_i = \mathbf{x}_i + \mathbf{h}_i\mathbf{y}_i$, $(\Pi_{\boldsymbol{\mu}_i}, \mathsf{F}_{\boldsymbol{\mu}_i}) = \mathsf{Samp}(\mu_i) \in \mathsf{Per}(\mathcal{R})$, $\mathsf{P}(\theta') = (\mathbf{e}', \mathbf{r}_1', \mathbf{r}_2') \in \mathcal{S}_{w_e}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2)$, $\theta' \xleftarrow{\$} \mathbb{F}_2^\alpha$ is a random α-bit string. It increases the epoch i to $i+1$, keeps $\mathsf{sk}_{i+1}^{(\mathsf{u})}$ secret to itself and forwards $(\mathsf{up}_{i+1}^{(\mathsf{u})}, \mathsf{pk}_{i+1}^{(\mathsf{u})})$ to $\mathcal{A}$. Note that $(\mathbf{h}_0, \mathbf{s}_0) \in \mathbb{F}_{2,b_1}^n \times \mathbb{F}_{2,b_2}^n$ implies $\mathbf{h}_i, \mathbf{h}_{i+1} \in \mathbb{F}_{2,b_1}^n$ as they will have the same parity by the weight preserving property of $\Pi_{\boldsymbol{\mu}_i}$ and $\mathbf{s}_i, \mathbf{s}_{i+1} \in \mathbb{F}_{2,b_2}^n$ as they have the same parity b_2 by the weight preserving property of $\mathsf{F}_{\boldsymbol{\mu}_i}$.

Challenge Phase: On receiving the message pair $(\mathbf{m}_0^*, \mathbf{m}_1^*) \in \mathbb{F}_2^k \times \mathbb{F}_2^k$ from $\mathcal{A}$ at epoch t, the challenger $\mathcal{C}$ samples a bit $b \xleftarrow{\$} \{0,1\}$, computes

$$\mathsf{uhqcPKE.Enc}(\mathsf{pp}_{\mathsf{uhqcPKE}}, \mathsf{pk}_t^{(\mathsf{u})}, \mathbf{m}_b^*; \theta) \longrightarrow \mathsf{CT}_{\mathbf{m}_b^*}$$

where $\mathsf{CT}_{\mathbf{m}_b^*} = (\mathbf{u}_{\mathbf{m}_b^*} = \mathbf{r}_1 + \mathbf{h}_t\mathbf{r}_2, \mathbf{v}_{\mathbf{m}_b^*} = \mathbf{m}_b^*\mathbf{G} + \mathbf{s}_t\mathbf{r}_2 + \mathbf{e}) \in \mathcal{R} \times \mathcal{R}$ and sends $\mathsf{CT}_{\mathbf{m}_b^*}$ to $\mathcal{A}$ where $\mathsf{pk}_t = (\mathbf{h}_t, \mathbf{s}_t)$ is the public key at epoch t satisfying $\mathbf{s}_t = \mathbf{x}_t + \mathbf{h}_t\mathbf{y}_t$, $\theta \in \mathbb{F}_2^\alpha$ is a random α-bit string, $\mathsf{P}(\theta) = (\mathbf{e}, \mathbf{r}_1, \mathbf{r}_2) \in \mathcal{S}_{w_e}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2)$.

Post-Challenge Query Phase: The adversary $\mathcal{A}$ makes $\mathcal{O}_{\mathsf{upd}}^{\mathsf{uhqcPKE}}(\boldsymbol{\mu}_i)$ queries for $i = \{t+1, t+2, \ldots, t'-1\}$ as in the pre-challenge query phase for $\boldsymbol{\mu}_i \in \mathbb{F}_2^k$.

Reveal Phase: The challenger $\mathcal{C}$ chooses a new randomness $\boldsymbol{\mu}^* \xleftarrow{\$} \mathbb{F}_2^k$ at epoch $t'+1$ then generates

$$(\mathsf{up}^*, \mathsf{pk}^*) \longleftarrow \mathsf{uhqcPKE.PkUpdate}(\mathsf{pp}_{\mathsf{uhqcPKE}}, \mathsf{pk}_{t'}^{(\mathsf{u})}; \boldsymbol{\mu}^*)$$

and $\mathsf{sk}^* \longleftarrow \mathsf{uhqcPKE.SkUpdate}(\mathsf{pp}_{\mathsf{uhqcPKE}}, \mathsf{up}^*, \mathsf{sk}_{t'}^{(\mathsf{u})})$ and sends $(\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{up}^*)$ to $\mathcal{A}$ where

$$\mathsf{up}^* = \mathsf{CT}_{\boldsymbol{\mu}^*} \longleftarrow \mathsf{uhqcPKE.Enc}(\mathsf{pp}_{\mathsf{uhqcPKE}}, \mathsf{pk}_{t'}^{(\mathsf{u})}, \boldsymbol{\mu}^*; \theta''),$$

where $\mathsf{CT}_{\boldsymbol{\mu}^*} = (\mathbf{u}_{\boldsymbol{\mu}^*} = \mathbf{r}_1'' + \mathbf{h}_{t'}\mathbf{r}_2'',\ \mathbf{v}_{\boldsymbol{\mu}^*} = \boldsymbol{\mu}^*\mathbf{G} + \mathbf{s}_{t'}\mathbf{r}_2'' + \mathbf{e}'')$,

$$\mathsf{pk}^* = (\mathbf{h}^* = \Pi_{\boldsymbol{\mu}^*}(\mathbf{h}_{t'}),\ \mathbf{s}^* = \mathsf{F}_{\boldsymbol{\mu}^*}(\mathbf{s}_{t'})) \in \mathcal{R} \times \mathcal{R},$$

$$\mathsf{sk}^* = (\mathsf{F}_{\mu^*}(\mathbf{x}_{t'}),\ \mathsf{F}_{\mu^*}(\mathbf{y}_{t'})) = (\mathbf{x}^*,\ \mathbf{y}^*) \in \mathcal{S}_w^n(\mathbb{F}_2) \times \mathcal{S}_w^n(\mathbb{F}_2)$$

satisfying $\mathbf{s}^* = \mathbf{x}^* + \mathbf{h}^*\mathbf{y}^*$, α-bit string $\theta'' \in \mathbb{F}_2^\alpha$, $\mathsf{P}(\theta'') = (\mathbf{e}'', \mathbf{r}_1'', \mathbf{r}_2'') \in \mathcal{S}_{w_e}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2)$, $(\Pi_{\boldsymbol{\mu}^*}, \mathsf{F}_{\boldsymbol{\mu}^*}) = \mathsf{Samp}(\boldsymbol{\mu}^*) \in \mathsf{Per}(\mathcal{R})$, $\mathsf{pk}_{t'}^{(\mathsf{u})} = (\mathbf{h}_{t'}, \mathbf{s}_{t'}) \in \mathcal{R} \times \mathcal{R}$, $\mathsf{sk}_{t'}^{(\mathsf{u})} = (\mathbf{x}_{t'}, \mathbf{y}_{t'}) \in \mathcal{S}_w^n(\mathbb{F}_2) \times \mathcal{S}_w^n(\mathbb{F}_2)$ with $\mathbf{s}_{t'} = \mathbf{x}_{t'} + \mathbf{h}_{t'}\mathbf{y}_{t'}$.
As $(\mathbf{h}_0, \mathbf{s}_0) \in \mathbb{F}_{2,b_1}^n \times \mathbb{F}_{2,b_2}^n$ it follows that $\mathbf{h}_{t'}, \mathbf{h}^* \in \mathbb{F}_{2,b_1}^n$ (see Proposition 3) and $\mathbf{s}_{t'}, \mathbf{s}^* \in \mathbb{F}_{2,b_2}^n$ (see Proposition 1).

Guess Phase: The adversary outputs a bit $b' \in \{0,1\}$ as a guess for $b \in \{0,1\}$ chosen by $\mathcal{C}$ during the challenge phase.

- *Game* $\mathcal{G}_1$: This game is similar to $\mathcal{G}_0$ except that the secret key-public key pair $(\mathsf{sk}^*, \mathsf{pk}^*)$ is set as follows by $\mathcal{C}$ in the reveal phase at epoch $t'+1$ by choosing a randomness $\boldsymbol{\mu}^* \xleftarrow{\$} \mathbb{F}_2^k$ and generating $(\Pi_{\boldsymbol{\mu}^*}, \mathsf{F}_{\boldsymbol{\mu}^*}) = \mathsf{Samp}(\boldsymbol{\mu}^*) \in \mathsf{Per}(\mathcal{R})$ where $\mathsf{pk}_{t'}^{(\mathsf{u})} = (\mathbf{h}_{t'}, \mathbf{s}_{t'}) \in \mathcal{R} \times \mathcal{R}$.

$$\boxed{\mathsf{sk}^* = (\boldsymbol{\rho}^*, \boldsymbol{\gamma}^*) \xleftarrow{\$} \mathcal{S}_w^n(\mathbb{F}_2) \times \mathcal{S}_w^n(\mathbb{F}_2)}$$

$$\boxed{\mathsf{pk}^* = (\mathbf{h}^* = \Pi_{\boldsymbol{\mu}^*}(\mathbf{h}_{t'}), \mathbf{s}^* = \boldsymbol{\rho}^* + \mathbf{h}^*\boldsymbol{\gamma}^*) \in \mathcal{R} \times \mathcal{R}}$$

Note that in game $\mathcal{G}_0$,

$$\mathsf{sk}^* = (\mathbf{x}^* = \mathsf{F}_{\boldsymbol{\mu}^*}(\mathbf{x}_{t'}), \mathbf{y}^* = \mathsf{F}_{\boldsymbol{\mu}^*}(\mathbf{y}_{t'})) \in \mathcal{S}_w^n(\mathbb{F}_2) \times \mathcal{S}_w^n(\mathbb{F}_2),$$

$$\mathsf{pk}^* = (\mathbf{h}^* = \Pi_{\boldsymbol{\mu}^*}(\mathbf{h}_{t'}), \mathbf{s}^* = \mathbf{x}^* + \mathbf{h}^*\mathbf{y}^*) \in \mathcal{R} \times \mathcal{R}$$

where $\mathsf{sk}_{t'}^{(\mathsf{u})} = (\mathbf{x}_{t'}, \mathbf{y}_{t'}) \in \mathcal{S}_w^n(\mathbb{F}_2) \times \mathcal{S}_w^n(\mathbb{F}_2)$ and $\mathsf{pk}_{t'}^{(\mathsf{u})} = (\mathbf{h}_{t'}, \mathbf{s}_{t'}) \in \mathcal{R} \times \mathcal{R}$ satisfying $\mathbf{s}_{t'} = \mathbf{x}_{t'} + \mathbf{h}_{t'}\mathbf{y}_{t'}$. Besides $\mathsf{pk}_0^{(\mathsf{u})} = (\mathbf{h}_0, \mathbf{s}_0) \in \mathcal{R} \times \mathcal{R}$, $\mathsf{pk}_{t'}^{(\mathsf{u})} = (\mathbf{h}_{t'}, \mathbf{s}_{t'}) \in \mathcal{R} \times \mathcal{R}$ and $\mathsf{sk}_0^{(\mathsf{u})} = (\mathbf{x}_0, \mathbf{y}_0) \in \mathcal{S}_w^n(\mathbb{F}_2) \times \mathcal{S}_w^n(\mathbb{F}_2)$, $\mathsf{sk}_{t'}^{(\mathsf{u})} = (\mathbf{x}_{t'}, \mathbf{y}_{t'}) \in \mathcal{S}_w^n(\mathbb{F}_2) \times \mathcal{S}_w^n(\mathbb{F}_2)$ in game $\mathcal{G}_1$ are set exactly in the same way as in game $\mathcal{G}_0$. The n length vectors $\mathbf{h}_0, \mathbf{h}_{t'}, \mathbf{h}^* \in \mathbb{F}_{2,b_1}^n$ in both $\mathcal{G}_0$ and $\mathcal{G}_1$ as they have the same parity $b_1 = \mathbf{h}_0 \cdot \mathbf{1} \bmod 2$ by the property of $\Pi_{\boldsymbol{\mu}^*}$ by Proposition 3 for $\mathbf{h}_0 \in \mathcal{R}$ and the n length vectors $\mathbf{s}_0 = \mathbf{x}_0 + \mathbf{h}_0\mathbf{y}_0, \mathbf{s}_{t'} = \mathbf{x}_{t'} + \mathbf{h}_{t'}\mathbf{y}_{t'}$, $\mathbf{s}^* = \mathbf{x}^* + \mathbf{h}^*\mathbf{y}^* \in \mathbb{F}_{2,b_2}^n$ in both $\mathcal{G}_0$ and $\mathcal{G}_1$ (Proposition 1) as they have the same parity $b_2 = w + b_1 \times w \bmod 2$.
Here, we rely on computational hiding and statistical arguments. First, by the IND-CPA security of the underlying hqcPKE scheme, the update ciphertext up^* hides the true value of $\boldsymbol{\mu}^*$ from any PPT adversary. Therefore, from the adversary's perspective, $\boldsymbol{\mu}^*$ acts as a hidden uniform random variable.

Since $\boldsymbol{\mu}^*$ is hidden, the application of $(\Pi_{\boldsymbol{\mu}^*}, \mathsf{F}_{\boldsymbol{\mu}^*}) = \mathsf{Samp}(\boldsymbol{\mu}^*)$ to the previous key state acts as a random transformation over $\mathsf{Per}(\mathcal{R})$. By the distributional equivalence property of $\mathsf{Per}(\mathcal{R})$, the honestly derived key is statistically close to the uniformly sampled key $(\boldsymbol{\rho}^*, \boldsymbol{\gamma}^*)$, bounded by ε_1. Therefore, the adversary can only distinguish this game by either breaking the IND-CPA security of the ciphertext or detecting the statistical variance. Thus

$$| \Pr[S_0] - \Pr[S_1] | \leq \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{hqcPKE}}(1^\lambda) + \varepsilon_1.$$

- *Game* $\mathcal{G}_{\mathbf{2}}$: This game is identical to $\mathcal{G}_{\mathbf{1}}$ except for a change in the generation of the initial public key $\mathsf{pk}_0^{(\mathsf{u})} = (\mathbf{h}_0, \mathbf{s}_0) \in \mathbb{F}_{2,b_1}^n \times \mathbb{F}_{2,b_2}^n$ during the setup. The component $\mathbf{s}_0 = \mathbf{x}_0 + \mathbf{h}_0\mathbf{y}_0 \in \mathbb{F}_{2,b_2}^n$ is made independent of $\mathbf{h}_0 \in \mathbb{F}_{2,b_1}^n$ and replaced by a uniformly sampled vector $\mathbf{s}'_0 \xleftarrow{\$} \mathbb{F}_{2,b_2}^n$, thereby affecting the simulation of $\mathsf{CT}_{\mathbf{m}_b^*}$ in challenge phase and up^* in the reveal phase and $\mathsf{pk}_{i+1}^{(\mathsf{u})}, \mathsf{up}_{i+1}^{(\mathsf{u})}, \mathsf{sk}_{i+1}^{(\mathsf{u})}$ in the pre-challenge query and post-challenge query phase. The challenger does not set any initial secret key $\mathsf{sk}_0^{(\mathsf{u})}$ and $(\mathbf{h}_0, \mathbf{s}'_0) \in \mathbb{F}_{2,b_1}^n \times \mathbb{F}_{2,b_2}^n$ are unrelated unlike $\mathcal{G}_1$. Thus in Game $\mathcal{G}_2$,

$\mathsf{pk}_0^{(\mathsf{u})} = (\mathbf{h}_0, \mathbf{s}'_0) \in \mathbb{F}_{2,b_1}^n \times \mathbb{F}_{2,b_2}^n$,
$\mathsf{pk}_{i+1}^{(\mathsf{u})} = (\mathbf{h}_{i+1} = \Pi_{\boldsymbol{\mu}_i}(\mathbf{h}_i), \mathbf{s}'_{i+1} = \mathsf{F}_{\boldsymbol{\mu}_i}(\mathbf{s}'_i)) \in \mathbb{F}_{2,b_1}^n \times \mathbb{F}_{2,b_2}^n$,
$\mathsf{up}_{i+1}^{(\mathsf{u})} = \mathsf{CT}_{\boldsymbol{\mu}_i} = (\mathbf{u}_{\boldsymbol{\mu}_i} = \mathbf{r}'_1 + \mathbf{h}_i\mathbf{r}'_2, \mathbf{v}_{\boldsymbol{\mu}_i} = \boldsymbol{\mu}_i\mathbf{G} + \mathbf{s}'_i\mathbf{r}'_2 + \mathbf{e}') \leftarrow \mathsf{uhqcPKE.Enc}(\mathsf{pp}_{\mathsf{uhqcPKE}}, \mathsf{pk}_i^{(\mathsf{u})}, \boldsymbol{\mu}_i; \theta')$
$\mathsf{CT}_{\mathbf{m}_b^*} = (\mathbf{u}_{\mathbf{m}_b^*} = \mathbf{r}_1 + \mathbf{h}_t\mathbf{r}_2, \mathbf{v}_{\mathbf{m}_b^*} = \mathbf{m}_b^*\mathbf{G} + \mathbf{s}'_t\mathbf{r}_2 + \mathbf{e}) \leftarrow \mathsf{uhqcPKE.Enc}(\mathsf{pp}_{\mathsf{uhqcPKE}}, \mathsf{pk}_t^{(\mathsf{u})}, \mathbf{m}_b^*; \theta)$
$\mathsf{up}^* = \mathsf{CT}_{\boldsymbol{\mu}^*} = (\mathbf{u}_{\boldsymbol{\mu}^*} = \mathbf{r}''_1 + \mathbf{h}_{t'}\mathbf{r}''_2, \mathbf{v}_{\boldsymbol{\mu}^*} = \boldsymbol{\mu}^*\mathbf{G} + \mathbf{s}'_{t'}\mathbf{r}''_2 + \mathbf{e}'') \leftarrow \mathsf{uhqcPKE.Enc}(\mathsf{pp}_{\mathsf{uhqcPKE}}, \mathsf{pk}_{t'}^{(\mathsf{u})}, \boldsymbol{\mu}^*; \theta'')$
$\mathsf{sk}^* = (\boldsymbol{\rho}^*, \boldsymbol{\gamma}^*) \xleftarrow{\$} \mathcal{S}_w^n(\mathbb{F}_2) \times \mathcal{S}_w^n(\mathbb{F}_2)$
$\mathsf{pk}^* = (\mathbf{h}^* = \Pi_{\boldsymbol{\mu}^*}(\mathbf{h}_{t'}), \mathbf{s}^* = \boldsymbol{\rho}^* + \mathbf{h}^*\boldsymbol{\gamma}^*) \in \mathbb{F}_{2,b_1}^n \times \mathbb{F}_{2,b_2}^n$,

Here $(\Pi_{\boldsymbol{\mu}_i}, \mathsf{F}_{\boldsymbol{\mu}_i}) = \mathsf{Samp}(\boldsymbol{\mu}_i) \in \mathsf{Per}(\mathcal{R})$, $\boldsymbol{\mu}_i \in \mathbb{F}_2^k$ is the choice of randomness of adversary $\mathcal{A}$ for $i \in \{1, 2, \ldots, t-1\}$, $\mathsf{pk}_i^{(\mathsf{u})} = (\mathbf{h}_i, \mathbf{s}'_i) \in \mathbb{F}_{2,b_1}^n \times \mathbb{F}_{2,b_2}^n$ denotes the public key at epoch i, $\mathsf{P}(\theta') = (\mathbf{e}', \mathbf{r}'_1, \mathbf{r}'_2) \in \mathcal{S}_{w_e}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2)$, $\theta' \xleftarrow{\$} \mathbb{F}_2^\alpha$, $\mathsf{pk}_t^{(\mathsf{u})} = (\mathbf{h}_t, \mathbf{s}'_t) \in \mathbb{F}_{2,b_1}^n \times \mathbb{F}_{2,b_2}^n$, $\mathsf{P}(\theta) = (\mathbf{e}, \mathbf{r}_1, \mathbf{r}_2) \in \mathcal{S}_{w_e}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2)$, $\theta \xleftarrow{\$} \mathbb{F}_2^\alpha$. $\mathsf{pk}_{t'}^{(\mathsf{u})} = (\mathbf{h}_{t'}, \mathbf{s}_{t'}) \in \mathbb{F}_{2,b_1}^n \times \mathbb{F}_{2,b_2}^n$ is the public key at epoch t', $\mathsf{P}(\theta'') = (\mathbf{e}'', \mathbf{r}''_1, \mathbf{r}''_2) \in \mathcal{S}_{w_e}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2)$, $\theta'' \xleftarrow{\$} \mathbb{F}_2^\alpha$. $(\Pi_{\boldsymbol{\mu}^*}, \mathsf{F}_{\boldsymbol{\mu}^*}) = \mathsf{Samp}(\boldsymbol{\mu}^*) \in \mathsf{Per}(\mathcal{R})$ where $\boldsymbol{\mu}^* \xleftarrow{\$} \mathbb{F}_2^k$ is the randomness chosen by the challenger $\mathcal{C}$.

Claim 1: $| \Pr[S_1] - \Pr[S_2] | < \varepsilon_2$ for some negligible function ε_2.

- *Game* $\mathcal{G}_3$: This game proceeds in the same way as before except for a change in $\mathsf{up}_{i+1}^{(\mathsf{u})}$ during the pre-challenge and post-challenge query phase.

Instead of using $\mathcal{A}$'s choice of randomness $\boldsymbol{\mu}_i \in \mathbb{F}_2^k$ in generating $\mathsf{up}_{i+1}^{(\mathsf{u})}$ for $i = \{0, 1, 2, \ldots, t-1\}$ and invoking $\mathsf{uhqcPKE.Enc}$, the challenger $\mathcal{D}$ samples randomly weighted vectors $\hat{\mathbf{r}}_1', \hat{\mathbf{r}}_2' \xleftarrow{\$} \mathbb{F}_{2,w_r}^n$, $\hat{\mathbf{e}}' \xleftarrow{\$} \mathbb{F}_{2,w_e}^n$, computes $\boxed{\mathsf{up}_{i+1}^{(\mathsf{u})} = (\hat{\mathbf{r}}_1' + \mathbf{h}_i\hat{\mathbf{r}}_2', \boldsymbol{\mu}_i\mathbf{G} + \mathbf{s}_i'\hat{\mathbf{r}}_2' + \hat{\mathbf{e}}')}$ and sends it $\mathcal{A}$.

Claim 2: $|\Pr[S_2] - \Pr[S_3]| < \varepsilon_3$ for some negligible function ε_3.

- *Game* $\mathcal{G}_4$: This game is the same as the previous game $\mathcal{G}_3$ except for one change in the challenge phase. The challenge ciphertext is set as

$$\boxed{\mathsf{CT}_{\mathbf{m}_b^*} = (\mathbf{u}_{\mathbf{m}_b^*} = \hat{\mathbf{r}}_1 + \mathbf{h}_t\hat{\mathbf{r}}_2, \mathbf{v}_{\mathbf{m}_b^*} = \mathbf{m}_b^*\mathbf{G} + \mathbf{s}_t'\hat{\mathbf{r}}_2 + \hat{\mathbf{e}})}$$

by randomly sampling $\hat{\mathbf{r}}_1, \hat{\mathbf{r}}_2 \xleftarrow{\$} \mathbb{F}_{2,w_r}^n$, $\hat{\mathbf{e}} \xleftarrow{\$} \mathbb{F}_{2,w_e}^n$ where $\mathsf{pk}_t^{(\mathsf{u})} = (\mathbf{h}_t, \mathbf{s}_t) \in \mathbb{F}_{2,b_1}^n \times \mathbb{F}_{2,b_2}^n$ as in $\mathcal{G}_3$. The challenge ciphertext is $\mathcal{G}_3$ is

$$\mathsf{CT}_{\mathbf{m}_b^*} \longleftarrow \mathsf{uhqcPKE.Enc}(\mathsf{pp}_{\mathsf{uhqcPKE}}, \mathsf{pk}_i^{(\mathsf{u})}, \mathbf{m}_b^*; \theta)$$

where $\mathsf{CT}_{\mathbf{m}_b^*} = (\mathbf{u}_{\mathbf{m}_b^*} = \mathbf{r}_1 + \mathbf{h}_t\mathbf{r}_2, \mathbf{v}_{\mathbf{m}_b^*} = \mathbf{m}_b^*\mathbf{G} + \mathbf{s}_t'\mathbf{r}_2 + \mathbf{e})$ and $(\mathbf{r}_1, \mathbf{r}_2, \mathbf{e}) \in \mathcal{S}_{w_r}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2) \times \mathcal{S}_{w_e}^n(\mathbb{F}_2)$ is generated using randomness generator P on input $\theta \xleftarrow{\$} \mathbb{F}_2^\alpha$ sampled during the invocation of the encryption algorithm.

Claim 3: $|\Pr[S_3] - \Pr[S_4]| < \varepsilon_3$ for some negligible function ε_3.

- *Game* $\mathcal{G}_5$: This game proceeds as before except for a change in the up^* component in the reveal phase. Here, we replace the correctly weighted vector $\mathbf{r}_1'', \mathbf{r}_2''$ and $\mathbf{e}''$ with random vectors $\hat{\mathbf{r}}_1'', \hat{\mathbf{r}}_2'' \xleftarrow{\$} \mathbb{F}_{2,w_r}^n$, $\hat{\mathbf{e}}'' \xleftarrow{\$} \mathbb{F}_{2,w_e}^n$. Thus, $\boxed{\mathsf{up}^* = (\mathbf{u}^* = \hat{\mathbf{r}}_1' + \mathbf{h}_{t'}\hat{\mathbf{r}}_2', \mathbf{v}^* = \boldsymbol{\mu}^*\mathbf{G} + \mathbf{s}_{t'}'\hat{\mathbf{r}}_2' + \hat{\mathbf{e}}')}$.

Claim 4: $|\Pr[S_4] - \Pr[S_5]| = \mathsf{Adv}_{\mathcal{D}}^{\text{3-DQCSDP}}(1^\lambda) < \varepsilon_3$ for some negligible function ε_3.

In game $\mathcal{G}_5$ all values revealed to the adversary are uniformly randomized. In particular, the challenge ciphertext $\mathsf{CT}_{\mathbf{m}_b^*}$ is statistically independent of the message bit b as $\mathbf{u}_{\mathbf{m}_b^*}$ is fully independent of the bit b and $\mathbf{v}_{\mathbf{m}_b^*}$ is masked by a random vector $\mathbf{s}_{t'}'\hat{\mathbf{r}}_2' + \hat{\mathbf{e}}'$. Hence, $\Pr[S_5] = \frac{1}{2}$.

Thus, summing up all the results we get

$$\begin{aligned}|\Pr[S_0] - \Pr[S_5]| \leq &|\Pr[S_0] - \Pr[S_1]| + |\Pr[S_1] - \Pr[S_2]| \\ &+ |\Pr[S_2] - \Pr[S_3]| + |\Pr[S_3] - \Pr[S_4]| \\ &+ |\Pr[S_4] - \Pr[S_5]|\end{aligned}$$

Therefore,

$$| \Pr[S_0] - \frac{1}{2} | < \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{hqcPKE}}(1^\lambda) + \varepsilon_1 + \varepsilon_2 + 3\varepsilon_3 = \varepsilon$$

where $\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{hqcPKE}}(1^\lambda) + \varepsilon_1 + \varepsilon_2 + 3\varepsilon_3 = \varepsilon$. Therefore,

$$\mathsf{Adv}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{uhqcPKE},\mathcal{A}}(1^\lambda) = | \Pr[\mathsf{EXP}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{uhqcPKE},\mathcal{A}}(1^\lambda) = 1] - \frac{1}{2} | < \varepsilon$$

Hence, the theorem follows.

4.1 Proof of Claims

• *Proof of Claim 1:* We show that if the adversary $\mathcal{A}$ can distinguish between the games $\mathcal{G}_1$ and $\mathcal{G}_2$, then one can construct a distinguisher $\mathcal{D}$ for the 2-$\mathsf{DQCSDP}(n, w, b_1, b_2)$ (see Definition 13) problem. Let the 2-$\mathsf{DQCSDP}(n, w, b_1, b_2)$ challenger $\mathcal{B}$ provides an instance $\mathsf{Ins} = (\mathbf{H}, \mathbf{s}) \in \mathbb{F}^{n \times 2n}_{2,b_1} \times \mathbb{F}^n_{2,b_2}$ of the 2-$\mathsf{DQCSDP}(n, w, b_1, b_2)$ problem to the distinguisher $\mathcal{D}$ where $\mathbf{H} = (\mathbf{I}_n \ \mathsf{rot}(\mathbf{h}))$, $b_1 = \mathbf{h} \cdot \mathbf{1} \bmod 2$, $\mathbf{h} \in \mathbb{F}^n_{2,b_1}$, $b_2 = w + b_1 \times w \bmod 2$ and $\mathbf{s}$ is either a random element from $\mathbb{F}^n_{2,b_2}$ or of the form $\mathbf{s} = \mathbf{x} + \mathbf{h}\mathbf{y}$ for some $\mathbf{x}, \mathbf{y} \in \mathcal{S}^n_w(\mathbb{F}_2)$. The distinguisher $\mathcal{D}$ plays the role of the challenger to $\mathcal{A}$ and the simulates $\mathcal{A}'s$ view as follows.

– **Setup**: The distinguisher $\mathcal{D}$ sets

$$\mathsf{pp}_{\mathsf{uhqcPKE}} = (n, k, \alpha, \tau, w, w_e, w_r, \mathsf{Samp}, \mathbf{G}, \mathsf{P})$$

by choosing a τ error correcting decodable $[n, k]$ linear code C with a generator matrix $\mathbf{G}$, a positive integer a size α of encryption randomness, fixed weights w, w_e, w_r, a randomness generator $\mathsf{P} : \{0,1\}^* \longrightarrow \mathcal{S}^n_{w_e}(\mathbb{F}^n_2) \times \mathcal{S}^n_{w_r}(\mathbb{F}^n_2) \times \mathcal{S}^n_{w_r}(\mathbb{F}^n_2)$ and a sampling function $\mathsf{Samp} : \mathbb{F}^k_2 \longrightarrow \mathsf{Per}(\mathcal{R})$. It extracts $\mathbf{h}, \mathbf{s}$ from the given instance $\mathsf{Ins} = (\mathbf{H}, \mathbf{s})$, sets

$$\mathsf{pk}^{(\mathsf{u})}_0 = (\mathbf{h}_0, \mathbf{s}_0) = (\mathbf{h}, \mathbf{s}) \in \mathbb{F}^n_{2,b_1} \times \mathbb{F}^n_{2,b_2}$$

and forwards $\mathsf{pp}_{\mathsf{uhqcPKE}}$, $\mathsf{pk}^{(\mathsf{u})}_0$ to $\mathcal{A}$.

– **Pre-Challenge Query Phase**: The adversary $\mathcal{A}$ is allowed $\mathcal{O}^{\mathsf{uhqcPKE}}_{\mathsf{upd}}(\cdot)$ queries as described in the original game. On receiving a query $\boldsymbol{\mu}_i \in \mathbb{F}^k_2$ for $i \in \{0, 1, 2, \ldots, t-1\}$ from $\mathcal{A}$, the challenger $\mathcal{D}$ generates

$$(\mathsf{up}^{(\mathsf{u})}_{i+1}, \mathsf{pk}^{(\mathsf{u})}_{i+1}) \longleftarrow \mathsf{uhqcPKE.PkUpdate}(\mathsf{pp}_{\mathsf{uhqcPKE}}, \mathsf{pk}^{(\mathsf{u})}_i, \boldsymbol{\mu}_i)$$

and forwards it to $\mathcal{A}$ where

$$\mathsf{pk}^{(\mathsf{u})}_{i+1} = (\mathbf{h}_{i+1} = \Pi_{\boldsymbol{\mu}_i}(\mathbf{h}_i),\ \mathbf{s}_{i+1} = \mathsf{F}_{\boldsymbol{\mu}_i}(\mathbf{s}_i)) \in \mathbb{F}^n_{2,b_1} \times \mathbb{F}^n_{2,b_2},$$

$$\mathsf{up}_{i+1}^{(\mathsf{u})} = \mathsf{CT}_{\boldsymbol{\mu}_i} \longleftarrow \mathsf{uhqcPKE.Enc}(\mathsf{pp}_{\mathsf{uhqcPKE}}, \mathsf{pk}_i^{(\mathsf{u})}, \boldsymbol{\mu}_i; \theta')$$

where $\mathsf{CT}_{\boldsymbol{\mu}_i} = (\mathbf{u}_{\boldsymbol{\mu}_i} = \mathbf{r}'_1 + \mathbf{h}_i\mathbf{r}'_2, \mathbf{v}_{\boldsymbol{\mu}_i} = \boldsymbol{\mu}_i\mathbf{G} + \mathbf{s}_i\mathbf{r}'_2 + \mathbf{e}')$ and $(\Pi_{\boldsymbol{\mu}_i}, \mathsf{F}_{\boldsymbol{\mu}_i}) = \mathsf{Samp}(\boldsymbol{\mu}_i) \in \mathsf{Per}(\mathcal{R})$, $\theta' \in \mathbb{F}_2^{\alpha}$, $\mathsf{P}(\theta') = (\mathbf{e}', \mathbf{r}'_1, \mathbf{r}'_2) \in \mathcal{S}_{w_e}^n(\mathbb{F}_2^n) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2^n) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2^n)$ and $\mathsf{pk}_i^{(\mathsf{u})} = (\mathbf{h}_i, \mathbf{s}_i) \in \mathbb{F}_{2,b_1}^n \times \mathbb{F}_{2,b_2}^n$ for $i \in \{1, 2, \ldots, t-1\}$.

– **Challenge Phase**: On receiving a pair of challenge messages $(\mathbf{m}_0^*, \mathbf{m}_1^*) \in \mathbb{F}_2^k \times \mathbb{F}_2^k$ from $\mathcal{A}$ at epoch t, the distinguisher $\mathcal{D}$ samples a bit $b \xleftarrow{\$} \{0,1\}$, computes

$$\mathsf{CT}_{\mathbf{m}_b^*} \longleftarrow \mathsf{uhqcPKE.Enc}(\mathsf{pp}_{\mathsf{uhqcPKE}}, \mathsf{pk}_t^{(\mathsf{u})}, \mathbf{m}_b^*; \theta)$$

where $\mathsf{CT}_{\mathbf{m}_b^*} = (\mathbf{u}_{\mathbf{m}_b^*} = \mathbf{r}_1 + \mathbf{h}_t\mathbf{r}_2, \mathbf{v}_{\mathbf{m}_b^*} = \mathbf{m}_b^*\mathbf{G} + \mathbf{s}_t\mathbf{r}_2 + \mathbf{e})$ and forwards it to $\mathcal{A}$ where $\mathsf{P}(\theta) = (\mathbf{e}, \mathbf{r}_1, \mathbf{r}_2) \in \mathcal{S}_{w_e}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2)$, $\theta \in \mathbb{F}_2^{\alpha}$ and $\mathsf{pk}_t^{(\mathsf{u})} = (\mathbf{h}_t, \mathbf{s}_t) \in \mathbb{F}_{2,b_1}^n \times \mathbb{F}_{2,b_2}^n$.

– **Post-Challenge Query Phase**: The adversary $\mathcal{A}$ is allowed to make $\mathcal{O}_{\mathsf{upd}}^{\mathsf{uhqcPKE}}(\boldsymbol{\mu}_i)$ queries for $i \in \{t+1, \ldots, t'-1\}$ which is described and simulated in the pre-challenge query phase fot $\boldsymbol{\mu}_i \in \mathbb{F}_2^k$.

– **Reveal Phase**: The distinguisher $\mathcal{D}$ samples a k-bit string $\boldsymbol{\mu}^* \xleftarrow{\$} \mathbb{F}_2^k$ at epoch t' and picks

$$\mathsf{sk}^* = (\boldsymbol{\rho}^*, \boldsymbol{\gamma}^*) \xleftarrow{\$} \mathcal{S}_w^n(\mathbb{F}_2) \times \mathcal{S}_w^n(\mathbb{F}_2)$$

independent of $\boldsymbol{\mu}^*$, computes

$$\mathsf{up}^* = \mathsf{CT}_{\boldsymbol{\mu}^*} \leftarrow \mathsf{uhqcPKE.Enc}(\mathsf{pp}_{\mathsf{uhqcPKE}}, \mathsf{pk}_{t'}^{(\mathsf{u})}, \boldsymbol{\mu}^*; \theta'')$$

$$\mathsf{pk}^* = (\mathbf{h}^* = \Pi_{\boldsymbol{\mu}^*}(\mathbf{h}_{t'}), \mathbf{s}^* = \boldsymbol{\rho}^* + \mathbf{h}^*\boldsymbol{\gamma}^*) \in \mathbb{F}_{2,b_1}^n \times \mathbb{F}_{2,b_2}^n$$

where $\mathsf{CT}_{\boldsymbol{\mu}^*} = \left(\mathbf{u}_{\boldsymbol{\mu}^*} = \mathbf{r}''_1 + \mathbf{h}_{t'}\mathbf{r}''_2, \mathbf{v}_{\boldsymbol{\mu}^*} = \boldsymbol{\mu}^*\mathbf{G} + \mathbf{s}_{t'}\mathbf{r}''_2 + \mathbf{e}''\right)$ and forwards the tuple $(\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{up}^*)$ to $\mathcal{A}$ where $\theta'' \in \mathbb{F}_2^{\alpha}$, $\mathsf{P}(\theta'') = (\mathbf{e}'', \mathbf{r}''_1, \mathbf{r}''_2) \in \mathcal{S}_{w_e}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2)$, $\mathsf{pk}_{t'}^{(\mathsf{u})} = (\mathbf{h}_{t'}, \mathbf{s}_{t'}) \in \mathbb{F}_{2,b_1}^n \times \mathbb{F}_{2,b_2}^n$, $(\Pi_{\boldsymbol{\mu}^*}, \mathsf{F}_{\boldsymbol{\mu}^*}) = \mathsf{Samp}(\boldsymbol{\mu}^*) \in \mathsf{Per}(\mathcal{R})$.

– **Guess Phase**: After receiving a guess bit b' from $\mathcal{A}$ for bit b chosen by $\mathcal{D}$ in the challenge phase, $\mathcal{D}$ forwards $\mathcal{B}$, $\beta = 0$ if $b' \neq b$ and $\beta = 1$ if $b' = b$.

Observe that if the public key component $\mathbf{s}_i \in \mathbb{F}_{2,b_2}^n$ is sampled from the 2-$\mathsf{QCSDP}(n, w, b_1, b_2)$ distribution then $\mathbf{s}_i$ is of the form $\mathbf{s}_i = \mathbf{x}_i + \mathbf{h}_i\mathbf{y}_i$ for some unknown vectors $\mathbf{x}_i, \mathbf{y}_i \in \mathcal{S}_w^n(\mathbb{F}_2) \times \mathcal{S}_w^n(\mathbb{F}_2)$ which implies that $\mathbf{s}_{i+1} \in \mathbb{F}_{2,b_2}^n$ belongs to the 2-$\mathsf{QCSDP}(n, w, b_1, b_2)$ distribution as $\mathbf{s}_{i+1} = \mathsf{F}_{\boldsymbol{\mu}_i}(\mathbf{s}_i) = \mathsf{F}_{\boldsymbol{\mu}_i}(\mathbf{x}_i + \mathbf{h}_i\mathbf{y}_i) = \mathsf{F}_{\boldsymbol{\mu}_i}(\mathbf{x}_i) + \Pi_{\boldsymbol{\mu}_i}(\mathbf{h}_i)\mathsf{F}_{\boldsymbol{\mu}_i}(\mathbf{y}_i) = \mathbf{x}_{i+1} + \mathbf{h}_{i+1}\mathbf{y}_{i+1} \in \mathbb{F}_{2,b_2}^n$ where $\mathbf{x}_{i+1}, \mathbf{y}_{i+1} \in \mathcal{S}_w^n(\mathbb{F}_2)$ for all $i \in \{1, 2, \ldots, t'\}$. On the other hand, if $\mathbf{s}_i \in \mathbb{F}_{2,b_2}^n$ is sampled from the uniform distribution over $\mathbb{F}_{2,b_2}^n$ then $\mathbf{s}_{i+1} \in \mathbb{F}_{2,b_2}^n$ follows

the uniform distribution over $\mathbb{F}^n_{2,b_1} \times \mathbb{F}^n_{2,b_2}$ for all $i \in \{1, 2, \ldots, t'\}$.

In the setup phase, the public parameters $\mathsf{pp}_{\mathsf{uhqcPKE}}$ and the initial public key $\mathsf{pk}_0^{(\mathsf{u})} \in \mathbb{F}^n_{2,b_1} \times \mathbb{F}^n_{2,b_2}$ are identically distributed in $\mathcal{G}_1$ and $\mathcal{G}_2$. During the pre-challenge query phase and post challenge query phase, the generated update ciphertext $\mathsf{up}_{i+1}^{(\mathsf{u})} \in \mathcal{R} \times \mathcal{R}$ and updated public key $\mathsf{pk}_{i+1}^{(\mathsf{u})} \in \mathbb{F}^n_{2,b_1} \times \mathbb{F}^n_{2,b_2}$ follow the same distribution in both the games. Moreover, the distribution of the challenge ciphertext $\mathsf{CT}_{\mathbf{m}_b^*} \in \mathcal{R} \times \mathcal{R}$ in the challenge phase are identical in $\mathcal{G}_1$ and $\mathcal{G}_2$ and the simulation of $(\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{up}^*)$ in the reveal phase. Therefore, the adversary's view to the distributions are same in both the games which leads to the following two cases.

Case $\beta = 0$**:** If $\mathsf{Ins} = (\mathbf{H}, \mathbf{s} = \mathbf{s}')$ for $\mathbf{s}' \xleftarrow{\$} \mathbb{F}^n_{2,b_2}$, then $\mathcal{D}$ perfectly simulates game $\mathcal{G}_2$ and therefore

$$\Pr\left[\mathcal{D}(\mathbf{H}, \mathbf{s})) = 1 \;\mid\; \mathbf{H} = (\mathbf{I}_n \; \mathsf{rot}(\mathbf{h})),\; \mathbf{s} = \mathbf{s}' \xleftarrow{\$} \mathbb{F}^n_{2,b_2}\right] = \Pr[S_2].$$

Case $\beta = 1$**:** If $\mathsf{Ins} = (\mathbf{H}, \mathbf{s} = \mathbf{x} + \mathbf{h}\mathbf{y})$ for $\mathbf{s} = \mathbf{x} + \mathbf{h}\mathbf{y} \in \mathbb{F}^n_{2,b_2}$ with some $(\mathbf{x}, \mathbf{y}) \in \mathcal{S}^n_w(\mathbb{F}_2) \times \mathcal{S}^n_w(\mathbb{F}_2)$, then $\mathcal{D}$ simulates game $\mathcal{G}_1$ and hence,

$$\Pr\left[\mathcal{D}(\mathbf{H}, \mathbf{s}) = 1 \;\mid\; \mathbf{H} = (\mathbf{I}_n \; \mathsf{rot}(\mathbf{h})),\; \mathbf{s} = \mathbf{x} + \mathbf{h}\mathbf{y},\; \mathbf{x}, \mathbf{y} \xleftarrow{\$} \mathcal{S}^n_w(\mathbb{F}_2)\right] = \Pr[S_1].$$

Therefore, the advantage of distinguishing game $\mathcal{G}_1$ and game $\mathcal{G}_2$ is the same as the advantage of breaking the 2-$\mathsf{DQCSDP}(n, w, b_1, b_2)$ problem. Thus, $\mid \Pr[S_1] - \Pr[S_2] \mid = \mathsf{Adv}^{\text{2-DQCSDP}}_{\mathcal{D}}(1^\lambda) < \varepsilon_2$ by the 2-$\mathsf{DQCSDP}(n, w, b_1, b_2)$ assumption for some negligible function ε_2.

• *Proof of Claim 2:* We show that if there exists an adversary $\mathcal{A}$ who can distinguish between the games $\mathcal{G}_2$ and $\mathcal{G}_3$ then one can construct a distinguisher $\mathcal{D}$ for the 3-$\mathsf{DQCSDP}(n, w_e, w_r, b_1, b_2)$ problem (see Definition 15). Let $\mathcal{B}$ be the 3-$\mathsf{DQCSDP}(n, w_e, w_r, b_1, b_2)$ challenger who provides an instance $\mathsf{Ins} = (\mathbf{H}, \mathbf{a} = (\mathbf{a}_1, \mathbf{a}_2)) \in \mathbb{F}^{2n \times 3n}_{2,b_1,b_2} \times \left(\mathbb{F}^n_{2,b_3} \times \mathbb{F}^n_{2,b'_3}\right)$ of the 3-$\mathsf{DQCSDP}(n, w_e, w_r, b_1, b_2)$ problem to the distinguisher $\mathcal{D}$ where

$$\mathbf{H} = \begin{pmatrix} \mathbf{I}_n & \mathbf{0} & \mathsf{rot}(\mathbf{h}) \\ \mathbf{0} & \mathbf{I}_n & \mathsf{rot}(\mathbf{s}) \end{pmatrix},\; \mathbf{h} \in \mathbb{F}^n_{2,b_1},\; \mathbf{s} \in \mathbb{F}^n_{2,b_2},$$

$$b_1 = \mathbf{h} \cdot \mathbf{1} \bmod 2,\; b_2 = w + b_1 w \bmod 2, b_3 = w_r + b_1 w_r \bmod 2,$$

$$b'_3 = w_e + b_2 w_r \bmod 2, \mathbf{a} = \left(\mathbf{a}_1 = \tilde{\mathbf{r}}_1 + \mathbf{h}\tilde{\mathbf{r}}_2,\; \mathbf{a}_2 = \mathbf{s}\tilde{\mathbf{r}}_2 + \tilde{\mathbf{e}}\right) \in \mathbb{F}^n_{2,b_3} \times \mathbb{F}^n_{2,b'_3},$$

and $(\tilde{\mathbf{r}}_1, \tilde{\mathbf{r}}_2, \tilde{\mathbf{e}})$ is either a random tuple from $\mathbb{F}^n_{2,w_r} \times \mathbb{F}^n_{2,w_r} \times \mathbb{F}^n_{2,w_e}$ or from $\mathcal{S}^n_{w_r}(\mathbb{F}_2) \times \mathcal{S}^n_{w_r}(\mathbb{F}_2) \times \mathcal{S}^n_{w_e}(\mathbb{F}_2)$. The distinguisher $\mathcal{D}$ acts as a challenger to $\mathcal{A}$ and simulates its views as follows.

– **Setup**: The distinguisher $\mathcal{D}$ chooses a τ error correcting decodable $[n, k]$

linear code C with a generator matrix is $\mathbf{G}$, a size α of encryption randomness, fixed weights w, w_e, w_r, a randomness generator $\mathsf{P} : \{0,1\}^* \longrightarrow \mathcal{S}^n_{w_e}(\mathbb{F}^n_2) \times \mathcal{S}^n_{w_r}(\mathbb{F}^n_2) \times \mathcal{S}^n_{w_r}(\mathbb{F}^n_2)$, a sampling function $\mathsf{Samp} : \mathbb{F}^k_2 \longrightarrow \mathsf{Per}(\mathcal{R})$ and sets the public parameter as

$$\mathsf{pp}_{\mathsf{uhqcPKE}} = (n, k, \alpha, \tau, w, w_e, w_r, \mathsf{Samp}, \mathbf{G}, \mathsf{P}).$$

It extracts $\mathbf{h}, \mathbf{s}$ from $\mathbf{H}$ in the given instance $(\mathbf{H}, \mathbf{a})$, sets

$$\mathsf{pk}^{(\mathsf{u})}_0 = (\mathbf{h}_0, \mathbf{s}_0) = (\mathbf{h}, \mathbf{s}) \in \mathbb{F}^n_{2,b_1} \times \mathbb{F}^n_{2,b_2}$$

and forwards $\mathsf{pp}_{\mathsf{uhqcPKE}}$, $\mathsf{pk}^{(\mathsf{u})}_0$ to $\mathcal{A}$. The distinguisher sets $\mathbf{a}_{1,0} = \mathbf{a}_1$ and $\mathbf{a}_{2,0} = \mathbf{a}_2$ extracting $\mathbf{a} = (\mathbf{a}_1, \mathbf{a}_2) \in \mathbb{F}^n_{2,b_3} \times \mathbb{F}^n_{2,b'_3}$ from the given instance $\mathsf{Ins} = (\mathbf{H}, \mathbf{a})$. $\mathcal{D}$ initializes a list $\mathbf{a}_\mathsf{List} = \mathbf{a}_\mathsf{List} \cup \{\mathbf{a}_{1,0}, \mathbf{a}_{2,0}\}$.

– **Pre-Challenge Query Phase**: On receiving a query $\boldsymbol{\mu}_i \in \mathbb{F}^k_2$ for $i = \{0, 1, 2, \ldots, t-1\}$ from $\mathcal{A}$, the distinguisher $\mathcal{D}$ updates the list $\mathbf{a}_\mathsf{List} = \mathbf{a}_\mathsf{List} \cup \{\mathbf{a}_{1,i}, \mathbf{a}_{2,i}\}$ and computes

$$\mathbf{a}_{1,i+1} = \mathsf{\Pi}_{\boldsymbol{\mu}_i}(\mathbf{a}_{1,i}),\ \mathbf{a}_{2,i+1} = \mathsf{F}_{\boldsymbol{\mu}_i}(\mathbf{a}_{2,i}),$$

$$\boxed{\mathsf{up}^{(\mathsf{u})}_{i+1} = (\mathbf{u}_{\boldsymbol{\mu}_i}, \mathbf{v}_{\boldsymbol{\mu}_i}) = (\mathbf{a}_{1,i}, \boldsymbol{\mu}_i\mathbf{G} + \mathbf{a}_{2,i}) \in \mathcal{R} \times \mathcal{R},}$$

$$\mathsf{pk}^{(\mathsf{u})}_{i+1} = (\mathsf{\Pi}_{\boldsymbol{\mu}_i}(\mathbf{h}_i), \mathsf{F}_{\boldsymbol{\mu}_i}(\mathbf{s}_i)) = (\mathbf{h}_{i+1}, \mathbf{s}_{i+1}) \in \mathcal{R} \times \mathcal{R}$$

and forwards $(\mathsf{up}^{(\mathsf{u})}_{i+1}, \mathsf{pk}^{(\mathsf{u})}_{i+1})$ to $\mathcal{A}$ where $\mathsf{pk}^{(\mathsf{u})}_i = (\mathbf{h}_i, \mathbf{s}_i) \in \mathbb{F}^n_{2,b_1} \times \mathbb{F}^n_{2,b_2}$, $(\mathsf{\Pi}_{\boldsymbol{\mu}_i}, \mathsf{F}_{\boldsymbol{\mu}_i}) = \mathsf{Samp}(\boldsymbol{\mu}_i) \in \mathsf{Per}(\mathcal{R})$. Note that $\mathbf{a}_{1,i}, \mathbf{a}_{2,i}$ are the updated instance component at epoch i and $\mathbf{a}_{1,0} = \mathbf{a}_1,\ \mathbf{a}_{2,0} = \mathbf{a}_2$.

– **Challenge Phase**: On receiving challenge message pair $(\mathbf{m}^*_0, \mathbf{m}^*_1) \in \mathbb{F}^k_2 \times \mathbb{F}^k_2$ from $\mathcal{A}$ at epoch t, the distinguisher $\mathcal{D}$ samples a bit $b \xleftarrow{\$} \{0,1\}$, computes

$$\mathsf{CT}_{\mathbf{m}^*_b} \leftarrow \mathsf{uhqcPKE.Enc}(\mathsf{pp}_{\mathsf{uhqcPKE}}, \mathsf{pk}^{(\mathsf{u})}_t, \mathbf{m}^*_b; \theta),$$

where $\mathsf{CT}_{\mathbf{m}^*_b} = (\mathbf{u}_{\mathbf{m}^*_b} = \mathbf{r}_1 + \mathbf{h}_t\mathbf{r}_2, \mathbf{v}_{\mathbf{m}^*_b} = \mathbf{m}^*_b\mathbf{G} + \mathbf{s}_t\mathbf{r}_2 + \mathbf{e})$ and forwards it to $\mathcal{A}$ where $\theta \in \mathbb{F}^\alpha_2$, $\mathsf{P}(\theta) = (\mathbf{e}, \mathbf{r}_1, \mathbf{r}_2) \in \mathcal{S}^n_{w_e}(\mathbb{F}_2) \times \mathcal{S}^n_{w_r}(\mathbb{F}_2) \times \mathcal{S}^n_{w_r}(\mathbb{F}_2)$ and $\mathsf{pk}^{(\mathsf{u})}_t = (\mathbf{h}_t, \mathbf{s}_t) \in \mathbb{F}^n_{2,b_1} \times \mathbb{F}^n_{2,b_2}$.

– **Post-Challenge Query Phase**: The adversary $\mathcal{A}$ can make $\mathcal{O}^{\mathsf{uhqcPKE}}_{\mathsf{upd}}(\boldsymbol{\mu}_i)$ queries for $i = \{t, t+1, \cdots, t'-1\}$ which are simulated in the same manner by $\mathcal{D}$ as described in the pre-challenge query phase for $\boldsymbol{\mu}_i \in \mathbb{F}^k_2$.

– **Reveal Phase**: The distinguisher $\mathcal{D}$ samples $\boldsymbol{\mu}^* \xleftarrow{\$} \mathbb{F}^k_2$ at epoch $t'+1$ and selects

$$\mathsf{sk}^* = (\boldsymbol{\rho}^*, \boldsymbol{\gamma}^*) \xleftarrow{\$} \mathcal{S}^n_w(\mathbb{F}_2) \times \mathcal{S}^n_w(\mathbb{F}_2)$$

$$\mathsf{pk}^* = (\mathbf{h}^* = \mathsf{\Pi}_{\boldsymbol{\mu}^*}(\mathbf{h}_{t'}), \mathbf{s}^* = \boldsymbol{\rho}^* + \mathbf{h}^*\boldsymbol{\gamma}^*) \in \mathbb{F}^n_{2,b_1} \times \mathbb{F}^n_{2,b_2}$$

$$\mathsf{up}^* = \mathsf{CT}_{\boldsymbol{\mu}^*} \leftarrow \mathsf{uhqcPKE.Enc}(\mathsf{pp}_{\mathsf{uhqcPKE}}, \mathsf{pk}_{t'}^{(\mathsf{u})}, \boldsymbol{\mu}^*; \theta''),$$

where $\mathsf{CT}_{\boldsymbol{\mu}^*} = (\mathbf{u}_{\boldsymbol{\mu}^*} = \mathbf{r}_1'' + \mathbf{h}_{t'}\mathbf{r}_2'', \mathbf{v}_{\boldsymbol{\mu}^*} = \boldsymbol{\mu}^*\mathbf{G} + \mathbf{s}_{t'}\mathbf{r}_2'' + \mathbf{e}'')$ and forwards $(\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{up}^*)$ to the adversary $\mathcal{A}$ where $(\Pi_{\boldsymbol{\mu}^*}, \mathsf{F}_{\boldsymbol{\mu}^*}) = \mathsf{Samp}(\boldsymbol{\mu}^*) \in \mathsf{Per}(\mathcal{R})$, $\mathsf{pk}_{t'}^{(\mathsf{u})} = (\mathbf{h}_{t'}, \mathbf{s}_{t'}) \in \mathcal{R} \times \mathcal{R}$, $\theta'' \xleftarrow{\$} \mathbb{F}_2^{\alpha}$, $\mathsf{P}(\theta'') = (\mathbf{e}'', \mathbf{r}_1'', \mathbf{r}_2'') \in \mathcal{S}_{w_e}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2)$.

– **Guess Phase**: After receiving a guess bit b' from $\mathcal{A}$ for bit b chosen by $\mathcal{D}$ in the challenge phase, $\mathcal{D}$ forwards to $\mathcal{B}$, $\beta = 0$ if $b' \neq b$ and $\beta = 1$ if $b' = b$.

Note that $\mathcal{D}$ faithfully simulates the experiment using the given 3-$\mathsf{DQCSDP}(n, w_e, w_r, b_1, b_2)$ instance. In the setup phase, the generated public parameters $\mathsf{pp}_{\mathsf{uhqcPKE}}$ and the initial public key $\mathsf{pk}_0^{(\mathsf{u})} = (\mathbf{h}_0, \mathbf{s}_0) \in \mathbb{F}_{2,b_1}^n \times \mathbb{F}_{2,b_2}^n$ each have identical distribution in game $\mathcal{G}_2$ and game $\mathcal{G}_3$. During the pre-challenge query phase and post-challenge query phase, the update ciphertext $\mathsf{up}_{i+1}^{(\mathsf{u})} \in \mathcal{R} \times \mathcal{R}$ and updated public key $\mathsf{pk}_{i+1}^{(\mathsf{u})} = (\mathbf{h}_{i+1}, \mathbf{s}_{i+1}) \in \mathbb{F}_{2,b_1}^n \times \mathbb{F}_{2,b_2}^n$ are simulated following to the same distribution in both the games. Likewise, the challenge ciphertext $\mathsf{CT}_{\mathbf{m}_b^*} \in \mathcal{R} \times \mathcal{R}$ in the challenge phase as the simulation of $(\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{up}^*)$ in the reveal phase are identically distributed in game $\mathcal{G}_2$ and game $\mathcal{G}_3$. Consequently, the adversary's view to the distributions are same in both games which yields the following two cases.

Case $\beta = 0$: If $\mathsf{Ins} = (\mathbf{H}, \mathbf{a})$ for $\mathbf{H} = \begin{pmatrix} \mathbf{I}_n & \mathbf{0} & \mathsf{rot}(\mathbf{h}) \\ \mathbf{0} & \mathbf{I}_n & \mathsf{rot}(\mathbf{s}) \end{pmatrix}$, $\mathbf{a} = (\mathbf{a}_1 = \tilde{\mathbf{r}}_1 + \mathbf{h}\tilde{\mathbf{r}}_2, \mathbf{a}_2 = \mathbf{s}\tilde{\mathbf{r}}_2 + \tilde{\mathbf{e}}) \in \mathbb{F}_{2,b_3}^n \times \mathbb{F}_{2,b_3'}^n$, $\tilde{\mathbf{r}}_1, \tilde{\mathbf{r}}_2 \xleftarrow{\$} \mathbb{F}_{2,w_r}^n$, $\tilde{\mathbf{e}} \xleftarrow{\$} \mathbb{F}_{2,w_e}^n$ then the simulation follows as in game $\mathcal{G}_3$ and therefore,

$$\Pr\Big[\mathcal{D}(\mathbf{H}, \mathbf{a}) = 1 \;\Big|\; \mathbf{H} = \begin{pmatrix} \mathbf{I}_n & \mathbf{0} & \mathsf{rot}(\mathbf{h}) \\ \mathbf{0} & \mathbf{I}_n & \mathsf{rot}(\mathbf{s}) \end{pmatrix},$$
$$\mathbf{a} = \big(\mathbf{a}_1 = \tilde{\mathbf{r}}_1 + \mathbf{h}\tilde{\mathbf{r}}_2,\ \mathbf{a}_2 = \mathbf{s}\tilde{\mathbf{r}}_2 + \tilde{\mathbf{e}}\big) \in \mathbb{F}_{2,b_3}^n \times \mathbb{F}_{2,b_3'}^n, \tilde{\mathbf{r}}_1, \tilde{\mathbf{r}}_2 \xleftarrow{\$} \mathbb{F}_{2,w_r}^n, \tilde{\mathbf{e}} \xleftarrow{\$} \mathbb{F}_{2,w_e}^n\Big]$$
$$= \Pr[S_3].$$

Case $\beta = 1$: If $\mathsf{Ins} = (\mathbf{H}, \mathbf{a})$ for $\mathbf{H} = \begin{pmatrix} \mathbf{I}_n & \mathbf{0} & \mathsf{rot}(\mathbf{h}) \\ \mathbf{0} & \mathbf{I}_n & \mathsf{rot}(\mathbf{s}) \end{pmatrix}$, $\mathbf{a} = (\tilde{\mathbf{r}}_1 + \mathbf{h}\tilde{\mathbf{r}}_2, \mathbf{s}\tilde{\mathbf{r}}_2 + \tilde{\mathbf{e}})$, $\tilde{\mathbf{r}}_1, \tilde{\mathbf{r}}_2 \xleftarrow{\$} \mathcal{S}_{w_r}^n(\mathbb{F}_2)$, $\tilde{\mathbf{e}} \xleftarrow{\$} \mathcal{S}_{w_e}^n(\mathbb{F}_2)$ then the simulation follows as in game $\mathcal{G}_2$ and therefore

$$\Pr\Big[\mathcal{D}(\mathbf{H}, \mathbf{a}) = 1 \;\Big|\; \mathbf{H} = \begin{pmatrix} \mathbf{I}_n & \mathbf{0} & \mathsf{rot}(\mathbf{h}) \\ \mathbf{0} & \mathbf{I}_n & \mathsf{rot}(\mathbf{s}) \end{pmatrix},$$
$$\mathbf{a} = \big(\mathbf{a}_1 = \tilde{\mathbf{r}}_1 + \mathbf{h}\tilde{\mathbf{r}}_2,\ \mathbf{a}_2 = \mathbf{s}\tilde{\mathbf{r}}_2 + \tilde{\mathbf{e}}\big) \in \mathbb{F}_{2,b_3}^n \times \mathbb{F}_{2,b_3'}^n, \tilde{\mathbf{r}}_1, \tilde{\mathbf{r}}_2 \xleftarrow{\$} \mathcal{S}_{w_r}^n(\mathbb{F}_2), \tilde{\mathbf{e}} \xleftarrow{\$} \mathcal{S}_{w_e}^n(\mathbb{F}_2)\Big]$$
$$= \Pr[S_2].$$

Note that

$$\begin{pmatrix} \mathbf{I}_n & \mathbf{0} & \mathsf{rot}(\mathbf{h}) \\ \mathbf{0} & \mathbf{I}_n & \mathsf{rot}(\mathbf{s}) \end{pmatrix} (\tilde{\mathbf{r}}_1, \tilde{\mathbf{e}}, \tilde{\mathbf{r}}_2)^T = (\tilde{\mathbf{r}}_1 + \mathbf{h}\tilde{\mathbf{r}}_2,\ \tilde{\mathbf{e}} + \mathbf{s}\tilde{\mathbf{r}}_2)^T = (\mathbf{a}_1, \mathbf{a}_2)^T.$$

If the instance $\mathsf{Ins} = (\mathbf{H}, \mathbf{a})$ is sampled from the 3-$\mathsf{QCSDP}(n, w_e, w_r, b_1, b_2)$ distribution then $\mathbf{a}_1 = \tilde{\mathbf{r}}_1 + \mathbf{h}\tilde{\mathbf{r}}_2,\ \mathbf{a}_2 = \mathbf{s}\tilde{\mathbf{r}}_2 + \tilde{\mathbf{e}}$ for some $\tilde{\mathbf{r}}_1, \tilde{\mathbf{r}}_2 \xleftarrow{\$} \mathcal{S}^n_{w_r}(\mathbb{F}_2), \tilde{\mathbf{e}} \xleftarrow{\$} \mathcal{S}^n_{w_e}(\mathbb{F}_2)$ and therefore

$$\begin{aligned}(\mathbf{u}_{\boldsymbol{\mu}_i}, \mathbf{v}_{\boldsymbol{\mu}_i} - \boldsymbol{\mu}_i\mathbf{G}) &= (\mathbf{a}_{1,i} = \Pi_{\boldsymbol{\mu}_{i-1}}(\mathbf{a}_{1,i-1}), \mathbf{a}_{2,i} = \mathsf{F}_{\boldsymbol{\mu}_{i-1}}(\mathbf{a}_{2,i-1})) \\ &= (\tilde{\mathbf{r}}'_1 + \mathbf{h}_i\tilde{\mathbf{r}}'_2,\ \tilde{\mathbf{e}}' + \mathbf{s}_i\tilde{\mathbf{r}}'_2)\end{aligned}$$

for some $\tilde{\mathbf{r}}'_1, \tilde{\mathbf{r}}'_2 \in \mathcal{S}^n_{w_r}(\mathbb{F}_2)$, $\tilde{\mathbf{e}}' \in \mathcal{S}^n_{w_e}(\mathbb{F}_2)$ by the Proposition 3 and Theorem 5 where $\mathbf{a}_{1,i-1},\ \mathbf{a}_{2,i-1}$ the updated instance components formed by extracting $\mathbf{a}_1 = \mathbf{a}_{1,0}, \mathbf{a}_2 = \mathbf{a}_{2,0}$ from the instance and $(\Pi_{\boldsymbol{\mu}_{i-1}}, \mathsf{F}_{\boldsymbol{\mu}_{i-1}}) = \mathsf{Samp}(\boldsymbol{\mu}_{i-1}) \in \mathsf{Per}(\mathcal{R})$. Then

$$\mathsf{up}^{(\mathsf{u})}_{i+1} = \mathsf{CT}_{\boldsymbol{\mu}_i} = (\mathbf{a}_{1,i}, \boldsymbol{\mu}_i\mathbf{G} + \mathbf{a}_{2,i}) = (\mathbf{u}_{\boldsymbol{\mu}_i}, = \tilde{\mathbf{r}}'_1 + \mathbf{h}_i\tilde{\mathbf{r}}'_2, \mathbf{v}_{\boldsymbol{\mu}_i} = \boldsymbol{\mu}_i\mathbf{G} + \mathbf{s}_i\tilde{\mathbf{r}}'_2 + \tilde{\mathbf{e}}').$$

Hence the simulation of $\mathsf{up}^{(\mathsf{u})}_{i+1}$ is follows the same distribution as that in $\mathcal{G}_2$. On the other hand if the instance $\mathsf{Ins} = (\mathbf{H}, \mathbf{a})$ is sampled from the uniform distribution over $\mathbb{F}^{2n\times 3n}_{2,b_1,b_2} \times \left(\mathbb{F}^n_{2,b_3} \times \mathbb{F}^n_{2,b'_3}\right)$ then $\mathbf{a}_1 = \tilde{\mathbf{r}}_1 + \mathbf{h}\tilde{\mathbf{r}}_2,\ \mathbf{a}_2 = \mathbf{s}\tilde{\mathbf{r}}_2 + \tilde{\mathbf{e}}$ for some $\tilde{\mathbf{r}}_1, \tilde{\mathbf{r}}_2 \xleftarrow{\$} \mathbb{F}^n_{2,w_r}, \tilde{\mathbf{e}} \xleftarrow{\$} \mathbb{F}^n_{2,w_e}$ and therefore

$$\begin{aligned}(\mathbf{u}_{\boldsymbol{\mu}_i}, \mathbf{v}_{\boldsymbol{\mu}_i} - \boldsymbol{\mu}_i\mathbf{G}) &= (\mathbf{a}_{1,i} = \Pi_{\boldsymbol{\mu}_{i-1}}(\mathbf{a}_{1,i-1}), \mathbf{a}_{2,i} = \mathsf{F}_{\boldsymbol{\mu}_{i-1}}(\mathbf{a}_{2,i-1})) \\ &= (\tilde{\mathbf{r}}'_1 + \mathbf{h}_i\tilde{\mathbf{r}}'_2,\ \tilde{\mathbf{e}}' + \mathbf{s}_i\tilde{\mathbf{r}}'_2)\end{aligned}$$

for some $\tilde{\mathbf{r}}'_1, \tilde{\mathbf{r}}'_2 \in \mathbb{F}^n_{2,w_r}$, $\tilde{\mathbf{e}}' \in \mathbb{F}^n_{2,w_e}$ by the Proposition 3 and Theorem 5 where $\mathbf{a}_{1,i-1},\ \mathbf{a}_{2,i-1}$ the updated instance components formed by extracted $\mathbf{a}_1 = \mathbf{a}_{1,0}, \mathbf{a}_2 = \mathbf{a}_{2,0}$ from the instance and $(\Pi_{\boldsymbol{\mu}_{i-1}}, \mathsf{F}_{\boldsymbol{\mu}_{i-1}}) = \mathsf{Samp}(\boldsymbol{\mu}_{i-1}) \in \mathsf{Per}(\mathcal{R})$. Then

$$\mathsf{up}^{(\mathsf{u})}_{i+1} = \mathsf{CT}_{\boldsymbol{\mu}_i} = (\mathbf{a}_{1,i}, \boldsymbol{\mu}_i\mathbf{G} + \mathbf{a}_{2,i}) = (\mathbf{u}_{\boldsymbol{\mu}_i} = \tilde{\mathbf{r}}'_1 + \mathbf{h}_i\tilde{\mathbf{r}}'_2, \mathbf{v}_{\boldsymbol{\mu}_i} = \boldsymbol{\mu}_i\mathbf{G} + + \mathbf{s}_i\tilde{\mathbf{r}}'_2 + \tilde{\mathbf{e}}').$$

Hence the simulation of $\mathsf{up}^{(\mathsf{u})}_{i+1}$ is follows the same distribution as in $\mathcal{G}_3$. Thus the probability of distinguishing game $\mathcal{G}_2$ and game $\mathcal{G}_3$ is same as the advantage of breaking the 3-$\mathsf{DQCSDP}(n, w_e, w_r, b_1, b_2)$ problem. Hence, $|\Pr[S_2] - \Pr[S_3]| = \mathsf{Adv}^{\text{3-DQCSDP}}_{\mathcal{D}}(1^\lambda) < \varepsilon_3$ by the 3-$\mathsf{DQCSDP}(n, w_e, w_r, b_1, b_2)$ assumption for some negligible function ε_3.

• *Proof of Claim 3:* We show that if there exists an adversary $\mathcal{A}$ who can distinguish between the games $\mathcal{G}_3$ and $\mathcal{G}_4$ with advantage ε_3 then one can build a distinguisher $\mathcal{D}$ for the 3-$\mathsf{DQCSDP}(n, w_e, w_r, b_1, b_2)$ problem with advantage ε_3. Let $\mathcal{B}$ be the 3-$\mathsf{DQCSDP}(n, w_e, w_r, b_1, b_2)$ challenger who provides an 3-$\mathsf{DQCSDP}(n, w_e, w_r, b_1, b_2)$ instance $\mathsf{Ins} = (\mathbf{H}, \mathbf{a} = (\mathbf{a}_1, \mathbf{a}_2)) \in$

$\mathbb{F}_{2,b_1,b_2}^{2n\times 3n} \times \left(\mathbb{F}_{2,b_3}^{n} \times \mathbb{F}_{2,b_3'}^{n}\right)$ to the distinguisher $\mathcal{D}$ where

$$\mathbf{H} = \begin{pmatrix} \mathbf{I}_n & \mathbf{0} & \mathsf{rot}(\mathbf{h}) \\ \mathbf{0} & \mathbf{I}_n & \mathsf{rot}(\mathbf{s}) \end{pmatrix},\ \mathbf{h} \in \mathbb{F}_{2,b_1}^n,\ \mathbf{s} \in \mathbb{F}_{2,b_2}^n, b_1 = \mathbf{h}\cdot\mathbf{1} \bmod 2,$$
$$b_2 = w + b_1 w \bmod 2, b_3 = w_r + b_1 w_r \bmod 2,\ b_3' = w_e + b_2 w_r \bmod 2,$$
$$\mathbf{a} = \left(\mathbf{a}_1 = \tilde{\mathbf{r}}_1 + \mathbf{h}\tilde{\mathbf{r}}_2,\ \mathbf{a}_2 = \mathbf{s}\tilde{\mathbf{r}}_2 + \tilde{\mathbf{e}}\right) \in \mathbb{F}_{2,b_3}^n \times \mathbb{F}_{2,b_3'}^n,$$

and $(\tilde{\mathbf{r}}_1, \tilde{\mathbf{r}}_2, \tilde{\mathbf{e}})$ is either a random tuple from $\mathbb{F}_{2,w_r}^n \times \mathbb{F}_{2,w_r}^n \times \mathbb{F}_{2,w_e}^n$ or from $\mathcal{S}_{w_r}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2) \times \mathcal{S}_{w_e}^n(\mathbb{F}_2)$. The distinguisher $\mathcal{D}$ acts as a challenger to $\mathcal{A}$ and simulates its views as follows.

– **Setup**: The distinguisher $\mathcal{D}$ chooses a τ error correcting decodable $[n, k]$ linear code C with generator matrix $\mathbf{G}$. It also chooses an encryption randomness size α, fixed weights w, w_e, w_r along with $\mathsf{P} : \{0,1\}^* \longrightarrow \mathcal{S}_{w_e}^n(\mathbb{F}_2^n) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2^n) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2^n)$ a randomness generator and a sampling function $\mathsf{Samp} : \mathbb{F}_2^k \longrightarrow \mathsf{Per}(\mathcal{R})$ and sets the public parameter

$$\mathsf{pp}_{\mathsf{uhqcPKE}} = (n, k, \alpha, \tau, w, w_e, w_r, \mathsf{Samp}, \mathbf{G}, \mathsf{P}).$$

It sets

$$\mathsf{pk}_0^{(\mathsf{u})} = (\mathbf{h}_0, \mathbf{s}_0) = (\mathbf{h}, \mathbf{s}) \in \mathbb{F}_{2,b_1}^n \times \mathbb{F}_{2,b_2}^n$$

extracting $\mathbf{h}, \mathbf{s}$ from $\mathbf{H}$ in the instance $\mathsf{Ins} = (\mathbf{H}, \mathbf{a})$ and forwards $\mathsf{pp}_{\mathsf{uhqcPKE}}$, $\mathsf{pk}_0^{(\mathsf{u})}$ to $\mathcal{A}$. The distinguisher sets $\mathbf{a}_{1,0} = \mathbf{a}_1$ and $\mathbf{a}_{2,0} = \mathbf{a}_2$ where $\mathbf{a} = (\mathbf{a}_1, \mathbf{a}_2) \in \mathbb{F}_{2,b_3}^n \times \mathbb{F}_{2,b_3'}^n$ is extracted from the given instance $(\mathbf{H}, \mathbf{a})$. $\mathcal{D}$ initializes a list $\mathbf{a}_\mathsf{List} = \mathbf{a}_\mathsf{List} \cup \{\mathbf{a}_{1,0}, \mathbf{a}_{2,0}\}$.

– **Pre-Challenge Query Phase**: On receiving a query $\boldsymbol{\mu}_i \in \mathbb{F}_2^k$ for $i = \{0, 1, 2, \ldots, t-1\}$ from $\mathcal{A}$ the distinguisher $\mathcal{D}$ updates the list $\mathbf{a}_\mathsf{List} = \mathbf{a}_\mathsf{List} \cup \{\mathbf{a}_{1,i}, \mathbf{a}_{2,i}\}$ and computes

$$\mathsf{up}_{i+1}^{(\mathsf{u})} = \mathsf{CT}_{\boldsymbol{\mu}_i} = (\mathbf{r}_1' + \mathbf{h}_i\mathbf{r}_2', \boldsymbol{\mu}_i\mathbf{G} + \mathbf{s}_i\mathbf{r}_2' + \mathbf{e}'),$$

$$\mathsf{pk}_{i+1}^{(\mathsf{u})} = (\Pi_{\boldsymbol{\mu}_i}(\mathbf{h}_i) = \mathbf{h}_{i+1}, \mathsf{F}_{\boldsymbol{\mu}_i}(\mathbf{s}_i) = \mathbf{s}_{i+1}) \in \mathbb{F}_{2,b_1}^n \times \mathbb{F}_{2,b_2}^n$$
$$\mathbf{a}_{1,i+1} = \Pi_{\boldsymbol{\mu}_i}(\mathbf{a}_{1,i}),\ \mathbf{a}_{2,i+1} = \mathsf{F}_{\boldsymbol{\mu}_i}(\mathbf{a}_{2,i})$$

and forwards $\mathsf{up}_{i+1}^{(\mathsf{u})}$, $\mathsf{pk}_{i+1}^{(\mathsf{u})}$ to $\mathcal{A}$ where $\mathbf{r}_1', \mathbf{r}_2' \xleftarrow{\$} \mathbb{F}_{2,w_r}^n, \mathbf{e}' \xleftarrow{\$} \mathbb{F}_{2,w_e}^n, (\Pi_{\boldsymbol{\mu}_i}, \mathsf{F}_{\boldsymbol{\mu}_i}) = \mathsf{Samp}(\boldsymbol{\mu}_i) \in \mathsf{Per}(\mathcal{R})$, $\mathsf{pk}_i^{(\mathsf{u})} = (\mathbf{h}_i, \mathbf{s}_i) \in \mathbb{F}_{2,b_1}^n \times \mathbb{F}_{2,b_2}^n$ and $\mathbf{a}_{1,i}, \mathbf{a}_{2,i}$ are the updated instance components at epoch i.

– **Challenge Phase**: The adversary $\mathcal{A}$ submits a message pair $(\mathbf{m}_0^*, \mathbf{m}_1^*) \in \mathbb{F}_2^k \times \mathbb{F}_2^k$ to $\mathcal{D}$ at epoch t who in return samples a bit $b \xleftarrow{\$} \{0,1\}$, computes

$$\boxed{\mathsf{CT}_{\mathbf{m}_b^*} = (\mathbf{u}_{\mathbf{m}_b^*} = \mathbf{a}_{1,t},\ \mathbf{v}_{\mathbf{m}_b^*} = \mathbf{m}_b^*\mathbf{G} + \mathbf{a}_{2,t}) \in \mathcal{R} \times \mathcal{R}}$$

and submits it to $\mathcal{A}$ where $\mathbf{a}_{1,t},\ \mathbf{a}_{2,t}$ are the updated instance component at epoch t denotes the current epoch.

– **Post-Challenge Query Phase**: The adversary $\mathcal{A}$ can make $\mathcal{O}_{\mathsf{upd}}^{\mathsf{uhqcPKE}}(\boldsymbol{\mu}_i)$ queries for $i = \{t, t+1, \ldots, t'-1\}$ which are simulated exactly in the same way by $\mathcal{D}$ as described in the pre-challenge query phase for $\boldsymbol{\mu}_i \in \mathbb{F}_2^k$.

– **Reveal Phase**: The distinguisher $\mathcal{D}$ samples $\boldsymbol{\mu}^* \xleftarrow{\$} \mathbb{F}_2^k$ at epoch $t'+1$, independent of $\boldsymbol{\mu}^*$, computes

$$\mathsf{sk}^* = (\boldsymbol{\rho}^*, \boldsymbol{\gamma}^*) \xleftarrow{\$} \mathcal{S}_w^n(\mathbb{F}_2) \times \mathcal{S}_w^n(\mathbb{F}_2)$$

$$\mathsf{pk}^* = (\mathbf{h}^* = \Pi_{\boldsymbol{\mu}^*}(\mathbf{h}_{t'}), \mathbf{s}^* = \boldsymbol{\rho}^* + \mathbf{h}^*\boldsymbol{\gamma}^*) \in \mathbb{F}_{2,b_1}^n \times \mathbb{F}_{2,b_2}^n,$$

$$\mathsf{up}^* = \mathsf{CT}_{\boldsymbol{\mu}^*} \longleftarrow \mathsf{uhqcPKE.Enc}(\mathsf{pp}_{\mathsf{uhqcPKE}}, \mathsf{pk}_{t'}^{(\mathsf{u})}, \boldsymbol{\mu}^*; \theta'')$$

where $\mathsf{CT}_{\boldsymbol{\mu}^*} = (\mathbf{u}_{\boldsymbol{\mu}^*} = \mathbf{r}_1'' + \mathbf{h}_{t'}\mathbf{r}_2'', \mathbf{v}_{\boldsymbol{\mu}^*} = \boldsymbol{\mu}^*\mathbf{G} + \mathbf{s}_{t'}\mathbf{r}_2'' + \mathbf{e}'')$ and forwards the tuple $(\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{up}^*)$ to $\mathcal{A}$. $(\Pi_{\boldsymbol{\mu}^*}, \mathsf{F}_{\boldsymbol{\mu}^*}) = \mathsf{Samp}(\boldsymbol{\mu}^*) \in \mathsf{Per}(\mathcal{R})$, $\mathsf{pk}_{t'}^{(\mathsf{u})} = (\mathbf{h}_{t'}, \mathbf{s}_{t'}) \in \mathbb{F}_{2,b_1}^n \times \mathbb{F}_{2,b_2}^n, \theta'' \in \mathbb{F}_2^\alpha$, $\mathsf{P}(\theta'') = (\mathbf{e}'', \mathbf{r}_1'', \mathbf{r}_2'') \in \mathcal{S}_{w_e}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2)$.

– **Guess Phase**: The adversary $\mathcal{A}$ submits a guess bit b' for the bit b chosen by $\mathcal{D}$ in the challenge phase, $\mathcal{D}$ forwards to $\mathcal{B}$, $\beta = 0$ if $b' \neq b$ and $\beta = 1$ if $b' = b$.

Observe that $\mathcal{D}$ perfectly simulates the experiment using the given instance of 3-$\mathsf{DQCSDP}(n, w_e, w_r, b_1, b_2)$. The public parameters $\mathsf{pp}_{\mathsf{uhqcPKE}}$, the initial public key $\mathsf{pk}_0^{(\mathsf{u})} \in \mathbb{F}_{2,b_1}^n \times \mathbb{F}_{2,b_2}^n$ in the setup phase, the update ciphertext $\mathsf{up}_{i+1}^{(\mathsf{u})} \in \mathcal{R} \times \mathcal{R}$, updated public key $\mathsf{pk}_{i+1}^{(\mathsf{u})} \in \mathbb{F}_{2,b_1}^n \times \mathbb{F}_{2,b_2}^n$ in the pre-challenge and post-challenge query phase, the challenge ciphertext $\mathsf{CT}_{\mathbf{m}_b^*} \in \mathcal{R} \times \mathcal{R}$ in the challenge phase and the simulation of $(\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{up}^*)$ in the reveal phase are identically distributed in both $\mathcal{G}_3$ and $\mathcal{G}_4$. Thereby, adversary's view in both game are same and the following two cases hold.

Case $\beta = 0$: If $\mathsf{Ins} = (\mathbf{H}, \mathbf{a})$ for $\mathbf{H} = \begin{pmatrix} \mathbf{I}_n & \mathbf{0} & \mathsf{rot}(\mathbf{h}) \\ \mathbf{0} & \mathbf{I}_n & \mathsf{rot}(\mathbf{s}) \end{pmatrix}$, $\mathbf{a} = (\mathbf{a}_1 = \tilde{\mathbf{r}}_1 + \mathbf{h}\tilde{\mathbf{r}}_2, \mathbf{a}_2 = \mathbf{s}\tilde{\mathbf{r}}_2 + \tilde{\mathbf{e}}) \in \mathbb{F}_{2,b_3}^n \times \mathbb{F}_{2,b_3'}^n$, $\tilde{\mathbf{r}}_1, \tilde{\mathbf{r}}_2 \xleftarrow{\$} \mathbb{F}_{2,w_r}^n$, $\tilde{\mathbf{e}} \xleftarrow{\$} \mathbb{F}_{2,w_e}^n$ then the simulation follows as in game $\mathcal{G}_3$ and therefore,

$$\Pr\Big[\mathcal{D}(\mathbf{H}, \mathbf{a}) = 1 \;\Big|\; \mathbf{H} = \begin{pmatrix} \mathbf{I}_n & \mathbf{0} & \mathsf{rot}(\mathbf{h}) \\ \mathbf{0} & \mathbf{I}_n & \mathsf{rot}(\mathbf{s}) \end{pmatrix},$$
$$\mathbf{a} = \big(\mathbf{a}_1 = \tilde{\mathbf{r}}_1 + \mathbf{h}\tilde{\mathbf{r}}_2,\ \mathbf{a}_2 = \mathbf{s}\tilde{\mathbf{r}}_2 + \tilde{\mathbf{e}}\big) \in \mathbb{F}_{2,b_3}^n \times \mathbb{F}_{2,b_3'}^n, \tilde{\mathbf{r}}_1, \tilde{\mathbf{r}}_2 \xleftarrow{\$} \mathbb{F}_{2,w_r}^n, \tilde{\mathbf{e}} \xleftarrow{\$} \mathbb{F}_{2,w_e}^n\Big]$$
$$= \Pr[S_4].$$

Case $\beta = 1$**:** If $\mathsf{Ins} = (\mathbf{H}, \mathbf{a})$ for $\mathbf{H} = \begin{pmatrix} \mathbf{I}_n & \mathbf{0} & \mathsf{rot}(\mathbf{h}) \\ \mathbf{0} & \mathbf{I}_n & \mathsf{rot}(\mathbf{s}) \end{pmatrix}$, $\mathbf{a} = (\tilde{\mathbf{r}}_1 + \mathbf{h}\tilde{\mathbf{r}}_2, \mathbf{s}\tilde{\mathbf{r}}_2 + \tilde{\mathbf{e}})$, $\tilde{\mathbf{r}}_1, \tilde{\mathbf{r}}_2 \xleftarrow{\$} \mathcal{S}^n_{w_r}(\mathbb{F}_2)$, $\tilde{\mathbf{e}} \xleftarrow{\$} \mathcal{S}^n_{w_e}(\mathbb{F}_2)$ then the simulation follows as in game $\mathcal{G}_2$ and therefore

$$\Pr\Big[\mathcal{D}(\mathbf{H}, \mathbf{a}) = 1 \;\Big|\; \mathbf{H} = \begin{pmatrix} \mathbf{I}_n & \mathbf{0} & \mathsf{rot}(\mathbf{h}) \\ \mathbf{0} & \mathbf{I}_n & \mathsf{rot}(\mathbf{s}) \end{pmatrix},$$
$$\mathbf{a} = \big(\mathbf{a}_1 = \tilde{\mathbf{r}}_1 + \mathbf{h}\tilde{\mathbf{r}}_2,\ \mathbf{a}_2 = \mathbf{s}\tilde{\mathbf{r}}_2 + \tilde{\mathbf{e}}\big) \in \mathbb{F}^n_{2,b_3} \times \mathbb{F}^n_{2,b'_3}, \tilde{\mathbf{r}}_1, \tilde{\mathbf{r}}_2 \xleftarrow{\$} \mathcal{S}^n_{w_r}(\mathbb{F}_2), \tilde{\mathbf{e}} \xleftarrow{\$} \mathcal{S}^n_{w_e}(\mathbb{F}_2)\Big]$$
$$= \Pr[S_3].$$

Note that,

$$\begin{pmatrix} \mathbf{I}_n & \mathbf{0} & \mathsf{rot}(\mathbf{h}) \\ \mathbf{0} & \mathbf{I}_n & \mathsf{rot}(\mathbf{s}) \end{pmatrix} (\tilde{\mathbf{r}}_1, \tilde{\mathbf{e}}, \tilde{\mathbf{r}}_2)^T = (\tilde{\mathbf{r}}_1 + \mathbf{h}\tilde{\mathbf{r}}_2,\ \tilde{\mathbf{e}} + \mathbf{s}\tilde{\mathbf{r}}_2)^T = (\mathbf{a}_1, \mathbf{a}_2)^T.$$

Therefore if the instance $\mathsf{Ins} = (\mathbf{H}, \mathbf{a})$ is from the 3-$\mathsf{QCSDP}(n, w_e, w_r, b_1, b_2)$ distribution then $\tilde{\mathbf{r}}_1, \tilde{\mathbf{r}}_2 \xleftarrow{\$} \mathcal{S}^n_{w_r}(\mathbb{F}_2), \tilde{\mathbf{e}} \xleftarrow{\$} \mathcal{S}^n_{w_e}(\mathbb{F}_2)$

$$\begin{aligned}(\mathbf{u}_{\mathbf{m}^*_b}, \mathbf{v}_{\mathbf{m}^*_b} - \mathbf{m}^*_b\mathbf{G}) &= (\mathbf{a}_{1,t} = \Pi_{\boldsymbol{\mu}_{t-1}}(\mathbf{a}_{1,t-1}), \mathbf{a}_{2,t} = \mathsf{F}_{\boldsymbol{\mu_{t-1}}}(\mathbf{a}_{2,t-1})) \\ &= (\tilde{\mathbf{r}}'_1 + \mathbf{h}_t\tilde{\mathbf{r}}'_2,\ \tilde{\mathbf{e}}' + \mathbf{s}_t\tilde{\mathbf{r}}'_2)\end{aligned}$$

for some $\tilde{\mathbf{r}}'_1, \tilde{\mathbf{r}}'_2 \in \mathcal{S}^n_{w_r}(\mathbb{F}_2)$, $\tilde{\mathbf{e}}' \in \mathcal{S}^n_{w_e}(\mathbb{F}_2)$ by the Proposition 3 and Theorem 5 where $(\Pi_{\boldsymbol{\mu}_{t-1}}, \mathsf{F}_{\boldsymbol{\mu}_{t-1}}) = \mathsf{Samp}(\boldsymbol{\mu}_{t-1}) \in \mathsf{Per}(\mathcal{R})$, $\mathbf{a}_{1,t-1}, \mathbf{a}_{2,t-1}$ are the updated instance components at epoch $t-1$ formed using $\mathbf{a}_{1,0} = \mathbf{a}_1$ and $\mathbf{a}_{2,0} = \mathbf{a}_2$. Then

$$\mathsf{CT}_{\mathbf{m}^*_b} = (\mathbf{u}_{\mathbf{m}^*_b} = \tilde{\mathbf{r}}'_1 + \mathbf{h}_t\tilde{\mathbf{r}}'_2,\ \mathbf{v}_{\mathbf{m}^*_b} = \mathbf{m}^*_b\mathbf{G} + \mathbf{s}_t\tilde{\mathbf{r}}'_2 + \tilde{\mathbf{e}}').$$

follows the same distribution as in $\mathcal{G}_3$. On the other hand if the instance $\mathsf{Ins} = (\mathbf{H}, \mathbf{a})$ is from the uniform distribution over $\mathbb{F}^{2n\times 3n}_{2,b_1,b_2} \times \left(\mathbb{F}^n_{2,b_3} \times \mathbb{F}^n_{2,b'_3}\right)$ then $\mathbf{a}_1 = \tilde{\mathbf{r}}_1 + \mathbf{h}\tilde{\mathbf{r}}_2,\ \mathbf{a}_2 = \mathbf{s}\tilde{\mathbf{r}}_2 + \tilde{\mathbf{e}}$ for some $\tilde{\mathbf{r}}_1, \tilde{\mathbf{r}}_2 \xleftarrow{\$} \mathbb{F}^n_{2,w_r}, \tilde{\mathbf{e}} \xleftarrow{\$} \mathbb{F}^n_{2,w_e}$ and therefore

$$\begin{aligned}(\mathbf{u}_{\mathbf{m}^*_b}, \mathbf{v}_{\mathbf{m}^*_b} - \mathbf{m}^*_b\mathbf{G}) &= (\mathbf{a}_{1,t} = \Pi_{\boldsymbol{\mu}_{t-1}}(\mathbf{a}_{1,t-1}), \mathbf{a}_{2,t} = \mathsf{F}_{\boldsymbol{\mu_{t-1}}}(\mathbf{a}_{2,t-1})) \\ &= (\tilde{\mathbf{r}}'_1 + \mathbf{h}_t\tilde{\mathbf{r}}'_2,\ \tilde{\mathbf{e}}' + \mathbf{s}_t\tilde{\mathbf{r}}'_2)\end{aligned}$$

for some $\tilde{\mathbf{r}}'_1, \tilde{\mathbf{r}}'_2 \in \mathbb{F}^n_{2,w_r}$, $\tilde{\mathbf{e}}' \in \mathbb{F}^n_{2,w_e}$ by Proposition 3, and Theorem 5 where $\mathbf{a}_{1,t-1},\ \mathbf{a}_{2,t-1}$ the updated instance components formed by extracting $\mathbf{a}_1 = \mathbf{a}_{1,0}, \mathbf{a}_2 = \mathbf{a}_{2,0}$ from the instance and $(\Pi_{\boldsymbol{\mu}_{t-1}}, \mathsf{F}_{\boldsymbol{\mu}_{t-1}}) = \mathsf{Samp}(\boldsymbol{\mu}_{t-1}) \in \mathsf{Per}(\mathcal{R})$. Then

$$\mathsf{CT}_{\mathbf{m}^*_b} = (\mathbf{a}_{1,t}, \mathbf{m}^*_b\mathbf{G} + \mathbf{a}_{2,t}) = (\mathbf{u}_{\mathbf{m}^*_b} = \tilde{\mathbf{r}}'_1 + \mathbf{h}_t\tilde{\mathbf{r}}'_2, \mathbf{v}_{\mathbf{m}^*_b} = \boldsymbol{\mu}_t\mathbf{G} + \mathbf{s}_t\tilde{\mathbf{r}}'_2 + \tilde{\mathbf{e}}')$$

follows the same distribution as in $\mathcal{G}_4$. Thus the probability of distinguishing game $\mathcal{G}_3$ and game $\mathcal{G}_4$ is same as the advantage of breaking the 3-$\mathsf{DQCSDP}(n, w_e, w_r, b_1, b_2)$ game. Hence, $|\Pr[S_3] - \Pr[S_4]| = \mathsf{Adv}^{3\text{-}\mathsf{DQCSDP}}_{\mathcal{D}}(1^\lambda) <$

ε_3 for some negligible function ε_3 as 3-DQCSDP problem is hard.

• *Proof of Claim 4:* We show that if there exists an adversary $\mathcal{A}$ capable of distinguishing the games $\mathcal{G}_4$ and $\mathcal{G}_5$ then a distinguisher $\mathcal{D}$ can be constructed for the 3-DQCSDP(n, w_e, w_r, b_1, b_2) problem. Let $\mathcal{B}$ be the 3-DQCSDP(n, w_e, w_r, b_1, b_2) (see Definition 15) challenger who provides an 3-DQCSDP(n, w_e, w_r, b_1, b_2) problem instance $\mathsf{Ins} = (\mathbf{H}, \mathbf{a} = (\mathbf{a}_1, \mathbf{a}_2)) \in \mathbb{F}_{2,b_1,b_2}^{2n\times 3n} \times \left(\mathbb{F}_{2,b_3}^n \times \mathbb{F}_{2,b_3'}^n\right)$ to the distinguisher $\mathcal{D}$ where

$$\mathbf{H} = \begin{pmatrix} \mathbf{I}_n & \mathbf{0} & \mathsf{rot}(\mathbf{h}) \\ \mathbf{0} & \mathbf{I}_n & \mathsf{rot}(\mathbf{s}) \end{pmatrix},\ \mathbf{h} \in \mathbb{F}_{2,b_1}^n,\ \mathbf{s} \in \mathbb{F}_{2,b_2}^n, b_1 = \mathbf{h}\cdot\mathbf{1} \bmod 2,$$
$$b_2 = w + b_1 w \bmod 2, b_3 = w_r + b_1 w_r \bmod 2,\ b_3' = w_e + b_2 w_r \bmod 2,$$
$$\mathbf{a} = \left(\mathbf{a}_1 = \tilde{\mathbf{r}}_1 + \mathbf{h}\tilde{\mathbf{r}}_2,\ \mathbf{a}_2 = \mathbf{s}\tilde{\mathbf{r}}_2 + \tilde{\mathbf{e}}\right) \in \mathbb{F}_{2,b_3}^n \times \mathbb{F}_{2,b_3'}^n,$$

and $(\tilde{\mathbf{r}}_1, \tilde{\mathbf{r}}_2, \tilde{\mathbf{e}})$ is either a random tuple from $\mathbb{F}_{2,w_r}^n \times \mathbb{F}_{2,w_r}^n \times \mathbb{F}_{2,w_e}^n$ or from $\mathcal{S}_{w_r}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2) \times \mathcal{S}_{w_e}^n(\mathbb{F}_2)$. The distinguisher $\mathcal{D}$ acts as a challenger to $\mathcal{A}$ and simulates its views as follows.

– **Setup**: The distinguisher $\mathcal{D}$ chooses a τ error correcting decodable $[n, k]$ linear code C with a generator matrix is $\mathbf{G}$. It also selects an encryption randomness size α, fixed weights w, w_e, w_r, a randomness generator $\mathsf{P} : \{0,1\}^* \longrightarrow \mathcal{S}_{w_e}^n(\mathbb{F}_2^n) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2^n) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2^n)$ along with a sampling function $\mathsf{Samp} : \mathbb{F}_2^k \longrightarrow \mathsf{Per}(\mathcal{R})$ and sets

$$\mathsf{pp}_{\mathsf{uhqcPKE}} = (n, k, \alpha, \tau, w, w_e, w_r, \mathsf{Samp}, \mathbf{G}, \mathsf{P}).$$

It extracts $\mathbf{h}, \mathbf{s}$ from instance $(\mathbf{H}, \mathbf{a})$ and sets

$$\mathsf{pk}_0^{(\mathsf{u})} = (\mathbf{h}_0, \mathbf{s}_0) = (\mathbf{h}, \mathbf{s}) \in \mathbb{F}_{2,b_1}^n \times \mathbb{F}_{2,b_2}^n$$

and forwards $\mathsf{pp}_{\mathsf{uhqcPKE}}$, $\mathsf{pk}_0^{(\mathsf{u})}$ to its adversary $\mathcal{A}$. The distinguisher sets $\mathbf{a}_1 = \mathbf{a}_{1,0}$ and $\mathbf{a}_2 = \mathbf{a}_{2,0}$ where $\mathbf{a} = (\mathbf{a}_1, \mathbf{a}_2) \in \mathbb{F}_{2,b_3}^n \times \mathbb{F}_{2,b_3'}^n$ extracted from the instance $(\mathbf{H}, \mathbf{a})$. $\mathcal{D}$ initializes a list $\mathbf{a}_\mathsf{List} = \mathbf{a}_\mathsf{List} \cup \{\mathbf{a}_{1,0}, \mathbf{a}_{2,0}\}$.

– **Pre-Challenge Query Phase**: On receiving a query $\boldsymbol{\mu}_i \in \mathbb{F}_2^k$ for $i = \{0, 1, 2, \ldots, t-1\}$ from $\mathcal{A}$ the distinguisher $\mathcal{D}$ updates the list $\mathbf{a}_\mathsf{List} = \mathbf{a}_\mathsf{List} \cup \{\mathbf{a}_{1,i}, \mathbf{a}_{2,i}\}$ and computes

$$\mathsf{up}_{i+1}^{(\mathsf{u})} = (\mathbf{r}_1' + \mathbf{h}_i\mathbf{r}_2', \boldsymbol{\mu}_i\mathbf{G} + \mathbf{s}_i\mathbf{r}_2' + \mathbf{e}'),$$

$$\mathsf{pk}_{i+1}^{(\mathsf{u})} = (\Pi_{\boldsymbol{\mu}_i}(\mathbf{h}_i), \mathsf{F}_{\boldsymbol{\mu}_i}(\mathbf{s}_i)) = (\mathbf{h}_{i+1}, \mathbf{s}_{i+1}) \in \mathbb{F}_{2,b_1}^n \times \mathbb{F}_{2,b_2}^n$$
$$\mathbf{a}_{1,i+1} = \Pi_{\boldsymbol{\mu}_i}(\mathbf{a}_{1,i}),\ \mathbf{a}_{2,i+1} = \mathsf{F}_{\boldsymbol{\mu}_i}(\mathbf{a}_{2,i})$$

and forwards $\mathsf{up}_{i+1}^{(\mathsf{u})}$, $\mathsf{pk}_{i+1}^{(\mathsf{u})}$ to $\mathcal{A}$ where $\mathbf{r}_1', \mathbf{r}_2' \xleftarrow{\$} \mathbb{F}_{2,w_r}^n, \mathbf{e}' \xleftarrow{\$} \mathbb{F}_{2,w_e}^n, (\Pi_{\boldsymbol{\mu}_i}, \mathsf{F}_{\boldsymbol{\mu}_i}) = \mathsf{Samp}(\boldsymbol{\mu}_i) \in \mathsf{Per}(\mathcal{R})$, $\mathsf{pk}_i^{(\mathsf{u})} = (\mathbf{h}_i, \mathbf{s}_i) \in \mathbb{F}_{2,b_1}^n \times \mathbb{F}_{2,b_2}^n$, $\mathbf{a}_{1,i}, \mathbf{a}_{2,i}$ are the updated

instance component at epoch i.

– **Challenge Phase**: On receiving challenge message pair $(\mathbf{m}_0^*, \mathbf{m}_1^*) \in \mathbb{F}_2^k \times \mathbb{F}_2^k$ from $\mathcal{A}$, its distinguisher $\mathcal{D}$ samples a bit $b \xleftarrow{\$} \{0,1\}$, computes

$$\mathsf{CT}_{\mathbf{m}_b^*} = (\mathbf{r}_1 + \mathbf{h}_t\mathbf{r}_2, \mathbf{m}_b^*\mathbf{G} + \mathbf{s}_t\mathbf{r}_2 + \mathbf{e}) \leftarrow \mathsf{uhqcPKE.Enc}(\mathsf{pp}_{\mathsf{uhqcPKE}}, \mathsf{pk}_t^{(\mathsf{u})}, \mathbf{m}_b^*; \theta),$$

and forwards it to $\mathcal{A}$ where $\theta \in \mathbb{F}_2^\alpha$ is a random α-bit string, $\mathsf{P}(\theta) = (\mathbf{e}, \mathbf{r}_1, \mathbf{r}_2) \in \mathcal{S}_{w_e}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2)$ and $\mathsf{pk}_t^{(\mathsf{u})} = (\mathbf{h}_t, \mathbf{s}_t) \in \mathbb{F}_{2,b_1}^n \times \mathbb{F}_{2,b_2}^n$ is the public key at epoch t.

– **Post-Challenge Query Phase**: The adversary $\mathcal{A}$ can make $\mathcal{O}_{\mathsf{upd}}^{\mathsf{uhqcPKE}}(\boldsymbol{\mu}_i)$ queries for $i = \{t, t+1, \cdots, t'-1\}$ which are simulated in the same manner by $\mathcal{D}$ as described in the pre-challenge query phase for $\boldsymbol{\mu}_i \in \mathbb{F}_2^k$.

– **Reveal Phase**: The distinguisher $\mathcal{D}$ samples $\boldsymbol{\mu}^* \xleftarrow{\$} \mathbb{F}_2^k$, $(\boldsymbol{\rho}^*, \boldsymbol{\gamma}^*) \xleftarrow{\$} \mathcal{S}_w^n(\mathbb{F}_2) \times \mathcal{S}_w^n(\mathbb{F}_2)$ at epoch $t'+1$ and sets

$$\mathsf{sk}^* = (\boldsymbol{\rho}^*, \boldsymbol{\gamma}^*)$$

$$\mathsf{pk}^* = (\mathbf{h}^* = \Pi_{\boldsymbol{\mu}^*}(\mathbf{h}_{t'}), \mathbf{s}^* = \boldsymbol{\rho}^* + \mathbf{h}^*\boldsymbol{\gamma}^*) \in \mathbb{F}_{2,b_1}^n \times \mathbb{F}_{2,b_2}^n$$

$$\boxed{\mathsf{up}^* = (\mathbf{u}_{\boldsymbol{\mu}^*} = \mathbf{a}_{1,t'}, \mathbf{v}_{\boldsymbol{\mu}^*} = \boldsymbol{\mu}^*\mathbf{G} + \mathbf{a}_{2,t'}) \in \mathcal{R} \times \mathcal{R},}$$

and forwards the tuple $(\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{up}^*)$ to the adversary $\mathcal{A}$. $(\Pi_{\boldsymbol{\mu}^*}, \mathsf{F}_{\boldsymbol{\mu}^*}) = \mathsf{Samp}(\boldsymbol{\mu}^*) \in \mathsf{Per}(\mathcal{R})$, $\mathsf{pk}_{t'}^{(\mathsf{u})} = (\mathbf{h}_{t'}, \mathbf{s}_{t'}) \in \mathcal{R} \times \mathcal{R}$, $\theta'' \xleftarrow{\$} \mathbb{F}_2^\alpha$, $\mathsf{P}(\theta'') = (\mathbf{e}'', \mathbf{r}_1'', \mathbf{r}_2'') \in \mathcal{S}_{w_e}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2) \times \mathcal{S}_{w_r}^n(\mathbb{F}_2)$ where $\mathbf{a}_{1,t}$, $\mathbf{a}_{2,t}$ are the updated instance component at epoch t denotes the current epoch.

– **Guess Phase**: The adversary $\mathcal{A}$ submits a guess bit b' for the bit b chosen by $\mathcal{D}$ in the challenge phase, $\mathcal{D}$ forwards to $\mathcal{B}$, $\beta = 0$ if $b' \neq b$ and $\beta = 1$ if $b' = b$.

Note that $\mathcal{D}$ faithfully simulates the experiment using the given 3-$\mathsf{DQCSDP}(n, w_e, w_r, b_1, b_2)$ instance. In the setup phase, the generated public parameters $\mathsf{pp}_{\mathsf{uhqcPKE}}$ and the initial public key $\mathsf{pk}_0^{(\mathsf{u})} = (\mathbf{h}_0, \mathbf{s}_0) \in \mathbb{F}_{2,b_1}^n \times \mathbb{F}_{2,b_2}^n$ each have identical distribution in game $\mathcal{G}_4$ and game $\mathcal{G}_5$. During the pre-challenge query phase and post-challenge query phase, the update ciphertext $\mathsf{up}_{i+1}^{(\mathsf{u})} \in \mathcal{R} \times \mathcal{R}$ and updated public key $\mathsf{pk}_{i+1}^{(\mathsf{u})} = (\mathbf{h}_{i+1}, \mathbf{s}_{i+1}) \in \mathbb{F}_{2,b_1}^n \times \mathbb{F}_{2,b_2}^n$ are simulated following to the same distribution in both the games. Likewise, the challenge ciphertext $\mathsf{CT}_{\mathbf{m}_b^*} \in \mathcal{R} \times \mathcal{R}$ in the challenge phase as well as the simulations of $(\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{up}^*)$ reveal phases are identically distributed in both game $\mathcal{G}_4$ and game $\mathcal{G}_5$. Therefore, the adversary's view the distributions are same in both games which yields the following two cases.

Case $\beta = 0$**:** If $\mathsf{Ins} = (\mathbf{H}, \mathbf{a})$ for $\mathbf{H} = \begin{pmatrix} \mathbf{I}_n & \mathbf{0} & \mathsf{rot}(\mathbf{h}) \\ \mathbf{0} & \mathbf{I}_n & \mathsf{rot}(\mathbf{s}) \end{pmatrix}$, $\mathbf{a} = (\mathbf{a}_1 =$

$\tilde{\mathbf{r}}_1 + \mathbf{h}\tilde{\mathbf{r}}_2, \mathbf{a}_2 = \mathbf{s}\tilde{\mathbf{r}}_2 + \tilde{\mathbf{e}}) \in \mathbb{F}_{2,b_3}^n \times \mathbb{F}_{2,b_3'}^n,\ \tilde{\mathbf{r}}_1, \tilde{\mathbf{r}}_2 \xleftarrow{\$} \mathbb{F}_{2,w_r}^n,\ \tilde{\mathbf{e}} \xleftarrow{\$} \mathbb{F}_{2,w_e}^n$ then the simulation follows as in game $\mathcal{G}_5$ and therefore,

$$\Pr\Big[\mathcal{D}(\mathbf{H}, \mathbf{a}) = 1 \ \Big|\ \mathbf{H} = \begin{pmatrix} \mathbf{I}_n & \mathbf{0} & \mathsf{rot}(\mathbf{h}) \\ \mathbf{0} & \mathbf{I}_n & \mathsf{rot}(\mathbf{s}) \end{pmatrix},$$
$$\mathbf{a} = \big(\mathbf{a}_1 = \tilde{\mathbf{r}}_1 + \mathbf{h}\tilde{\mathbf{r}}_2,\ \mathbf{a}_2 = \mathbf{s}\tilde{\mathbf{r}}_2 + \tilde{\mathbf{e}}\big) \in \mathbb{F}_{2,b_3}^n \times \mathbb{F}_{2,b_3'}^n, \tilde{\mathbf{r}}_1, \tilde{\mathbf{r}}_2 \xleftarrow{\$} \mathbb{F}_{2,w_r}^n, \tilde{\mathbf{e}} \xleftarrow{\$} \mathbb{F}_{2,w_e}^n\Big]$$
$$= \Pr[S_5].$$

Case $\beta = 1$: If $\mathsf{Ins} = (\mathbf{H}, \mathbf{a})$ for $\mathbf{H} = \begin{pmatrix} \mathbf{I}_n & \mathbf{0} & \mathsf{rot}(\mathbf{h}) \\ \mathbf{0} & \mathbf{I}_n & \mathsf{rot}(\mathbf{s}) \end{pmatrix}$, $\mathbf{a} = (\tilde{\mathbf{r}}_1 + \mathbf{h}\tilde{\mathbf{r}}_2, \mathbf{s}\tilde{\mathbf{r}}_2 + \tilde{\mathbf{e}})$, $\tilde{\mathbf{r}}_1, \tilde{\mathbf{r}}_2 \xleftarrow{\$} \mathcal{S}_{w_r}^n(\mathbb{F}_2)$, $\tilde{\mathbf{e}} \xleftarrow{\$} \mathcal{S}_{w_e}^n(\mathbb{F}_2)$ then the simulation follows as in game $\mathcal{G}_4$ and therefore

$$\Pr\Big[\mathcal{D}(\mathbf{H}, \mathbf{a}) = 1 \ \Big|\ \mathbf{H} = \begin{pmatrix} \mathbf{I}_n & \mathbf{0} & \mathsf{rot}(\mathbf{h}) \\ \mathbf{0} & \mathbf{I}_n & \mathsf{rot}(\mathbf{s}) \end{pmatrix},$$
$$\mathbf{a} = \big(\mathbf{a}_1 = \tilde{\mathbf{r}}_1 + \mathbf{h}\tilde{\mathbf{r}}_2,\ \mathbf{a}_2 = \mathbf{s}\tilde{\mathbf{r}}_2 + \tilde{\mathbf{e}}\big) \in \mathbb{F}_{2,b_3}^n \times \mathbb{F}_{2,b_3'}^n, \tilde{\mathbf{r}}_1, \tilde{\mathbf{r}}_2 \xleftarrow{\$} \mathcal{S}_{w_r}^n(\mathbb{F}_2),$$
$$\tilde{\mathbf{e}} \xleftarrow{\$} \mathcal{S}_{w_e}^n(\mathbb{F}_2)\Big] = \Pr[S_4].$$

Observe that,

$$\begin{pmatrix} \mathbf{I}_n & \mathbf{0} & \mathsf{rot}(\mathbf{h}) \\ \mathbf{0} & \mathbf{I}_n & \mathsf{rot}(\mathbf{s}) \end{pmatrix} (\tilde{\mathbf{r}}_1, \tilde{\mathbf{e}}, \tilde{\mathbf{r}}_2)^T = (\tilde{\mathbf{r}}_1 + \mathbf{h}\tilde{\mathbf{r}}_2,\ \tilde{\mathbf{e}} + \mathbf{s}\tilde{\mathbf{r}}_2)^T = (\mathbf{a}_1, \mathbf{a}_2)^T.$$

Therefore if the instance $\mathsf{Ins} = (\mathbf{H}, \mathbf{a})$ is from the 3-$\mathsf{QCSDP}(n, w_e, w_r, b_1, b_2)$ distribution then $\tilde{\mathbf{r}}_1, \tilde{\mathbf{r}}_2 \xleftarrow{\$} \mathcal{S}_{w_r}^n(\mathbb{F}_2), \tilde{\mathbf{e}} \xleftarrow{\$} \mathcal{S}_{w_e}^n(\mathbb{F}_2)$

$$\begin{aligned}(\mathbf{u}_{\boldsymbol{\mu}^*}, \mathbf{v}_{\boldsymbol{\mu}^*} - \boldsymbol{\mu}^*\mathbf{G}) &= (\mathbf{a}_{1,t'} = \Pi_{\boldsymbol{\mu}_{t'-1}}(\mathbf{a}_{1,t'-1}), \mathbf{a}_{2,t'} = \mathsf{F}_{\boldsymbol{\mu}_{t'-1}}(\mathbf{a}_{2,t'-1})) \\ &= (\tilde{\mathbf{r}}_1' + \mathbf{h}_{t'}\tilde{\mathbf{r}}_2',\ \tilde{\mathbf{e}}' + \mathbf{s}_{t'}\tilde{\mathbf{r}}_2')\end{aligned}$$

for some $\tilde{\mathbf{r}}_1', \tilde{\mathbf{r}}_2' \in \mathcal{S}_{w_r}^n(\mathbb{F}_2), \tilde{\mathbf{e}}' \in \mathcal{S}_{w_e}^n(\mathbb{F}_2)$ using the Proposition 3 and Theorem 5 where $(\Pi_{\boldsymbol{\mu}^*}, \mathsf{F}_{\boldsymbol{\mu}^*}) = \mathsf{Samp}(\boldsymbol{\mu}^*) \in \mathsf{Per}(\mathcal{R})$, $\mathbf{a}_{1,t'-1}, \mathbf{a}_{2,t'-1}$ are the updated instance components at epoch $t'-1$ formed using $\mathbf{a}_{1,0} = \mathbf{a}_1$ and $\mathbf{a}_{2,0} = \mathbf{a}_2$. Therefore, the resulting ciphertext

$$\mathsf{CT}_{\boldsymbol{\mu}^*} = (\mathbf{u}_{\boldsymbol{\mu}^*} = \tilde{\mathbf{r}}_1' + \mathbf{h}_{t'}\tilde{\mathbf{r}}_2',\ \mathbf{v}_{\boldsymbol{\mu}^*} = \boldsymbol{\mu}^*\mathbf{G} + \mathbf{s}_{t'}\tilde{\mathbf{r}}_2' + \tilde{\mathbf{e}}').$$

follows the same distribution as in $\mathcal{G}_4$. On the other hand suppose that the instance $\mathsf{Ins} = (\mathbf{H}, \mathbf{a})$ is from the uniform distribution over $\mathbb{F}_{2,b_1,b_2}^{2n \times 3n} \times \Big(\mathbb{F}_{2,b_3}^n \times \mathbb{F}_{2,b_3'}^n\Big)$ then $\mathbf{a}_1 = \tilde{\mathbf{r}}_1 + \mathbf{h}\tilde{\mathbf{r}}_2,\ \mathbf{a}_2 = \mathbf{s}\tilde{\mathbf{r}}_2 + \tilde{\mathbf{e}}$ for some $\tilde{\mathbf{r}}_1, \tilde{\mathbf{r}}_2 \xleftarrow{\$} \mathbb{F}_{2,w_r}^n, \tilde{\mathbf{e}} \xleftarrow{\$} \mathbb{F}_{2,w_e}^n$ and therefore

$$\begin{aligned}(\mathbf{u}_{\boldsymbol{\mu}^*}, \mathbf{v}_{\boldsymbol{\mu}^*} - \boldsymbol{\mu}^*\mathbf{G}) &= (\mathbf{a}_{1,t'} = \Pi_{\boldsymbol{\mu}_{t'-1}}(\mathbf{a}_{1,t'-1}), \mathbf{a}_{2,t} = \mathsf{F}_{\boldsymbol{\mu}_{t-1}}(\mathbf{a}_{2,t'-1})) \\ &= (\tilde{\mathbf{r}}_1' + \mathbf{h}_{t'}\tilde{\mathbf{r}}_2',\ \tilde{\mathbf{e}}' + \mathbf{s}_{t'}\tilde{\mathbf{r}}_2')\end{aligned}$$

for some $\tilde{\mathbf{r}}'_1, \tilde{\mathbf{r}}'_2 \in \mathbb{F}^n_{2,w_r}$, $\tilde{\mathbf{e}}' \in \mathbb{F}^n_{2,w_e}$ using Proposition 3 and Theorem 5 where $\mathbf{a}_{1,t'-1}$, $\mathbf{a}_{2,t'-1}$ the updated instance components formed by extracting $\mathbf{a}_1 = \mathbf{a}_{1,0}, \mathbf{a}_2 = \mathbf{a}_{2,0}$ from the instance and $(\Pi_{\boldsymbol{\mu}_{t'-1}}, \mathsf{F}_{\boldsymbol{\mu}_{t'-1}}) = \mathsf{Samp}(\boldsymbol{\mu}_{t'-1}) \in \mathsf{Per}(\mathcal{R})$. Consequently, the ciphertext

$$\mathsf{CT}_{\boldsymbol{\mu}^*} = (\mathbf{a}_{1,t'}, \boldsymbol{\mu}^*\mathbf{G} + \mathbf{a}_{2,t'}) = (\mathbf{u}_{\boldsymbol{\mu}^*} = \tilde{\mathbf{r}}'_1 + \mathbf{h}_{t'}\tilde{\mathbf{r}}'_2, \mathbf{v}_{\boldsymbol{\mu}^*} = \boldsymbol{\mu}^*\mathbf{G} + \mathbf{s}_{t'}\tilde{\mathbf{r}}'_2 + \tilde{\mathbf{e}}')$$

follows the same distribution as in $\mathcal{G}_5$. Thus the probability of distinguishing game $\mathcal{G}_4$ and game $\mathcal{G}_5$ is same as the advantage of breaking the 3-$\mathsf{DQCSDP}(n, w_e, w_r, b_1, b_2)$ game.
Hence, $|\Pr[S_4] - \Pr[S_5]| = \mathsf{Adv}^{3\text{-}\mathsf{DQCSDP}}_{\mathcal{D}}(1^\lambda) < \varepsilon_3$ for some negligible function ε_3 considering the hardness of the 3-DQCSDP problem.

5 Algebraic Instantiations and Entropy Discussions of $\mathsf{Per}(\mathcal{R})$

In Section 3.1, we mentioned that any concrete instantiation of the sampling algorithm Samp must satisfy some structural properties (homomorphism and weight preservation) as well as a distributional equivalence (high entropy). In this section, we present a construction of $\mathsf{Per}(\mathcal{R})$ using cyclic shifts and ring automorphisms. We furnish that the proposed construction satisfies the required algebraic properties and then discuss its entropy.

We first define two fundamental maps over the ring $\mathcal{R} = \mathbb{F}_2[x]/(x^n - 1)$.

The Shift Map $\boldsymbol{\Sigma^{(\delta_j)}}$: Let $\Sigma^{(\delta_j)} : \mathcal{R} \longrightarrow \mathcal{R}$ be defined by

$$\Sigma^{(\delta_j)}(\boldsymbol{\gamma}(x)) = \boldsymbol{\gamma}(x)x^{\delta_j} \pmod{x^n - 1}$$

for some integer shift $\delta_j \in M = \{0, 1, \ldots, n-1\}$ and any $\boldsymbol{\gamma}(x) \in \mathcal{R}$.

Fig. 5: Definition of the cyclic shift map $\Sigma^{(\delta_j)}$

Proposition 2. *Let $\Sigma^{(\delta_j)} : \mathcal{R} \to \mathcal{R}$ be the map presented in Figure 5. Then, for every $\mathbf{h}(x), \mathbf{y}(x) \in \mathcal{R}$:*

1. ***Additivity:*** $\Sigma^{(\delta_j)}(\mathbf{h} + \mathbf{y}) = \Sigma^{(\delta_j)}(\mathbf{h}) + \Sigma^{(\delta_j)}(\mathbf{y})$.
2. ***Vector Multiplication:*** $\Sigma^{(\delta_j)}(\mathbf{hy}) = \mathbf{h}\Sigma^{(\delta_j)}(\mathbf{y})$.
3. ***Weight Preservation:*** $\mathsf{wt}(\mathbf{h}) = \mathsf{wt}(\Sigma^{(\delta_j)}(\mathbf{h}))$.

Proof. By the definition given in Figure 5, $\Sigma^{(\delta_j)}(\mathbf{h}(x)+\mathbf{y}(x)) = (\mathbf{h}(x)+\mathbf{y}(x))x^{\delta_j} \pmod{x^n - 1}$. Since multiplication distributes over addition in $\mathcal{R}$, this evaluates to $\mathbf{h}(x)x^{\delta_j} + \mathbf{y}(x)x^{\delta_j} = \Sigma^{(\delta_j)}(\mathbf{h}(x)) + \Sigma^{(\delta_j)}(\mathbf{y}(x))$, proving additivity.

Similarly, due to the associativity of multiplication in $\mathcal{R}$, $\Sigma^{(\delta_j)}(\mathbf{h}(x)\mathbf{y}(x)) = \mathbf{h}(x)\big(\mathbf{y}(x)x^{\delta_j}\big) \pmod{x^n-1} = \mathbf{h}(x)\Sigma^{(\delta_j)}(\mathbf{y}(x))$, proving the vector multiplication property.

Finally, let $\mathbf{h}(x) = \sum_{i=0}^{n-1} h_i x^i$. The operation $\mathbf{h}(x)x^{\delta_j} \bmod (x^n-1)$ corresponds to taking the exponents modulo n. Thus, $\Sigma^{(\delta_j)}(\mathbf{h})$ is simply a cyclic shift of the coefficients of $\mathbf{h}$. Since a cyclic shift preserves the number of nonzero coefficients, $\mathsf{wt}(\Sigma^{(\delta_j)}(\mathbf{h})) = \mathsf{wt}(\mathbf{h})$.

The Automorphism Map $\Pi^{(p_j)}$: Let $\Pi^{(p_j)} : \mathcal{R} \longrightarrow \mathcal{R}$ be defined by

$$\Pi^{(p_j)}(\boldsymbol{\gamma}(x)) = \boldsymbol{\gamma}(x^{p_j}) \bmod (x^n-1)$$

for an integer $p_j \in N$, where $N = \{p_j \mid p_j \leq n \text{ and } \mathsf{gcd}(p_j, n) = 1\}$.

Fig. 6: Definition of the ring automorphism $\Pi^{(p_j)}$

Proposition 3. *The map $\Pi^{(p_j)}$ presented in Figure 6 is a weight-preserving ring automorphism on $\mathcal{R}$. Specifically, for all $\mathbf{h}, \mathbf{y} \in \mathcal{R}$:*

1. $\Pi^{(p_j)}(\mathbf{h}+\mathbf{y}) = \Pi^{(p_j)}(\mathbf{h}) + \Pi^{(p_j)}(\mathbf{y})$.
2. $\Pi^{(p_j)}(\mathbf{hy}) = \Pi^{(p_j)}(\mathbf{h})\Pi^{(p_j)}(\mathbf{y})$.
3. $\mathsf{wt}(\Pi^{(p_j)}(\mathbf{h})) = \mathsf{wt}(\mathbf{h})$.

Proof. We evaluate

$$\Pi^{(p_j)}(\mathbf{h}+\mathbf{y})(x) = (\mathbf{h}+\mathbf{y})(x^{p_j}) = \mathbf{h}(x^{p_j}) + \mathbf{y}(x^{p_j}) = \Pi^{(p_j)}(\mathbf{h}(x)) + \Pi^{(p_j)}(\mathbf{y}(x))$$

proving additivity. Similarly,

$$\Pi^{(p_j)}(\mathbf{hy})(x) = (\mathbf{hy})(x^{p_j}) = \mathbf{h}(x^{p_j})\mathbf{y}(x^{p_j}) = \Pi^{(p_j)}(\mathbf{h}(x))\Pi^{(p_j)}(\mathbf{y}(x))$$

proving multiplicativity. Because $\mathsf{gcd}(n, p_j) = 1$, the map $x \mapsto x^{p_j}$ induces a bijection of the exponents modulo n, making $\Pi^{(p_j)}$ a ring automorphism. Because this bijection simply permutes the coefficients without changing their values, the number of non-zero coefficients is invariant, yielding $\mathsf{wt}(\Pi^{(p_j)}(\mathbf{h})) = \mathsf{wt}(\mathbf{h})$.

Using the primitive maps Σ and Π, we construct a candidate set $\mathsf{Per}(\mathcal{R})$ through function composition.

Theorem 5. *For any $(\Pi, \mathsf{F}) \in \mathsf{Per}(\mathcal{R})$ defined in Figure 7 and any $\mathbf{x}, \mathbf{h}, \mathbf{y} \in \mathcal{R}$, it holds that:*

$$\mathsf{F}(\mathbf{x} + \mathbf{hy}) = \mathsf{F}(\mathbf{x}) + \Pi(\mathbf{h})\mathsf{F}(\mathbf{y}).$$

Furthermore, both Π and F strictly preserve Hamming weight.

Let $\mathsf{Per}(\mathcal{R}) = \{(\Pi, \mathsf{F})\}$ where:

$$\Pi = \Pi^{(p_{k'})} \circ \Pi^{(p_{k'-1})} \circ \cdots \circ \Pi^{(p_1)}$$

$$\mathsf{F} = (\Pi^{(p_{k'})} \circ \Sigma^{(\delta_{k'})}) \circ (\Pi^{(p_{k'-1})} \circ \Sigma^{(\delta_{k'-1})}) \circ \cdots \circ (\Pi^{(p_1)} \circ \Sigma^{(\delta_1)})$$

for uniformly sampled $\delta_j \in M$ and $p_j \in N$.

Fig. 7: $\mathsf{Per}(\mathcal{R})$

Proof. Using Proposition 2 and Proposition 3, the additivity of both primitive functions yields $\mathsf{F}(\mathbf{x} + \mathbf{hy}) = \mathsf{F}(\mathbf{x}) + \mathsf{F}(\mathbf{hy})$. We now evaluate $\mathsf{F}(\mathbf{hy})$.

By repeated application of the scalar property from Proposition 2 and the automorphism property from Proposition 3, the factor $\mathbf{h}$ propagates through the composition. Each Σ operation leaves $\mathbf{h}$ intact, while each Π operation transforms it. Thus:

$$\mathsf{F}(\mathbf{hy}) = (\Pi^{(p_{k'})} \circ \cdots \circ \Pi^{(p_1)})(\mathbf{h}) \cdot \mathsf{F}(\mathbf{y}) = \Pi(\mathbf{h})\mathsf{F}(\mathbf{y}).$$

Finally, since every individual Σ and Π map preserves Hamming weight, any composition of these maps, including Π and F, must also preserve Hamming weight.

Remark 1. (Discussion on Entropy). While the construction in Figure 7 satisfies the required algebraic and weight-preserving properties, it can not achieve distributional equivalence due to low entropy. The operations Σ and Π generate a space of size at most $|\mathsf{Per}(\mathcal{R})| \leq n\phi(n)$, since there are only n cyclic shifts and $\phi(n)$ automorphisms, and composition does not expand this set. For cryptographic parameters ($n \approx 10^4 - 10^5$), this is negligible compared to the $\binom{n}{w}$ possibilities in $\mathcal{S}_w^n(\mathbb{F}_2)$. We leave the construction of $\mathsf{Per}(\mathcal{R})$ satisfying the above conditions along with sufficiently high entropy for forward security as an open question.

6 Conclusion

This article presents a design for an IND-CR-CPA secure updatable public key encryption scheme in code-based setting. In terms of storage, our work performs well compared to some other existing protocols. Besides, our protocol accommodates both asynchronous and unbounded key updates. However, achieving stronger security requires allowing the adversary access to a decryption oracle associated with the secret key of the current epoch. The formal definition of this enhanced security guarantee is captured by the indistinguishability under chosen-randomness and chosen-ciphertext attack (IND-CR-CCA) security notion. As future work, we plan to explore the updatable key encapsulation mechanism that attains IND-CR-CCA security.

References

1. Abou Haidar, C., Libert, B., Passelègue, A.: Updatable public key encryption from dcr: Efficient constructions with stronger security. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. p. 11–22. CCS '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3548606.3559376, https://doi.org/10.1145/3548606.3559376
2. Albrecht, M.R., Benčina, B., Lai, R.W.F.: Hollow LWE: A new spin - unbounded updatable encryption from LWE and PCE. In: Fehr, S., Fouque, P. (eds.) Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques Madrid, Spain, May 4-8, 2025, Proceedings, Part VIII. Lecture Notes in Computer Science, vol. 15608, pp. 363–392. Springer (2025). https://doi.org/10.1007/978-3-031-91101-9_13, https://doi.org/10.1007/978-3-031-91101-9_13
3. Alwen, J., Coretti, S., Dodis, Y., Tselekounis, Y.: Security analysis and improvements for the IETF MLS standard for group messaging. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12170, pp. 248–277. Springer (2020). https://doi.org/10.1007/978-3-030-56784-2_9, https://doi.org/10.1007/978-3-030-56784-2_9
4. Alwen, J., Fuchsbauer, G., Mularczyk, M.: Updatable public-key encryption, revisited. In: Joye, M., Leander, G. (eds.) Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part VI. Lecture Notes in Computer Science, vol. 14656, pp. 346–376. Springer (2024). https://doi.org/10.1007/978-3-031-58754-2_13, https://doi.org/10.1007/978-3-031-58754-2_13
5. Asano, K., Watanabe, Y.: Updatable public key encryption with strong CCA security: Security analysis and efficient generic construction. In: Patra, A. (ed.) Topics in Cryptology - CT-RSA 2025 - Cryptographers' Track at the RSA Conference 2025, San Francisco, CA, USA, April 28-May 1, 2025, Proceedings. Lecture Notes in Computer Science, vol. 15598, pp. 223–246. Springer (2025). https://doi.org/10.1007/978-3-031-88661-4_10, https://doi.org/10.1007/978-3-031-88661-4_10
6. 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). https://doi.org/10.1109/TIT.1978.1055873
7. Berlekamp, E.R., McEliece, R.J., van Tilborg, H.C.A.: On the inherent intractability of certain coding problems. IEEE Transactions on Information Theory **24**(3), 384–386 (1978)
8. Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. In: Cachin, C., Camenisch, J. (eds.) Advances in Cryptology – EUROCRYPT 2004. Lecture Notes in Computer Science, vol. 3027, pp. 207–222. Springer, Heidelberg, Germany (2004)
9. Castryck, W., Decru, T.: An efficient key recovery attack on SIDH. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V. Lecture Notes in Computer

Science, vol. 14008, pp. 423–447. Springer (2023). https://doi.org/10.1007/978-3-031-30589-4_15, https://doi.org/10.1007/978-3-031-30589-4_15
10. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory **22**(6), 644–654 (1976)
11. Dodis, Y., Karthikeyan, H., Wichs, D.: Updatable public key encryption in the standard model. In: Theory of Cryptography Conference. pp. 254–285. Springer (2021)
12. Duparc, M., Fouotsa, T.B., Vaudenay, S.: SILBE: an updatable public key encryption scheme from lollipop attacks. In: Eichlseder, M., Gambs, S. (eds.) Selected Areas in Cryptography - SAC 2024 - 31st International Conference, Montreal, QC, Canada, August 28-30, 2024, Revised Selected Papers, Part I. Lecture Notes in Computer Science, vol. 15516, pp. 151–177. Springer (2024). https://doi.org/10.1007/978-3-031-82852-2_7, https://doi.org/10.1007/978-3-031-82852-2_7
13. Eaton, E., Jao, D., Komlo, C., Mokrani, Y.: Towards post-quantum key-updatable public-key encryption via supersingular isogenies. In: AlTawy, R., Hülsing, A. (eds.) Selected Areas in Cryptography - 28th International Conference, SAC 2021, Virtual Event, September 29 - October 1, 2021, Revised Selected Papers. Lecture Notes in Computer Science, vol. 13203, pp. 461–482. Springer (2021). https://doi.org/10.1007/978-3-030-99277-4_22, https://doi.org/10.1007/978-3-030-99277-4_22
14. Finiasz, M., Sendrier, N.: Distinguishing mceliece public keys. In: Advances in Cryptology - ASIACRYPT 2009. pp. 25–39. Springer (2009)
15. Gaborit, P., Aguilar-Melchor, C., Aragon, N., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.C., Persichetti, E., Zémor, G., Bos, J., et al.: Hamming quasi-cyclic (hqc). NIST Post-Quantum Standardization, 4th Round; National Institute of Standards and Technology: Gaithersburg, MD, USA (2025)
16. Gentry, C., Silverberg, A.: Hierarchical id-based cryptography. In: Zheng, Y. (ed.) Advances in Cryptology – ASIACRYPT 2002. Lecture Notes in Computer Science, vol. 2501, pp. 548–566. Springer, Heidelberg, Germany (2002)
17. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences **28**(2), 270–299 (1984). https://doi.org/https://doi.org/10.1016/0022-0000(84)90070-9, https://www.sciencedirect.com/science/article/pii/0022000084900709
18. Gulliver, T.A., Bhargava, V.K.: Some best rate 1/p and rate (p-1)/p systematic quasi-cyclic codes. IEEE Transactions on Information Theory **37**(3), 552–555 (1991). https://doi.org/10.1109/18.79914
19. Haidar, C.A., Passelègue, A., Stehlé, D.: Efficient updatable public-key encryption from lattices. In: Guo, J., Steinfeld, R. (eds.) Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings, Part V. Lecture Notes in Computer Science, vol. 14442, pp. 342–373. Springer (2023). https://doi.org/10.1007/978-981-99-8733-7_11, https://doi.org/10.1007/978-981-99-8733-7_11
20. Horwitz, J., Lynn, B.: Toward hierarchical identity-based encryption. In: Knudsen, L.R. (ed.) Advances in Cryptology – EUROCRYPT 2002. Lecture Notes in Computer Science, vol. 2332, pp. 466–481. Springer, Heidelberg, Germany (2002)
21. Jost, D., Maurer, U., Mularczyk, M.: Efficient ratcheting: Almost-optimal guarantees for secure messaging. In: Ishai, Y., Rijmen, V. (eds.) Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I. Lecture Notes in Computer Science, vol. 11476,

pp. 159–188. Springer (2019). https://doi.org/10.1007/978-3-030-17653-2_6, https://doi.org/10.1007/978-3-030-17653-2_6
22. MacWilliams, F.J., Sloane, N.J.A.: The Theory of Error-Correcting Codes, North-Holland Mathematical Library, vol. 16. North-Holland, Amsterdam (1977)
23. Maino, L., Martindale, C., Panny, L., Pope, G., Wesolowski, B.: A direct key recovery attack on SIDH. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V. Lecture Notes in Computer Science, vol. 14008, pp. 448–471. Springer (2023). https://doi.org/10.1007/978-3-031-30589-4_16, https://doi.org/10.1007/978-3-031-30589-4_16
24. Melchor, C.A., Aragon, N., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.C., Gaborit, P., Persichetti, E., Zémor, G., Bourges, I.: Hamming quasi-cyclic (hqc). NIST PQC Round **2**(4), 13 (2018)
25. Misoczki, R., Tillich, J.P., Sendrier, N., Barreto, P.: Mdpc-mceliece: New mceliece variants from moderate density parity-check codes. pp. 2069–2073 (07 2013). https://doi.org/10.1109/ISIT.2013.6620590
26. Robert, D.: Breaking SIDH in polynomial time. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V. Lecture Notes in Computer Science, vol. 14008, pp. 472–503. Springer (2023). https://doi.org/10.1007/978-3-031-30589-4_17, https://doi.org/10.1007/978-3-031-30589-4_17
27. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: 35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, November 20-22, 1994. pp. 124–134. IEEE Computer Society (1994). https://doi.org/10.1109/SFCS.1994.365700, https://doi.org/10.1109/SFCS.1994.365700
28. Statista: Quantum computing growth (2020), https://www.statista.com/chart/17896/quantum-computing-developments/, accessed: Apr. 4, 2020