UPKE and UKEM Schemes from Supersingular Isogenies

Pratima Jana[1*] and Ratna Dutta[1]

[1*]Department of Mathematics, Indian Institute of Technology Kharagpur, Kharagpur, 721302, West Bengal, India.

*Corresponding author(s). E-mail(s): pratimajanahatiary@kgpian.iitkgp.ac.in;
Contributing authors: ratna@maths.iitkgp.ac.in;

Abstract

Forward-secure public key encryption (**FS-PKE**) is a key-evolving public-key paradigm that ensures the confidentiality of past encryptions even if the secret key is compromised at some later point in time. However, existing **FS-PKE** schemes are considerably complex and less efficient compared to standard public-key encryption. *Updatable public-key encryption* (**UPKE**), introduced by Jost et al. (Eurocrypt 2019), was designed to achieve forward security in secure group messaging while maintaining efficiency. However, existing **UPKE** constructions either lack post-quantum security or do not support an unbounded number of updates. We focus on isogeny-based cryptosystems due to their suitability for handling an unbounded number of updates in long-term secure messaging. Existing isogeny-based **UPKE** schemes lack strong security guarantees and formal security proofs. They do not support asynchronous key updates and require sender-receiver coordination.

In this work, we present two isogeny-based **UPKE** schemes. The first scheme **UhPKE** extends Moriya et al.'s hash-based public key encryption scheme **hPKE** to support key updates while the second scheme **USimS** is an updatable version of Fouotsa et al.'s public key encryption scheme *simplified sigamal* (**SimS**). The scheme **UhPKE** relies on the *commutative supersingular isogeny Diffie-Hellman* (**CSSIDH**) assumption and achieves *indistinguishability under chosen randomness and chosen plaintext attack* (**IND-CR-CPA**). The scheme **USimS** derives its security under the hardness of the **CSSIDH** problem and the *commutative supersingular isogeny knowledge of exponent* (**CSSIKoE**) problem. It is the *first* isogeny-based **UPKE** scheme that exhibits *indistinguishability under chosen randomness and chosen ciphertext attack* (**IND-CR-CCA**). The security of **UhPKE** and **USimS** is established by proving that their underlying schemes, **hPKE** and

SimS are *circular secure and leakage resilience* (**CS + LR**). We emphasized that our constructions support an unlimited number of key updates while retaining the efficiency of their underlying public key encryption schemes. Besides, proposed **UPKE**s enable asynchronous key updates, allowing senders to update the public key independently. More affirmatively, **UhPKE** and **USimS** offer improved storage, computation and communication efficiency compared to existing **UPKE** schemes.
Furthermore, we extend and refine the security notion of the *updatable key encapsulation mechanism* (**UKEM**) introduced by Haidar et al. (Asiacrypt 2023) from the bounded number of updates to the unbounded number of updates. We present the *first* post-quantum secure **UKEM** that does not rely on zero-knowledge proofs. More precisely, we introduce two **UKEM** schemes which are the *first* of their kind in the isogeny setting. Our first scheme $\mathbf{UKEM_1}$, is derived from our **UhPKE** and achieves **IND-CR-CPA** security. Our second construction, $\mathbf{UKEM_2}$, is based on our **USimS** scheme and achieves **IND-CR-CCA** security. We provide security for our **UKEM**s in our proposed enhanced security framework that supports an unbounded number of key updates. More positively, our **UKEM**'s not only support unlimited key updates but also enable independent encapsulation and decapsulation key updates without requiring sender-receiver synchronization similar to our **UPKE**'s. Both $\mathbf{UKEM_1}$ and $\mathbf{UKEM_2}$ exhibit compact storage and communication costs with minimal size ciphertexts while their computational efficiency differs in decapsulation and key updates where $\mathbf{UKEM_2}$ incurs an additional discrete logarithm computation in decapsulation phase, but potentially offering stronger **IND-CR-CCA** security in contrast to $\mathbf{UKEM_1}$ which is **IND-CR-CPA** secure.

Keywords: Updatable public key encryption, Updatable key encapsulation mechanism, Isogeny, Forward Security.

1 Introduction

Ensuring secure communication over an untrusted channel is a fundamental challenge in cryptography. In group messaging scenarios where multiple users engage in long-term conversations, a critical security goal is to protect past communications even if a participant's secret key is later compromised. Forward security addresses this challenge by ensuring that previous messages remain confidential despite future key exposure. In symmetric-key systems, forward security can be implemented using a pseudo-random generator. Given an initial seed s_0, key updates follow the transformation $G(s_{i-1}) = (s_i, k_i)$ where k_i is a session key for encryption and s_i is used for subsequent key derivation. While this approach is computationally efficient, it presents a significant limitation in dynamic groups where securely adding or removing users becomes difficult without complex key management overhead. An alternative is group key agreement protocols which allow participants to negotiate a shared secret key interactively. In practical scenarios, group members may frequently go offline, making such synchronous operations impractical for secure messaging applications.

While forward security is well understood in symmetric-key cryptography, extending it to the public-key setting is challenging and introduces significant complexity. The concept of *forward-secure public-key encryption* (FS-PKE) was first formalized by Canetti et al. [1] where the key generation process produces an initial key pair $(pk_0^{(\mathsf{u})}, sk_0^{(\mathsf{u})})$ for a user u. This implicitly defines evolving public key chains $pk_0^{(\mathsf{u})} \rightarrow pk_1^{(\mathsf{u})} \rightarrow pk_2^{(\mathsf{u})} \rightarrow \cdots$ and evolving secret key chains $sk_0^{(\mathsf{u})} \rightarrow sk_1^{(\mathsf{u})} \rightarrow sk_2^{(\mathsf{u})} \rightarrow \cdots$. These key updates are designed to enable multiple senders to encrypt messages for a receiver who independently updates their secret key over time. Canetti et al. [1] demonstrated that FS-PKE can be built using *hierarchical identity-based encryption* (HIBE) [2, 3]. This connection led to a surge in research in FS-PKE over the last two decades with numerous constructions based on diverse cryptographic assumptions [4–7]. Despite these advancements, existing FS-PKE schemes remain significantly more complex and less efficient than standard *public-key encryption* (PKE).

UPKE. The notion of *updatable public key encryption* (UPKE) was initially introduced by Jost et al. [8] in 2019, as a relaxation of FS-PKE. In addition to standard PKE functionality, UPKE schemes allow encryption and decryption keys to be asynchronously updated with fresh entropy, thereby healing the protocol by restoring security even after exposure of secret values. The main difference between UPKE and FS-PKE is that in the former, new updated keys can be derived at any point of time by any sender whereas in FS-PKE only the recipient can update the key. In UPKE, to update the public-secret key pair $(\mathsf{pk}_i^{(\mathsf{u})}, \mathsf{sk}_i^{(\mathsf{u})})$ of a target recipient at epoch i, any sender can compute and send a public update consisting of a pair $(\mathsf{pk}_{i+1}^{(\mathsf{u})}, \mathsf{up}_{i+1})$ where $\mathsf{pk}_{i+1}^{(\mathsf{u})}$ is the updated public key of the target recipient at epoch $i+1$ using a randomness $\boldsymbol{\rho}_i$ and up_{i+1} is updated ciphertext which is an encryption of the randomness $\boldsymbol{\rho}_i$ under the public key $\mathsf{pk}_i^{(\mathsf{u})}$ at epoch i. The targe recipient can recover the randomness $\boldsymbol{\rho}_i$ by decrypting up_{i+1} using its secret key $\mathsf{sk}_i^{(\mathsf{u})}$ at epoch i and can transform $\mathsf{sk}_i^{(\mathsf{u})}$ to the secret key $\mathsf{sk}_{i+1}^{(\mathsf{u})}$ at epoch $i+1$ corresponding to $\mathsf{pk}_{i+1}^{(\mathsf{u})}$.

The security of UPKE ensures that ciphertexts encrypted under a user's public key at any epoch t remain confidential even if an adversary later compromises the secret key for some $j > t$. This guarantee holds as long as at least one of the key updates between epochs t and j was performed by an honest user, meaning the update was generated using private randomness unknown to the attacker. This security property is formally defined by the notion of *indistinguishability under chosen randomness and chosen plaintext attack* (IND-CR-CPA) where the adversary can impose updates on the target user's public key while selecting the randomness used in the update mechanism. However, stronger security requires granting the adversary to have access to a decryption oracle corresponding to the secret key of the current epoch. The formal definition of this enhanced security property is captured by the *indistinguishability under chosen randomness and chosen ciphertext attack* (IND-CR-CCA) security notion.

The UPKE constructions are Jost et al. [8] based on the hashed ElGamal public-key encryption scheme and rely on the *computational Diffie–Hellman* (CDH) assumption in the *random oracle model* (ROM). Over time, security definitions for UPKE have evolved, leading to more rigorous formalizations. A major advancement was made by Dodis et al. [9] in 2021 who introduced IND-CR-CPA and IND-CR-CCA security

notions for UPKE as standard definitions. Removing the dependence on random oracles, they proposed two concrete UPKE schemes in the standard model. Their first scheme derives its security from the hardness of the *decisional Diffie–Hellman* (DDH) problem and the second one relies on the *learning with errors* (LWE) assumption. However, these constructions for UPKE are bitwise encryption. To further advance UPKE without random oracles, Haidar et al. [10] introduced in 2022 a construction based on the hardness of the *decisional composite residuosity* (DCR) problem. More recently, in 2023, Haidar et al. [11] proposed an UPKE scheme based on the LWE assumption by significantly enhancing efficiency compared to Dodis et al. [9], making it a stronger candidate for practical deployment. In 2023, Asano and Watanabe [12] came up with an UPKE construction that further expands the landscape of updatable encryption schemes. Although Albrecht et al.'s [13] in 2024 achieves unbounded updatable encryption from LWE and PCE, it relies on large parameters due to the Leftover Hash Lemma and complex reductions.

Recent advancements in quantum computing [14] have reinforced the urgency of developing quantum-resistant cryptographic alternatives. Lattice-based cryptosystems are promising, but face inherent challenges due to error accumulation with each additional operation, thereby imposing limits on key updates or requiring the use of complex compression techniques to manage error accumulation. Consequently, these approaches are not ideal for secure messaging where long-term communication requires cryptographic protocols that are capable of supporting an unlimited number of key updates. Unlike other post-quantum approaches, isogeny-based cryptography provides algebraic structures that align well with UPKE requirements and naturally support an indefinite number of updates, making them strong desirable candidates for secure messaging applications that may span months or even years.

Currently, there exist two works on UPKE within the isogeny setting [15, 16]. In 2021, Eaton et al. [15], proposed two constructions for UPKE - one is based on the *Supersingular Isogeny Diffie-Hellman* (SIDH) assumption and the other derives its security from the *Commutative Supersingular Isogeny Diffie-Hellman* (CSSIDH) assumption. However, none of the schemes support asynchronous key updates for encryption and decryption, thereby limiting their practical applicability. Additionally, SIDH is later found to be insecure [17–19] due to attacks exploiting the accessible images of torsion points. These attacks leverage Kani's Lemma [20] to extract the secret isogeny. More recently, Duparc et al. [16] introduced in 2024 an UPKE scheme leveraging both the Deuring Correspondence and Kani's Lemma. However, their scheme is *one-way under chosen randomness and chosen plaintext attacks* (OW-CR-CPA) secure and lacks a formal security proof. Notably, none of the existing isogeny-based UPKE schemes achieve IND-CR-CCA security which is critical for resisting stronger adversarial attacks.

UKEM. In an effort to develop practical forward-secure cryptographic constructions, Haidar et al. [11] introduced the *updatable key encapsulation mechanism* (UKEM), an extension of the standard *key encapsulation mechanism* (KEM) that incorporates key update capabilities. They gave a generic construction of IND-CR-CCA secure UKEM from IND-CR-CPA secure UPKE along with a LWE-based instantiation.

It allows asynchronous key updates, but the formulation of their UKEM ensures correctness and security only under a predefined bound on the number of updates. The security of UKEM ensures that an encapsulated key generated under a user's public key $\mathsf{pk}_t^{(\mathsf{u})}$ at epoch t remains secure even if an adversary later compromises the corresponding secret key $\mathsf{sk}_j^{(\mathsf{u})}$ for some $j > t$ provided that at least one update between epochs t and j is performed honestly using the private randomness unknown to the adversary.

1.1 Contributions

Ensuring efficient and scalable cryptographic operations over extended periods is crucial when frequent key updates are necessary. Despite significant advancements in UPKE, existing constructions remain inefficient or impractical for real-world deployment, particularly in the context of post-quantum security. Lattice-based UPKE schemes [9, 11] while offering strong security guarantees, but accumulate noise with each update that enables only bounded updates or requires expensive compression techniques. On the other hand, existing isogeny-based UPKE constructions either need sender-receiver coordination due to a lack of asynchronous key updates or fail to achieve IND-CR-CCA security or rely on insecure assumptions like SIDH. As isogeny-based cryptography provides compact keys and compact ciphertexts with quantum resistance, a provably secure, efficient isogeny-based UPKE scheme is highly desirable. Our work aims to design a viable candidate for long-term secure messaging in a post-quantum world and design computationally efficient asynchronous UPKE in the isogeny setting that supports unbounded key updates with IND-CR-CCA security.

We propose two isogeny-based UPKE schemes with these elegant features that enhance their practical applicability in real-world scenarios. Our first scheme UhPKE integrates key update functionality in the existing isogeny-based hashed public key encryption scheme hPKE [21] and achieves IND-CR-CPA security based on the CSSIDH assumption. Our second construction USimS is based on the public key encryption such *simplified sigamal* (SimS) [22] and proven to be IND-CR-CCA secure under the CSSIDH and *commutative supersingular isogeny knowledge of exponent* (CSSIKoE) assumptions. Both UhPKE and USimS maintain efficiency comparable to their underlying PKE counterparts, support an unbounded number of key updates and enable asynchronous key updates for both encryption and decryption, eliminating the need for sender-receiver coordination. We sum up our contribution below.

– We first prove that both the public key encryption schemes hPKE and SimS achieve f-*circular-secure and leakage-resilient* (f-CS + LR) security under the CSSIDH assumption. We provide rigorous security analysis and formally establish the IND-CR-CPA security of UhPKE under the CSSIDH assumption and the IND-CR-CCA security of USimS under the CSSIDH and CSSIKoE assumptions using the f-CS + LR security of hPKE and SimS respectively. We emphasized that USimS is the *first* isogeny-based UPKE scheme that exhibits IND-CR-CCA security with a formal security proof in the standard security model. The existing isogeny-based UPKE scheme of Eaton et al. [15] achieves IND-CR-CPA security while the scheme of Duparc et al. [16] provides only OW-CR-CPA security which is a weaker security framework. Besides, none of them

Table 1 Comparative analysis of existing isogeny-based UPKE schemes with respect to storage and communication cost

Scheme	Key size		Ciphertext size	Size of updated ciphertext	Asynchronous
	$\lvert\mathsf{sk}\rvert$	$\lvert\mathsf{pk}\rvert$	$\lvert\mathsf{ct}_m\rvert$	$\lvert\mathsf{up}\rvert$	
[15]	1 in $[-\mu,\mu]^n$	1 in $\mathbb{F}_p$	1 in $\mathbb{F}_p$, $\lvert\mathsf{ct}_{\mathsf{DEM}}\rvert$	–	NO
[16]	10 in $\mathbb{F}_p$	2 in $\mathbb{F}_p$	2 in $\mathbb{F}_p$, 2 in $E(\overline{\mathbb{F}}_p)$	–	NO
UhPKE	1 in $[-\mu,\mu]^n$	1 in $\mathbb{F}_p$	1 in $\mathbb{F}_p$, 1 in $\{0,1\}^{\mathsf{mlen}(\lambda)}$	1 in $\mathbb{F}_p$, 1 in $\{0,1\}^{\mathsf{mlen}(\lambda)}$	YES
USimS	1 in $[-\mu,\mu]^n$	1 in $\mathbb{F}_p$	1 in $\mathbb{F}_p$, 1 in $E(\mathbb{F}_p)$	1 in $\mathbb{F}_p$, 1 in $E(\mathbb{F}_p)$	YES

$\lvert\mathsf{pk}\rvert$= the size of the public key, $\lvert\mathsf{sk}\rvert$= the size of the secret key, $\lvert\mathsf{ct}_m\rvert$ = the size of the ciphertext and $\lvert\mathsf{up}\rvert$= the size of the updated ciphertext. The field $\mathbb{F}_p$ consists of p elements where p is a prime number and $\overline{\mathbb{F}}_p$ represents its algebraic closure. The notation $E(\mathbb{F}_p)$ denotes an elliptic curve defined over a field $\mathbb{F}_p$. The function $\mathsf{mlen}(\lambda)$ is a polynomial dependent on the security parameter λ. μ and n are integers such that $(2\mu+1)^n \geq \#\mathsf{Cl}(\mathcal{O})$ where $\mathsf{Cl}(\mathcal{O})$ represents the ideal class group of an order $\mathcal{O}$.

Table 2 Complexity of our UKEM schemes in terms of storage cost and communication cost.

Scheme	Storage cost		Communication cost	
	$\lvert\mathsf{sk}\rvert$	$\lvert\mathsf{pk}\rvert$	$\lvert\mathsf{hct}\rvert$	$\lvert\mathsf{up}\rvert$
UKEM_1	1 in $[-\mu,\mu]^n$	1 in $\mathbb{F}_p$	1 in $\mathbb{F}_p$, 1 in $\{0,1\}^{\mathsf{mlen}(\lambda)}$	1 in $\mathbb{F}_p$, 1 in $\{0,1\}^{\mathsf{mlen}(\lambda)}$
UKEM_2	1 in $[-\mu,\mu]^n$	1 in $\mathbb{F}_p$	1 in $\mathbb{F}_p$, 1 in $E(\mathbb{F}_p)$	1 in $\mathbb{F}_p$, 1 in $E(\mathbb{F}_p)$

$\lvert\mathsf{pk}\rvert$= the size of the public key, $\lvert\mathsf{sk}\rvert$= the size of the secret key, $\lvert\mathsf{hct}\rvert$ = the size of the header ciphertext, $\lvert\mathsf{up}\rvert$= the size of the updated ciphertext. The field $\mathbb{F}_p$ consists of p elements where p is a prime number. The notation $E(\mathbb{F}_p)$ denotes an elliptic curve defined over a field $\mathbb{F}_p$. The function $\mathsf{mlen}(\lambda)$ is a polynomial dependent on the security parameter λ. μ and n are integers such that $(2\mu+1)^n \geq \#\mathsf{Cl}(\mathcal{O})$ where $\mathsf{Cl}(\mathcal{O})$ represents the ideal class group of an order $\mathcal{O}$.

Table 3 Complexity of our UKEM schemes in terms of computation cost.

Scheme	Computation cost				
	Key Generation	Enc	Dec	pk update	sk update
UKEM_1	1 GA	2 GA	1 GA	3 GA	1 GA
UKEM_2	1 GA	2 GA	1 GA, 1 DL	3 GA	1 GA, 1 DL

GA = Group action and DL = Discrete Logarithm.

supports asynchronous key updates for the encryption and decryption key, unlike our constructions for UPKE.

– Table 1 presents a comparative analysis of our proposed schemes UhPKE and USimS against existing UPKE constructions [15, 16] in terms of storage and communication overhead. We exclude the SIDH-based UPKE construction of [15] from our analysis as it is no longer considered secure. The secret and public keys of our proposed schemes are a single element from $[-\mu,\mu]^n$ and an element from $\mathbb{F}_p$ respectively, similar to the

UPKE scheme of [15]. In contrast, the UPKE scheme in [16] is more expensive in terms of secret key size and public key size as it requires ten elements from $\mathbb{F}_p$ for secret key and two elements from $\mathbb{F}_p$ for public keys. The ciphertext in UhPKE consists of one element from $\mathbb{F}_p$ and one element from $\{0,1\}^{\mathsf{mlen}(\lambda)}$ whereas that of USimS has one element from $\mathbb{F}_p$ and one element from $E(\mathbb{F}_p)$. In comparison, the ciphertext of UPKE in [16] requires two element from $\mathbb{F}_p$ and two elements from $E(\overline{\mathbb{F}}_p)$. In contrast, the ciphertext in [15] comprises of one element in $\mathbb{F}_p$ along with a ciphertext $\mathsf{ct}_{\mathsf{DEM}}$ from a *Data Encapsulation Mechanism* (DEM). In order to facilitate asynchronous key updates, our UPKE schemes send an additional updated ciphertext up that is required by none of the schemes [15] and [16]. However, the UPKE of [15, 16] are not synchronous and require sender-receiver coordination, making them unsuitable for key management.

– In the UPKE scheme of Eaton et al. [15], the secret key is randomly chosen from the range $[-\mu, \mu]^n$ and the public key is generated using group actions. The encryption process involves two group actions along with additional encryption via a generic DEM scheme. Decryption requires one group action and decryption via the DEM scheme. In the UPKE scheme by Duparc et al. [16], the secret key is a long isogeny walk of length t starting from a base supersingular curve E_0. The walk involves a sequence of isogeny computations including generating isogenies from kernels, evaluating torsion points and performing isogeny computations in higher dimensions. The public key is the resulting curve E_t. Encryption in this scheme involves a single isogeny computation and masking of torsion points while decryption requires computing the inverse isogeny in higher dimensions and solving a discrete logarithm problem over a cyclic group. The secret key in our proposed schemes UhPKE and USimS is chosen uniformly from $[-\mu, \mu]^n$ and the public key is generated through group actions similar to [15]. In our UhPKE, encryption and decryption require two and one group actions respectively, but do not require any DEM ciphertext formation and decryption unlike [15]. In our USimS scheme, encryption involves two group actions while decryption requires one group action and computing a discrete logarithm in a cyclic group. Updating the public and secret keys in the UPKE schemes of [16], [15] involves similar computations as those in their key generation algorithm. In contrast, our UhPKE scheme requires three group actions for an encryption key update and one for a decryption key update. In USimS, an encryption key update involves three group actions while a decryption key update requires one group action and computing a discrete logarithm in a cyclic group.

Beyond UPKE, research on UKEM remains relatively sparse. Haidar et al. [11] introduced the notion of UKEM, but their work primarily focuses on constructions with a bounded number of updates. Expanding UKEM to support unbounded key updates remains an open challenge with significant implications for the security and efficiency of long-term cryptographic protocols. We extend and refine the formal definition and security model of UKEM introduced in [11]. The framework in [11] defines UKEM with correctness and security constraints under a bounded number of updates. We generalize this model to support an unbounded number of updates to significantly enhance its flexibility and practical utility. We introduced two constructions for UKEM with

comprehensive security analysis in our proposed security framework. More concretely, our proposed UKEM constructions have the following salient features.

- Our first UKEM construction UKEM_1 is derived from UhPKE and achieves IND-CR-CPA security under CSSIDH assumption in our proposed security framework that supports unbounded number of key updates. Our second UKEM construction UKEM_2 is based on USimS and exhibits IND-CR-CCA security under CSSIDH and CSSIKoE assumptions in our proposed security framework allowing an unbounded number of key updates.
- In UKEM_1, the storage cost includes a secret key in $[-\mu, \mu]^n$ and a public key in $\mathbb{F}_p$. The communication cost involves a header ciphertext and an updated ciphertext each consisting of one element in $\mathbb{F}_p$ and one in $\{0,1\}^{\mathsf{mlen}(\lambda)}$. The computation cost includes two group actions for encapsulation, one for decapsulation and key updates requiring three group actions for encryption key updates and one for decryption key updates. In UKEM_2, the storage and communication costs remain similar as that of UKEM_1 shown [see Table 2, 3], but the computation cost differs as decapsulation involves one group action and computation of one discrete logarithm in a cyclic group. Key updates in UKEM_2 require three group actions for an encapsulation key update while a decapsulation key update involves one group action and computation of one discrete logarithm in a cyclic group.
- To the best of our knowledge, the UKEM construction of Haidar et al. [11] is the only existing post-quantum secure UKEM scheme. They have proposed a generic construction of UKEM from any IND-CR-CPA secure UPKE and presented an instantiation based on their IND-CR-CPA secure UPKE under the LWE assumption. Furthermore, they have achieved IND-CR-CCA secure UKEM supporting only a bounded number of updates. In contrast, our proposed UKEM_1 satisfies IND-CR-CPA security and USimS achieves IND-CR-CCA security with unbounded key updates. Our UKEM constructions do not require any zero-knowledge proofs unlike [11] and are computationally more friendly.

1.2 Technical Overview

IND-CR-CPA secure updatable encryption scheme UhPKE. The starting point of our UPKE constructions UhPKE is the hash-based public key encryption scheme hPKE from [21] in the isogeny setting where we skillfully introduced key update techniques asynchronously. A user u randomly generates its initial secret key $\mathsf{sk}_0^{(\mathsf{u})} = \boldsymbol{a}_0 \in [-\mu, \mu]^n$ and the corresponding public key is $\mathsf{pk}_0^{(\mathsf{u})} = [\boldsymbol{a}_0]E_0 \in \mathsf{Ell}_p(\mathcal{O})$ where E_0 denotes the publicly available base elliptic curve $y^2 = x^3 + x$ and $\mathsf{Ell}_p(\mathcal{O})$ represents the set of $\mathbb{F}_p$-isomorphic classes of supersingular curves E whose $\mathbb{F}_p$-endomorphism ring $\mathsf{End}_{\mathbb{F}_p}(E) \cong \mathcal{O} = \mathbb{Z}[\sqrt{-p}]$. The encryption of a message $m \in \{0,1\}^{\mathsf{mlen}(\lambda)}$ under the public key $\mathsf{pk}_i^{(\mathsf{u})}$ at epoch i is an hPKE ciphertext $\mathsf{ct}_m = (\mathsf{ct}_m^{(1)} = [\mathbf{b}]E_0, \mathsf{ct}_m^{(2)} = m \oplus H_k(M_C([\mathbf{b}]\mathsf{pk}_i^{(\mathsf{u})})))$ for some $\mathbf{b} \in [-\mu, \mu]^n$ where $H_k : \mathbb{F}_p \to \{0,1\}^{\mathsf{mlen}(\lambda)}$ is entropy smoothing hash function and $M_C(E)$ denotes the Montgomery coefficient of the elliptic curve E. To update a public key $\mathsf{pk}_i^{(\mathsf{u})}$ at the i-th epoch, one can simply sample randomness $\boldsymbol{\rho}_i \in \{0,1\}^{\mathsf{mlen}(\lambda)}$ and encrypts $\boldsymbol{\rho}_i$ to generate updated ciphertext $\mathsf{up}_{i+1} = \mathsf{ct}_{\boldsymbol{\rho}_i}$ under $\mathsf{pk}_i^{(\mathsf{u})}$

with $\mathsf{ct}_{\boldsymbol{\rho}_i} = (\mathsf{ct}^{(1)}_{\boldsymbol{\rho}_i} = [\mathbf{b}']E_0, \mathsf{ct}^{(2)}_{\boldsymbol{\rho}_i} = \boldsymbol{\rho}_i \oplus H_k(M_C([\mathbf{b}']\mathsf{pk}^{(\mathsf{u})}_i)))$ for some $\mathbf{b}' \in [-\mu, \mu]^n$ while $\mathsf{pk}^{(\mathsf{u})}_i$ to $\mathsf{pk}^{(\mathsf{u})}_{i+1} = [-\mathsf{KDF}(\boldsymbol{\rho}_i)]\mathsf{pk}^{(\mathsf{u})}_i$ for epoch $i+1$ using a key derivation function $\mathsf{KDF} : \{0,1\}^{\mathsf{mlen}(\lambda)} \to [-\mu, \mu]^n$. The updated secret key is $\mathsf{sk}^{(\mathsf{u})}_{i+1} = \mathsf{sk}^{(\mathsf{u})}_i - \mathsf{KDF}(\boldsymbol{\rho}_i)$.

An IND-CR-CPA attacker first observes the initial public key $\mathsf{pk}^{(\mathsf{u})}_0 = [\boldsymbol{a}_0]E_0$ and can make an initial sequence of updates using private randomness $\boldsymbol{\rho}_0, \ldots, \boldsymbol{\rho}_{t-1}$. At epoch t, the adversary requests a challenge ciphertext corresponding to a pair of plaintexts $(m^*_0, m^*_1) \in \{0,1\}^{\mathsf{mlen}(\lambda)} \times \{0,1\}^{\mathsf{mlen}(\lambda)}$. The challenge ciphertext $\mathsf{ct}_{m^*_b} = (\mathsf{ct}^{(1)}_{m^*_b}, \mathsf{ct}^{(2)}_{m^*_b})$ is an encryption of m^*_b under the updated public key $\mathsf{pk}^{(\mathsf{u})}_t = [-\sum_{i=0}^{t-1} \mathsf{KDF}(\boldsymbol{\rho}_i)]\mathsf{pk}^{(\mathsf{u})}_0$ where $b \in \{0,1\}$ is chosen randomly by the challenger. The adversary may then continue updating the public key with additional randomness $\boldsymbol{\rho}_t, \ldots, \boldsymbol{\rho}_{t'-1}$ before deciding to compromise the secret key. At this point, the challenger performs an additional honest update using a randomness $\boldsymbol{\rho}^*$ unknown to the adversary, yielding the compromised secret key $\mathsf{sk}^* = \mathsf{sk}^{(\mathsf{u})}_0 - \sum_{i=0}^{t'-1} \mathsf{KDF}(\boldsymbol{\rho}_i) - \mathsf{KDF}(\boldsymbol{\rho}^*)$ and the corresponding public key $\mathsf{pk}^* = [-\sum_{i=0}^{t'-1} \mathsf{KDF}(\boldsymbol{\rho}_i) - \mathsf{KDF}(\boldsymbol{\rho}^*)]\mathsf{pk}^{(\mathsf{u})}_0$. The adversary's goal is to guess the correct bit b using the challenge ciphertext $\mathsf{ct}_{m^*_b}$ given access to the secret key sk^*, the public key pk^* and the updated ciphertext up^* which encrypts the randomness $\boldsymbol{\rho}^*$ under pk^*. The IND-CR-CPA security of UhPKE requires the hPKE to construction satisfy f-CS + LR security. The f-CS + LR security framework for hPKE allows an adversary against the IND-CPA security of hPKE additionally receives a leakage function $f(\mathsf{sk}^{(\mathsf{u})}_0, \boldsymbol{\rho}^*) = \mathsf{sk}^{(\mathsf{u})}_0 - \mathsf{KDF}(\boldsymbol{\rho}^*)$ of $\mathsf{sk}^{(\mathsf{u})}_0$ along with an encryption $\mathsf{up} = \mathsf{ct}_{\boldsymbol{\rho}^*}$ of $\boldsymbol{\rho}^*$ under $\mathsf{pk}^{(\mathsf{u})}_0$. The proof is under the CSSIDDH assumption and follows three key steps:

i. eliminating all information about $\mathsf{sk}^{(\mathsf{u})}_0$ except $\mathsf{sk}^{(\mathsf{u})}_0 - \mathsf{KDF}(\boldsymbol{\rho}^*)$ while treating $\mathsf{KDF}(\boldsymbol{\rho}^*)$ as the secret key,
ii. replacing $\mathsf{sk}^{(\mathsf{u})}_0 - \mathsf{KDF}(\boldsymbol{\rho}^*)$ with a uniformly random element from $[-\mu, \mu]^n$ to ensure that the adversary's view is independent of sk and
iii. leveraging the CSSIDDH assumption and the entropy smoothness property of H_k to argue that distinguishing the correct plaintext remains computationally infeasible for the adversary.

This security argument demonstrates that the hPKE construction can be transformed into an efficient IND-CR-CPA secure UPKE scheme. Moreover, our approach of choosing randomness from the message space enables single-shot encryption of the entire update information $\boldsymbol{\rho}$, avoiding bit-by-bit encryption overhead. More concretely, we have the following theorems.

Theorem 1 (Informally)**.** *The scheme* hPKE *provides* f-CS + LR *security under the* CSSIDDH *assumption assuming* $\mathcal{H} = \{H_k\}_{k \in \mathcal{K}}$ *is an entropy smoothing hash function and* KDF *is a secure key derivation function.*

Theorem 2 (Informally)**.** *If* hPKE *is* f-CS + LR *secure then the isogeny based* UhPKE *construction provides* IND-CR-CPA *security.*

Definition 3 (Randomizing Function [22]). *A function $R_E : \mathbb{F}_p \to \mathbb{F}_p$ indexed by supersingular curves E defined over $\mathbb{F}_p$ is said to be a randomizing function if it satisfies the following properties:*

i. The function R_E is bijective and both R_E and its inverse $g_E = R_E^{-1}$ can be efficiently computed when the curve E is given.
ii. For any $x \in \mathbb{F}_p$, an adversary without access to x and E cannot distinguish $R_E(x)$ from a random element of $\mathbb{F}_p$.
iii. For any $x \in \mathbb{F}_p$ and any non-identical rational function $R(x) \in \mathbb{F}_p(X)$, an adversary without access to x and E cannot compute $R_E(R(x))$ given $R_E(x)$.

Example 1 Consider the function $R_E : \mathbb{F}_p \to \mathbb{F}_p$ defined by $R_E(x) = \mathsf{int}\left(\mathsf{bin}(x) \oplus \mathsf{bin}(M_C(E))\right)$ where $\mathsf{bin}(\cdot)$ and $\mathsf{int}(\cdot)$ denote the operations that convert an element in $\mathbb{F}_p$ to its binary representation and vice versa, respectively and $M_C(E)$ is the Montgomery coefficient of the elliptic curve E.

IND-CR-CCA secure updatable encryption scheme USimS. Our UPKE construction USimS builds upon the *simplified sigamal* scheme SimS introduced in [22]. We handle the asynchronous key update by employing a key derivation function $\mathsf{KDF} : \mathbb{Z}_{2^{q-2}} \to [-\mu, \mu]^n$. A user u randomly generates its initial secret key $\mathsf{sk}_0^{(\mathsf{u})} = \boldsymbol{a}_0 \in [-\mu, \mu]^n$ with the corresponding public key $\mathsf{pk}_0^{(\mathsf{u})} = [\boldsymbol{a}_0]E_0 \in \mathsf{Ell}_p(\mathcal{O})$ where E_0 represents the base elliptic curve $y^2 = x^3 + x$. The encryption corresponding to a message $m \in \mathbb{Z}_{2^{q-2}}$ under the public key $\mathsf{pk}_i^{(\mathsf{u})}$ at epoch i is an SimS ciphertext $\mathsf{ct}_m = \left(\mathsf{ct}_m^{(1)} = [\mathbf{b}]E_0,\ \mathsf{ct}_m^{(2)} = R_{E_{[\mathbf{b}]\mathsf{pk}_i^{(\mathsf{u})}}}\left(x([2m+1]P_{[\mathbf{b}]\mathsf{pk}_i^{(\mathsf{u})}})\right)\right)$ for some $\mathbf{b} \in [-\mu, \mu]^n$ where R_E is a randomizing function. To update the public key $\mathsf{pk}_i^{(\mathsf{u})}$ at the i-th epoch, the user samples randomness $\boldsymbol{\rho}_i \in \mathbb{Z}_{2^{q-2}}$ and generates an updated ciphertext $\mathsf{up}_{i+1} = \mathsf{ct}_{\boldsymbol{\rho}_i}$ by encrypting $\boldsymbol{\rho}_i$ under $\mathsf{pk}_i^{(\mathsf{u})}$ where $\mathsf{ct}_{\boldsymbol{\rho}_i} = \left(\mathsf{ct}_{\boldsymbol{\rho}_i}^{(1)} = [\mathbf{b}']E_0,\ \mathsf{ct}_{\boldsymbol{\rho}_i}^{(2)} = R_{E_{[\mathbf{b}]\mathsf{pk}_i^{(\mathsf{u})}}}\left(x([2\boldsymbol{\rho}_i+1]P_{[\mathbf{b}']\mathsf{pk}_i^{(\mathsf{u})}})\right)\right)$ for some $\mathbf{b}' \in [-\mu, \mu]^n$ and R_E is a randomizing function. The public key is then updated as $\mathsf{pk}_{i+1}^{(\mathsf{u})} = [-\mathsf{KDF}(\boldsymbol{\rho}_i)]\mathsf{pk}_i^{(\mathsf{u})}$ and the updated secret key is updated as $\mathsf{sk}_{i+1}^{(\mathsf{u})} = \mathsf{sk}_i^{(\mathsf{u})} - \mathsf{KDF}(\boldsymbol{\rho}_i)$.

To establish that USimS is an IND-CR-CCA secure updatable encryption scheme, we first demonstrate it's IND-CR-CPA secure under the hardness of the CSSIDDH assumption and then provide a reduction-based proof showing that IND-CR-CPA security of USimS implies its IND-CR-CCA security by additionally assuming the hardness of *commutative supersingular isogeny knowledge of exponent* (CSSIKoE) assumption. The IND-CR-CPA security of USimS requires SimS to satisfy f-CS + LR security which we prove by using a similar technique as f-CS + LR security of the hPKE and employing the second properties of the randomizing function R_E. We then follow the same approach as the IND-CR-CPA security proof of UhPKE to establish the IND-CR-CPA security of USimS. In contrast to the IND-CR-CPA security setting, the IND-CR-CCA game grants the adversary access to a decryption oracle. To prove that USimS is IND-CR-CCA secure, it suffices to prove that this decryption oracle is effectively useless.

This follows directly from the CSSIKoE assumption, which states that given $E_0, [\mathbf{b}]E_0$ and a valid ciphertext $\mathsf{ct}_m = (\mathsf{ct}_m^{(1)} = [\mathbf{b}]E_0, \mathsf{ct}_m^{(2)} = R_{E_{[\mathbf{b}]\mathsf{pk}_i^{(u)}}}(x([2m+1]P_{[\mathbf{b}]\mathsf{pk}_i^{(u)}})))$ a *probabilistic polynomial time* (PPT) adversary cannot construct a valid new ciphertext $\widetilde{\mathsf{ct}}$ from a previously obtained ciphertext unless $\widetilde{\mathsf{ct}}$ is generated using the encryption algorithm. This ensures that decryption queries do not provide the adversary with any advantage, thereby establishing the IND-CR-CCA security of USimS. More precisely, we have the following theorems.

Theorem 4 (Informally)**.** *The scheme* SimS *is* f-CS + LR *secure under the assumption that* R_E *satisfies the second property of a randomizing function,* KDF *is a secure key derivation function and the* CSSIDDH *assumption.*

Theorem 5 (Informally)**.** *If* SimS *is* f-CS + LR *secure then our isogeny based* UPKE *construction* USimS *is* IND-CR-CPA *secure.*

Theorem 6 (Informally)**.** *If* CSSIKoE *assumption holds then our isogeny based* IND-CR-CPA *secure* UPKE *construction* USimS *provides* IND-CR-CCA *security.*

Updatable key encapsulation mechanism UKEM_1 and UKEM_2. We transform our updatable public key encryption schemes UhPKE and USimS to updatable key encapsulation mechanism UKEM_1 and UKEM_2 respectively using an inherently similar technique. In this transformation, the key generation and key update algorithms remain unchanged. The encapsulation process samples a random message m from the message space, computes $\mathsf{KDF}(m)$, encrypts it under the recipient's public key to generate a header ciphertext hct and sends hct to the recipient. The decapsulation process run by the recipient to decrypt hct using its secret key, recovers m and computes $\mathsf{KDF}(m)$ using the public key derivation function $\mathsf{KDF} : \mathbb{Z}_{2^{q-2}} \to \{0,1\}^{\mathsf{klen}(\lambda)}$. This transformation ensures that the resulting UKEM inherits the security properties of the underlying UPKE, supporting asynchronous unlimited key updates. Specifically, we have the following theorems.

Theorem 7 (Informally)**.** *If* UhPKE *is* IND-CR-CPA *secure and* KDF *is a secure key derivation function then* UKEM_1 *provides* IND-CR-CPA *security.*

Theorem 8 (Informally)**.** *If* USimS *is* IND-CR-CCA *secure and* KDF *is a secure key derivation function then* UKEM_2 *provides* IND-CR-CCA *security.*

2 Preliminaries

Notation. Throughout the paper, we adopt the following notations. Let $\#S$ denote the *cardinality* of the set S, i.e., the number of elements in S. The notation $a \xleftarrow{\$} A$ indicates that a is uniformly sampled from the set A. A function $\epsilon(\cdot)$ is called *negligible* if, for every positive integer c, there exists an integer k such that for all $\lambda > k$, $|\epsilon(\lambda)| < 1/\lambda^c$.

Elliptic curves and isogenies [23]. Let K be a finite field and $\overline{K}$ be its algebraic closure. An elliptic curve E over K is a non-singular projective cubic curve having genus one with a special point O, called the point at infinity. The set of K-rational points of the elliptic curve E forms an additive abelian group with O as the identity element. If P is a point on E, its coordinates are denoted as $P = (x(P), y(P))$ where $x(P)$ and $y(P)$ represent its x- and y-coordinates, respectively. The set $E(\overline{K})$ consists of all points on E whose coordinates belong to $\overline{K}$, i.e., $E(\overline{K}) = \{P = (x, y) \mid x, y \in \overline{K}, P \text{ satisfies } E\} \cup \{O\}$. The *$\ell$-torsion subgroup* of E, denoted as $E[\ell]$, is the set of all points in $E(\overline{K})$ that satisfy $\ell P = O$. The *Montgomery coefficient* of the *Montgomery elliptic curve* $E_A : y^2 = x^3 + Ax^2 + x$ is denoted by M_C and defined by $M_C(E_A) = A$.

Let E_1 and E_2 be two elliptic curves over a field K. An *isogeny* from E_1 to E_2 is a non-constant morphism $\varphi : E_1 \to E_2$ over $\overline{K}$ preserving the point at infinity O. The isogeny $\varphi : E_1 \to E_2$ can be expressed in its simplest form as $\varphi(x, y) = \left(\frac{p(x)}{q(x)}, \frac{r(x)}{s(x)}y\right)$ where the polynomial $p(x)$ and $q(x)$ have no common factor and the polynomial $r(x)$ and $s(x)$ have no common factor. The *degree* of a nonzero isogeny is defined as the degree of the associated morphism and is given by $\mathsf{deg}(\varphi) = \max\{\mathsf{deg}(p(x)), \mathsf{deg}(q(x))\}$. A non-zero isogeny φ is called *separable* if and only if $\mathsf{deg}(\varphi) = \#\mathsf{ker}(\varphi)$ where $\mathsf{ker}(\varphi) = \varphi^{-1}(O_{E_2})$ where O_{E_2} is the identity element of the elliptic curve E_2.

Endomorphism ring. The set of all isogenies from E to itself defined over $\overline{K}$ forms a ring under pointwise addition and composition. This ring is called the *endomorphism ring* of the elliptic curve E and is denoted by $\mathsf{End}(E)$. By $\mathsf{End}_K(E)$, we mean the set of all isogenies from E to itself defined over K. If $\mathsf{End}(E)$ is isomorphic to an order in a quaternion algebra, the curve E is said to be *supersingular*. On the other hand, if $\mathsf{End}(E)$ is isomorphic to an order in an imaginary quadratic field, we say the curve E is *ordinary*.

Theorem 9 ([24]). *Let $p \geq 5$ be a prime such that $p \equiv 3 \pmod 8$ and let $E/\mathbb{F}_p$ be a supersingular elliptic curve. Then $\mathsf{End}_p(E) \cong \mathbb{Z}[\sqrt{-p}]$ if and only if there exists $A \in \mathbb{F}_p$ such that E is $\mathbb{F}_p$-isomorphic to the curve $E_A : y^2 = x^3 + Ax^2 + x$. Additionally, in the presence of such an A, it is guaranteed to be unique.*

Theorem 10 ([25]). *Let E_1 be a curve and G be its finite subgroup. Then there is a unique curve E_2 and a separable isogeny $\varphi : E_1 \to E_2$ with $\mathsf{ker}(\varphi) = G$ such that $E_2 \cong E_1/G$ which can be computed using Vélu's formulae (see Algorithm 1).*

Ideal class group. [21] Let $\mathcal{O}$ be an order in the imaginary quadratic field F. A fractional ideal $\mathfrak{a}$ of $\mathcal{O}$ is a finitely generated $\mathcal{O}$-submodule of F. Let $\mathcal{I}(O)$ be a set of invertible fractional ideals of $\mathcal{O}$. Then $\mathcal{I}(\mathcal{O})$ is an abelian group derived from the multiplication of ideals with the identity $\mathcal{O}$. Let $\mathcal{P}(\mathcal{O})$ be a subgroup of $\mathcal{I}(\mathcal{O})$ defined by $\mathcal{P}(\mathcal{O}) = \{\mathfrak{a} \mid \mathfrak{a} = \alpha\mathcal{O} \text{ for some } \alpha \in F \setminus \{0\}\}$. The abelian group $\mathsf{Cl}(\mathcal{O})$, defined by $\mathcal{I}(\mathcal{O})/\mathcal{P}(\mathcal{O})$, is called the *ideal class group* of $\mathcal{O}$. An element of $\mathsf{Cl}(\mathcal{O})$, denoted by $[\mathfrak{a}]$, is an equivalence class of $\mathfrak{a}$.

The class group action. Let p be a prime and $\mathsf{Ell}_p(\mathcal{O})$ denotes the set of $\mathbb{F}_p$-isomorphic classes of supersingular curves E whose $\mathbb{F}_p$-endomorphism ring

$\mathsf{End}_{\mathbb{F}_p}(E) \cong \mathcal{O} = \mathbb{Z}[\sqrt{-p}]$. The ideal class group $\mathsf{Cl}(\mathcal{O})$ acts freely and transitively on $\mathsf{Ell}_p(\mathcal{O})$. An element $[\mathfrak{a}]$ in $\mathsf{Cl}(\mathcal{O})$ consists of endomorphisms α in $\mathfrak{a}$ which an isogenies from E to itself over $\overline{\mathbb{F}}_p$. For the curve $E \in \mathsf{Ell}_p(\mathcal{O})$, the *action* $*$ of $[\mathfrak{a}] \in \mathsf{Cl}(\mathcal{O})$ on E is denoted by $[\mathfrak{a}] * E$ and defined as follows:

- Form the subgroup $E[\mathfrak{a}] = \bigcap_{\alpha \in \mathfrak{a}} \ker(\alpha)$.
- Apply Vélu's formula (see Algorithm 1) to compute the elliptic curve $E/E[\mathfrak{a}]$ and an isogeny $\varphi_{\mathfrak{a}} : E \to E/E[\mathfrak{a}]$.
- Return the elliptic curve $E/E[\mathfrak{a}]$.

Henceforth, we will use the notation $[\mathfrak{a}]E$ instead of $[\mathfrak{a}] * E$ to denote the elliptic curve $E/E[\mathfrak{a}]$ obtained by the action of class group element $[\mathfrak{a}] \in \mathsf{Cl}(\mathcal{O})$ on the elliptic curve $E \in \mathsf{Ell}_p(\mathcal{O})$.

Theorem 11 ([25])**.** *Let p be prime and $\mathcal{O}$ be an order of an imaginary quadratic field and E be an elliptic curve defined over $\mathbb{F}_p$. If $\mathsf{Ell}_p(\mathcal{O})$ contains the $\mathbb{F}_p$-isomorphism class of supersingular elliptic curves then the action of the ideal class group $\mathsf{Cl}(\mathcal{O})$ on $\mathsf{Ell}_p(\mathcal{O})$, defined by*

$$\begin{gathered}\mathsf{Cl}(\mathcal{O}) \times \mathsf{Ell}_p(\mathcal{O}) \to \mathsf{Ell}_p(\mathcal{O}) \\ ([\mathfrak{a}], E) \to E/E[\mathfrak{a}]\end{gathered}$$

is free and transitive where $\mathfrak{a}$ is an integral ideal of $\mathcal{O}$ and $E[\mathfrak{a}]$ is the intersection of the kernels of elements in $\mathfrak{a}$.

Let p be prime and E be an elliptic curve defined over the finite field $\mathbb{F}_p$. Consider the map π acting on the coordinates of points in $E(\overline{\mathbb{F}}_p)$, given by $\pi(x, y) = (x^p, y^p)$ and $\pi(O) = O$. This map π is an endomorphism of E and is known as the *Frobenius endomorphism*. For every small odd prime ℓ_i dividing $p + 1$, there are two prime ideals $\mathfrak{l}_i = \langle \ell_i, \pi - 1 \rangle$ and $\bar{\mathfrak{l}}_i = \langle \ell_i, \pi + 1 \rangle$ in $\mathsf{Cl}(\mathcal{O})$. Also, the kernel of the isogeny corresponding to the action of the prime ideals $\mathfrak{l}_i = \langle \ell_i, \pi - 1 \rangle$ and $\bar{\mathfrak{l}}_i = \langle \ell_i, \pi + 1 \rangle$ is generated by $P_{\mathfrak{l}_i} \in E_0[\ell_i] \cap \ker(\pi - 1) \setminus \{0\}$ and $P_{\bar{\mathfrak{l}}_i} \in E_0[\ell_i] \cap \ker(\pi + 1) \setminus \{0\}$ respectively. For the sake of simplicity, we will write $[\boldsymbol{a}]E$ instead of $[\mathfrak{a}]E$ for any element $[\mathfrak{a}] = [\mathfrak{l}_1^{a_1} \cdots \mathfrak{l}_n^{a_n}] \in \mathsf{Cl}(\mathcal{O})$ where $\boldsymbol{a} = (a_1, \ldots, a_n)$ and $\mathfrak{l}_i = \langle \ell_i, \pi - 1 \rangle$ and $[\boldsymbol{a} + \mathbf{b}]E$ in the place of $[\mathfrak{a}][\mathfrak{b}]E$ for any two elements $[\mathfrak{a}]$, $[\mathfrak{b}] \in \mathsf{Cl}(\mathcal{O})$. Let λ be the security parameter, E_0 be the supersingular elliptic curve $y^2 = x^3 + x$ defined over $\mathbb{F}_p$ and $[\boldsymbol{a}]$, $[\mathbf{b}]$ and $[\mathbf{c}]$ be uniformly random ideal classes in $\mathsf{Cl}(\mathcal{O})$.

Definition 12 (CSSICDH[24])**.** *The commutative supersingular isogeny computational Diffie-Hellman* (CSSICDH) *assumption holds if for any Probabilistic Polynomial Time (PPT) algorithm $\mathcal{A}$,*

$$\Pr\left[E = [\mathbf{b}][\boldsymbol{a}]E_0 \mid E = \mathcal{A}(E_0, [\boldsymbol{a}]E_0, [\mathbf{b}]E_0)\right] \leq \epsilon(\lambda)$$

Definition 13 (CSSIDDH[24]). *The commutative supersingular isogeny decisional Diffie-Hellman* (CSSIDDH) *assumption holds if for any PPT distinguisher* $\mathcal{D}$,

$$\mathsf{Adv}_{\mathcal{D}}^{\mathsf{CSSIDDH}}(\lambda) = \left| \Pr\left[\mathcal{D}(E_0, [\boldsymbol{a}]E_0,\ [\mathbf{b}]E_0,\ [\mathbf{b}][\boldsymbol{a}]E_0) = 1 \mid [\boldsymbol{a}], [\mathbf{b}], [\mathbf{c}] \xleftarrow{\$} \mathsf{Cl}(\mathcal{O})\right] - \Pr\left[\mathcal{D}(E_0, [\boldsymbol{a}]E_0,\ [\mathbf{b}]E_0, [\mathbf{c}]E_0) = 1 \mid [\boldsymbol{a}], [\mathbf{b}], [\mathbf{c}] \xleftarrow{\$} \mathsf{Cl}(\mathcal{O})\right] \right| \leq \epsilon(\lambda).$$

Definition 14 (CSSIKoE [22]). *Let* λ *be a security parameter and* $p = 2^q \ell_1 \cdots \ell_n - 1$ *be a prime such that* $\lambda + 2 \leq q \leq \frac{1}{2}\log p$. *Let* $[a], [b]$ *be uniformly sampled elements of* $\mathsf{Cl}(\mathcal{O})$. *Let* $(R_E)_{E \in \mathsf{Cl}(\mathcal{O})}$ *be a family of randomizing functions as defined in Definition 3 such that each of these functions satisfies the third property.*

The commutative supersingular isogeny knowledge of exponent (CSSIKoE) *assumption states that for every PPT adversary* $\mathcal{A}$ *that takes* E_0, $[a]E_0$ *and* $\big([b]E_0, R_{[a][b]E_0}(x(P))\big)$ *as inputs and returns* $([b']E_0, R_{[a][b']E_0}(x(P')))$ *such that* $([b']E_0, R_{[a][b']E_0}(x(P'))) \neq ([b]E_0, R_{[a][b]E_0}(x(P)))$ *where* $P \in [a][b]E_0$ *and* $P' \in [a][b']E_0$ *are points of order* 2^q, *there exists a PPT adversary* $\mathcal{A}'$ *that takes the same inputs and returns* $([b'], [b']E_0, R_{[a][b']E_0}(x(P')))$.

Definition 15 (Entropy Smoothing Hash Function [21]). *Let* $\mathcal{H} = \{H_k\}_{k \in \mathcal{K}}$ *be a family of keyed hash functions where each* H_k *be a function that maps from* G *to* $\{0,1\}^l$ *where* l *denotes the length of the string. Let* $\mathcal{D}$ *be a distinguisher that takes as input an element of key space* $\mathcal{K}$ *and an element of* $\{0,1\}^l$ *and outputs a bit. We define the entropy smoothing advantage* $\mathsf{Adv}_{\mathcal{D}}^{\mathsf{ES}}(\lambda)$ *of* $\mathcal{D}$ *to be* $\mathsf{Adv}_{\mathcal{D}}^{\mathsf{ES}}(\lambda) = \Big|\mathcal{D}(k, H_k(g)) = 1 \mid \Pr[k \xleftarrow{\$} \mathcal{K}, g \xleftarrow{\$} G] - \Pr[\mathcal{D}(k,h) = 1 \mid k \xleftarrow{\$} \mathcal{K}, h \xleftarrow{\$} \{0,1\}^l]\Big|$. *We say that the family of keyed hash functions is entropy smoothing if the entropy smoothing advantage* $\mathsf{Adv}_{\mathcal{D}}^{\mathsf{ES}}(\lambda)$ *of any PPT distinguisher* $\mathcal{D}$ *is negligible.*

Definition 16 (Key Derivation Function (KDF)[21]). *A key derivation function* $\mathsf{KDF} : S(\lambda) \to T(\lambda)$ *is a deterministic function that takes a randomly sampled input* $u \in S(\lambda)$ *and produces an output that is computationally indistinguishable from a uniformly random element of* $T(\lambda)$. *A cryptographic key derivation function* KDF *is considered secure if* $\mathsf{Adv}_{\mathcal{D}}^{\mathsf{KDF}}(\lambda)$ *is negligible for any PPT distinguisher* $\mathcal{D}$ *where* $\mathsf{Adv}_{\mathcal{D}}^{\mathsf{KDF}}(\lambda)$ *the advantage in distinguishing* $\mathsf{KDF}(u)$ *from a truly random string is defined as* $\mathsf{Adv}_{\mathcal{D}}^{\mathsf{KDF}}(\lambda) = \left|\Pr\left[\mathcal{D}(\mathsf{KDF}(u)) = 1 \mid u \xleftarrow{\$} S(\lambda)\right] - \Pr\left[\mathcal{D}(k) = 1 \mid k \xleftarrow{\$} T(\lambda)\right]\right|$.

2.1 Public Key Encryption

Definition 17 (Public Key Encryption [21]). *A Public key encryption* (PKE) *is a tuple of four PPT algorithms* $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *associated with a secret key space* $\mathcal{KS}$, *a message space* $\mathcal{MS}$ *and ciphertext space* $\mathcal{CS}$ *satisfying the following requirements:*

Setup: The challenger $\mathcal{C}$ computes public parameter $\mathsf{pp}_{\mathsf{pke}} \leftarrow \mathsf{PKE.Setup}(\lambda)$ and secret-public key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{PKE.KeyGen}(\mathsf{pp}_{\mathsf{pke}})$. It forwards $\mathsf{pp}_{\mathsf{pke}}$ and pk to the adversary $\mathcal{A}$ while keeps sk secret to itself.

Challenge Phase: After receiving (m_0^*, m_1^*) from the adversary $\mathcal{A}$, the challenger $\mathcal{C}$ uniformly samples $b \xleftarrow{\$} \{0,1\}$ and sends $\mathsf{ct}_{m_b^*} \leftarrow \mathsf{PKE.Enc}(\mathsf{pp}_{\mathsf{pke}}, \mathsf{pk}, m_b^*)$ to $\mathcal{A}$.

Guess Phase: The adversary $\mathcal{A}$ eventually submits a bit b' to the challenger $\mathcal{C}$. If $b = b'$ then the experiment $\mathsf{Exp}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE},\mathcal{A}}(\lambda)$ returns 1, otherwise, it returns 0.

Fig. 1 $\mathsf{Exp}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE},\mathcal{A}}(\lambda)$: Indistinguishability under chosen-plaintext attacks

$\mathsf{PKE.Setup}(\lambda) \rightarrow \mathsf{pp}_{\mathsf{pke}}$*: A trusted party runs this algorithm on input the security parameter* λ *and outputs the public parameter* $\mathsf{pp}_{\mathsf{pke}}$.

$\mathsf{PKE.KeyGen}(\mathsf{pp}_{\mathsf{pke}}) \rightarrow (\mathsf{sk}, \mathsf{pk})$*: On input the public parameter* $\mathsf{pp}_{\mathsf{pke}}$*, a user generates its secret-public key pair* $(\mathsf{sk}, \mathsf{pk})$ *by running this algorithm.*

$\mathsf{PKE.Enc}(\mathsf{pp}_{\mathsf{pke}}, \mathsf{pk}, m) \rightarrow \mathsf{ct}_m$*: Taking as input the public parameter* $\mathsf{pp}_{\mathsf{pke}}$*, public key* pk *and a message* $m \in \mathcal{MS}$*, an encrypter runs this algorithm and returns a ciphertext* $\mathsf{ct}_m \in \mathcal{CS}$.

$\mathsf{PKE.Dec}(\mathsf{pp}_{\mathsf{pke}}, \mathsf{sk}, \mathsf{ct}_m) \rightarrow m/\perp$*: On input the public parameter* $\mathsf{pp}_{\mathsf{pke}}$*, the secret key* sk *and a ciphertext* ct_m*, the decrypter runs this algorithm and returns plaintext* m *or* $\perp$ *to indicate description failure.*

Definition 18 (Correctness)**.** *We say that a* PKE *scheme is correct if for all security parameters* λ*, all* $\mathsf{pp}_{\mathsf{pke}} \leftarrow \mathsf{PKE.Setup}(\lambda)$*, all* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.KeyGen}(\mathsf{pp}_{\mathsf{pke}})$*, all* $m \in \mathcal{MS}$*, it must hold that*

$$\mathsf{PKE.Dec}(\mathsf{pp}_{\mathsf{pke}}, \mathsf{sk}, \mathsf{PKE.Enc}(\mathsf{pp}_{\mathsf{pke}}, \mathsf{pk}, m)) = m$$

Definition 19 (IND-CPA)**.** *A public key encryption scheme* PKE *is secure against indistinguishability under chosen-plaintext attacks* (IND-CPA) *if the advantage* $\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE},\mathcal{A}}(\lambda)$ *of any PPT adversary* $\mathcal{A}$ *defined as*

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

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

We define below circular-secure and leakage-resilient ($\mathsf{CS} + \mathsf{LR}$) security for a public key encryption scheme. Our notion of f-$\mathsf{CS} + \mathsf{LR}$ security differs slightly from that defined by Dodis et al. [9]. Specifically, in our definition, the adversary is provided with a randomized leakage $f(\mathsf{sk}, \boldsymbol{\rho})$ of the secret key sk along with an encryption of $\boldsymbol{\rho}$ whereas that in [9] considers an encryption of a function of sk.

Definition 20 (f-$\mathsf{CR} + \mathsf{LS}$)**.** *Let* $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a* PKE *scheme with secret key space* $\mathcal{KS}$ *and message space* $\mathcal{MS}$*. We say that a* PKE *scheme is*

Setup: The challenger $\mathcal{C}$ generates public parameter $\mathsf{pp}_{\mathsf{pke}} \leftarrow \mathsf{PKE.Setup}(\lambda)$, secret-public key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{PKE.KeyGen}(\mathsf{pp}_{\mathsf{pke}})$ and function $f : \mathcal{KS} \times \mathcal{MS} \rightarrow \mathcal{KS}$. It forwards $\mathsf{pp}_{\mathsf{pke}}$ and pk to the adversary $\mathcal{A}$ while keeps sk secret to itself.
Challenge Phase: After receiving (m_0^*, m_1^*) from the adversary $\mathcal{A}$, the challenger $\mathcal{C}$ samples $b \xleftarrow{\$} \{0,1\}$ and $\boldsymbol{\rho}^* \xleftarrow{\$} \mathcal{MS}$ and sends $(f(\mathsf{sk}, \boldsymbol{\rho}^*), \mathsf{PKE.Enc}(\mathsf{pp}_{\mathsf{pke}}, \mathsf{pk}, m_b^*), \mathsf{PKE.Enc}(\mathsf{pp}_{\mathsf{pke}}, \mathsf{pk}, \boldsymbol{\rho}^*))$ to $\mathcal{A}$.
Guess Phase: The adversary $\mathcal{A}$ eventually submits a bit b' to the challenger $\mathcal{C}$ and if $b = b'$ then the experiment $\mathsf{Exp}^{f\text{-}\mathsf{CS+LR}}_{\mathsf{PKE},\mathcal{A}}(\lambda)$ returns 1, otherwise, it returns 0.

Fig. 2 $\mathsf{Exp}^{f\text{-}\mathsf{CS+LR}}_{\mathsf{PKE},\mathcal{A}}(\lambda)$: Circular-secure and leakage-resilient

f-CS + LR secure if the advantage $\mathsf{Adv}^{f\text{-}\mathsf{CS+LR}}_{\mathsf{PKE},\mathcal{A}}(\lambda)$ of any PPT adversary $\mathcal{A}$ defined as

$$\mathsf{Adv}^{f\text{-}\mathsf{CS+LR}}_{\mathsf{PKE},\mathcal{A}}(\lambda) = \left| \Pr[\mathsf{Exp}^{f\text{-}\mathsf{CS+LR}}_{\mathsf{PKE},\mathcal{A}}(\lambda) = 1] - \frac{1}{2} \right|$$

is negligible where the experiment $\mathsf{Exp}^{f\text{-}\mathsf{CS+LR}}_{\mathsf{PKE},\mathcal{A}}(\lambda)$ between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ is depicted in Fig.2.

2.2 Updatable Public Key Encryption

Definition 21 (Updatable Public Key Encryption)**.** *An Updatable Public Key Encryption* (UPKE) *is a tuple of PPT algorithms* $\mathsf{UPKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{UpdatePk}, \mathsf{UpdateSk})$ *associated with a message space $\mathcal{MS}$, a randomness space $\mathcal{RS}$ and ciphertext space $\mathcal{CS}$ satisfying the following requirements:*

$\mathsf{UPKE.Setup}(\lambda) \rightarrow \mathsf{pp}_{\mathsf{upke}}$*: A trusted party on input a security parameter λ outputs the public parameter* $\mathsf{pp}_{\mathsf{upke}}$.

$\mathsf{UPKE.KeyGen}(\mathsf{pp}_{\mathsf{upke}}) \rightarrow (\mathsf{sk}_0^{(\mathsf{u})}, \mathsf{pk}_0^{(\mathsf{u})})$*: On input the public parameter* $\mathsf{pp}_{\mathsf{upke}}$*, a user generates a secret-public key pair* $(\mathsf{sk}_0^{(\mathsf{u})}, \mathsf{pk}_0^{(\mathsf{u})})$.

$\mathsf{UPKE.Enc}(\mathsf{pp}_{\mathsf{upke}}, \mathsf{pk}_i^{(\mathsf{u})}, m) \rightarrow \mathsf{ct}_m$*: Taking as input the public parameter* $\mathsf{pp}_{\mathsf{upke}}$*, public key* $\mathsf{pk}_i^{(\mathsf{u})}$ *and a message $m \in \mathcal{MS}$, an encrypter runs this algorithm and returns a ciphertext $\mathsf{ct}_m \in \mathcal{CS}$.*

$\mathsf{UPKE.Dec}(\mathsf{pp}_{\mathsf{upke}}, \mathsf{sk}_i^{(\mathsf{u})}, \mathsf{ct}_m) \rightarrow m/\perp$*: On input the public parameter* $\mathsf{pp}_{\mathsf{upke}}$*, the secret key* $\mathsf{sk}_i^{(\mathsf{u})}$ *and a ciphertext $\mathsf{ct}_m \in \mathcal{CS}$, the decrypter runs this algorithm and returns plaintext m or $\perp$ to indicate description failure.*

$\mathsf{UPKE.UpdatePk}(\mathsf{pp}_{\mathsf{upke}}, \mathsf{pk}_i^{(\mathsf{u})}; \boldsymbol{\rho}_i) \rightarrow (\mathsf{pk}_{i+1}^{(\mathsf{u})}, \mathsf{up}_{i+1})$*: Given the public parameter* $\mathsf{pp}_{\mathsf{upke}}$*, a public key* $\mathsf{pk}_i^{(\mathsf{u})}$ *and a random $\boldsymbol{\rho}_i \in \mathcal{RS}$, any user can run this algorithm and produce an updated ciphertext up_{i+1} and a new public key* $\mathsf{pk}_{i+1}^{(\mathsf{u})}$.

$\mathsf{UPKE.UpdateSk}(\mathsf{pp}_{\mathsf{upke}}, \mathsf{sk}_i^{(\mathsf{u})}, \mathsf{up}_{i+1}) \rightarrow \mathsf{sk}_{i+1}^{(\mathsf{u})}$*: Given the public parameter* $\mathsf{pp}_{\mathsf{upke}}$ *and an updated ciphertext up_{i+1}, a user with secret key* $\mathsf{sk}_i^{(\mathsf{u})}$ *runs this algorithm to generate the updated secret key* $\mathsf{sk}_{i+1}^{(\mathsf{u})}$.

Setup: The challenger $\mathcal{C}$ computes $\mathsf{pp}_{\mathsf{upke}} \leftarrow \mathsf{UPKE.Setup}(\lambda)$ and secret-public key pair $(\mathsf{sk}_0^{(\mathsf{u})}, \mathsf{pk}_0^{(\mathsf{u})}) \leftarrow \mathsf{UPKE.KeyGen}(\mathsf{pp}_{\mathsf{upke}})$. It forwards $\mathsf{pp}_{\mathsf{upke}}$ and $\mathsf{pk}_0^{(\mathsf{u})}$ to the adversary $\mathcal{A}$ while keeps $\mathsf{sk}_0^{(\mathsf{u})}$ secret to itself. It sets epoch $i = 0$.

Pre-challenge Query Phase: The adversary $\mathcal{A}$ issues polynomially many adaptive queries to the oracle $\mathcal{O}_{\mathsf{up}}(\cdot)$.

– $\mathcal{O}_{\mathsf{up}}^{\mathsf{UPKE}}(\boldsymbol{\rho}_i)$: Upon receiving a query on the $\boldsymbol{\rho}_i$, the challenger $\mathcal{C}$ performs $\mathsf{UPKE.UpdatePk}(\mathsf{pp}_{\mathsf{upke}}, \mathsf{pk}_i^{(\mathsf{u})}; \boldsymbol{\rho}_i) \rightarrow (\mathsf{pk}_{i+1}^{(\mathsf{u})}, \mathsf{up}_{i+1})$ and $\mathsf{UPKE.UpdateSk}(\mathsf{pp}_{\mathsf{upke}}, \mathsf{sk}_i^{(\mathsf{u})}, \mathsf{up}_{i+1}) \rightarrow \mathsf{sk}_{i+1}^{(\mathsf{u})}$ and increments the i to $i+1$.

Challenge Phase: After receiving (m_0^*, m_1^*) from the adversary $\mathcal{A}$, the challenger $\mathcal{C}$ uniformly samples $b \xleftarrow{\$} \{0,1\}$ and sends $\mathsf{ct}_{m_b^*} \leftarrow \mathsf{UPKE.Enc}(\mathsf{pp}_{\mathsf{upke}}, \mathsf{pk}_t^{(\mathsf{u})}, m_b^*)$ where t is the current epoch.

Post-challenge Query Phase: $\mathcal{A}$ is allowed to make $\mathcal{O}_{\mathsf{up}}^{\mathsf{UPKE}}(\cdot)$ queries as in post-challenge query phase.

Reveal Phase: The challenger $\mathcal{C}$ chooses uniformly random $\boldsymbol{\rho}^*$ and then computes $\mathsf{UPKE.UpdatePk}(\mathsf{pp}_{\mathsf{upke}}, \mathsf{pk}_{t'}^{(\mathsf{u})}; \boldsymbol{\rho}^*) \rightarrow (\mathsf{pk}^*, \mathsf{up}^*)$ and $\mathsf{UPKE.UpdateSk}(\mathsf{pp}_{\mathsf{upke}}, \mathsf{sk}_{t'}^{(\mathsf{u})}, \mathsf{up}^*) \rightarrow \mathsf{sk}^*$ where t' is the current epoch and sends $(\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{up}^*)$.

Guess Phase: The adversary $\mathcal{A}$ eventually submits a bit b' to the challenger $\mathcal{C}$ and if $b = b'$ then the experiment $\mathsf{Exp}_{\mathsf{UPKE},\mathcal{A}}^{\mathsf{IND\text{-}CR\text{-}CPA}}(\lambda)$ returns 1, otherwise, it returns 0.

Fig. 3 $\mathsf{Exp}_{\mathsf{UPKE},\mathcal{A}}^{\mathsf{IND\text{-}CR\text{-}CPA}}(\lambda)$: Indistinguishability under chosen-randomness chosen-plaintext attacks

Definition 22 (Correctness)**.** *Let* $\mathsf{pp}_{\mathsf{upke}} \leftarrow \mathsf{UPKE.Setup}(\lambda)$ *and* $(\mathsf{sk}_0, \mathsf{pk}_0) \leftarrow \mathsf{UPKE.KeyGen}(\mathsf{pp}_{\mathsf{upke}})$. *Define* $\mathsf{UPKE.UpdatePk}(\mathsf{pp}_{\mathsf{upke}}, \mathsf{pk}_{i-1}^{(\mathsf{u})}; \boldsymbol{\rho}_{i-1}) \rightarrow (\mathsf{pk}_i^{(\mathsf{u})}, \mathsf{up}_i)$ *and* $\mathsf{UPKE.UpdateSk}(\mathsf{pp}_{\mathsf{upke}}, \mathsf{sk}_{i-1}^{(\mathsf{u})}, \mathsf{up}_i) \rightarrow \mathsf{sk}_i^{(\mathsf{u})}$ *for* $i \leq \ell$ *and* $\ell \in \mathbb{N}$. *An* UPKE *scheme provides correctness if for any* $m \in \mathcal{MS}$ *and all* $i \leq \ell$*:* $\Pr[\mathsf{UPKE.Dec}(\mathsf{pp}_{\mathsf{upke}}, \mathsf{sk}_i^{(\mathsf{u})}, \mathsf{UPKE.Enc}(\mathsf{pp}_{\mathsf{upke}}, \mathsf{pk}_i^{(\mathsf{u})}, m)) = m] = 1$

Definition 23 (IND-CR-CPA)**.** *An updatable public key encryption scheme* UPKE *is secure against indistinguishability under chosen-randomness chosen-plaintext attacks* (IND-CR-CPA) *if the advantage* $\mathsf{Adv}_{\mathsf{UPKE},\mathcal{A}}^{\mathsf{IND\text{-}CR\text{-}CPA}}(\lambda)$ *of any PPT adversary* $\mathcal{A}$ *defined as* $\mathsf{Adv}_{\mathsf{UPKE},\mathcal{A}}^{\mathsf{IND\text{-}CR\text{-}CPA}}(\lambda) = \left| \Pr[\mathsf{Exp}_{\mathsf{UPKE},\mathcal{A}}^{\mathsf{IND\text{-}CR\text{-}CPA}}(\lambda) = 1] - \frac{1}{2} \right|$ *is negligible where* $\mathsf{Exp}_{\mathsf{UPKE},\mathcal{A}}^{\mathsf{IND\text{-}CR\text{-}CPA}}(\lambda)$ *is the experiment between an adversary* $\mathcal{A}$ *and a challenger* $\mathcal{C}$ *as depicted in Fig.3.*

Definition 24 (IND-CR-CCA)**.** *An updatable public key encryption scheme* UPKE *is secure against indistinguishability under chosen-randomness chosen-ciphertext attacks* (IND-CR-CCA) *if the advantage* $\mathsf{Adv}_{\mathsf{UPKE},\mathcal{A}}^{\mathsf{IND\text{-}CR\text{-}CCA}}(\lambda)$ *of any PPT adversary* $\mathcal{A}$ *defined as*

$$\mathsf{Adv}_{\mathsf{UPKE},\mathcal{A}}^{\mathsf{IND\text{-}CR\text{-}CCA}}(\lambda) = \left| \Pr[\mathsf{Exp}_{\mathsf{UPKE},\mathcal{A}}^{\mathsf{IND\text{-}CR\text{-}CCA}}(\lambda) = 1] - \frac{1}{2} \right|$$

Setup: The challenger $\mathcal{C}$ computes $\mathsf{pp}_{\mathsf{upke}} \leftarrow \mathsf{UPKE.Setup}(\lambda)$ and secret-public key pair $(\mathsf{sk}_0^{(\mathsf{u})}, \mathsf{pk}_0^{(\mathsf{u})}) \leftarrow \mathsf{UPKE.KeyGen}(\mathsf{pp}_{\mathsf{upke}})$. It forwards $\mathsf{pp}_{\mathsf{upke}}$ and $\mathsf{pk}_0^{(\mathsf{u})}$ to the adversary $\mathcal{A}$ while keeps $\mathsf{sk}_0^{(\mathsf{u})}$ secret to itself. It also sets epoch $i = 0$.

Pre-challenge Query Phase: The adversary $\mathcal{A}$ issues polynomially many adaptive queries to the oracle $\mathcal{O}_{\mathsf{up}}^{\mathsf{UPKE}}(\cdot)$ and $\mathcal{O}_{\mathsf{dec}}^{\mathsf{UPKE}}(\cdot)$.

- $\mathcal{O}_{\mathsf{up}}^{\mathsf{UPKE}}(\boldsymbol{\rho}_i)$: Upon receiving a query on the $\boldsymbol{\rho}_i$, the challenger $\mathcal{C}$ performs $\mathsf{UPKE.UpdatePk}(\mathsf{pp}_{\mathsf{upke}}, \mathsf{pk}_i^{(\mathsf{u})}; \boldsymbol{\rho}_i) \rightarrow (\mathsf{pk}_{i+1}^{(\mathsf{u})}, \mathsf{up}_{i+1})$ and $\mathsf{UPKE.UpdateSk}(\mathsf{pp}_{\mathsf{upke}}, \mathsf{sk}_i^{(\mathsf{u})}, \mathsf{up}_{i+1}) \rightarrow \mathsf{sk}_{i+1}^{(\mathsf{u})}$ and increments the i to $i+1$.
- $\mathcal{O}_{\mathsf{dec}}^{\mathsf{UPKE}}(\mathsf{ct}_m)$: Given a ciphertext ct_m returns $m = \mathsf{UPKE.Dec}(\mathsf{pp}_{\mathsf{upke}}, \mathsf{sk}_i^{(\mathsf{u})}, \mathsf{ct}_m)$ where $\mathsf{sk}_i^{(\mathsf{u})}$ is the secret key of the current epoch.

Challenge Phase: After receiving (m_0^*, m_1^*) from the adversary $\mathcal{A}$, the challenger $\mathcal{C}$ uniformly samples $b \xleftarrow{\$} \{0,1\}$ and sends $\mathsf{ct}_{m_b^*} \leftarrow \mathsf{UPKE.Enc}(\mathsf{pp}_{\mathsf{upke}}, \mathsf{pk}_t^{(\mathsf{u})}, m_b^*)$ where t is the current epoch.

Post challenge Query Phase: The adversary $\mathcal{A}$ issues polynomially many adaptive queries to the oracle $\mathcal{O}_{\mathsf{up}}^{\mathsf{UPKE}}(\cdot)$ and $\mathcal{O}_{\mathsf{dec}}^{\mathsf{UPKE}}(\cdot)$.

- $\mathcal{O}_{\mathsf{up}}^{\mathsf{UPKE}}(\boldsymbol{\rho}_i)$: Upon receiving a query on the $\boldsymbol{\rho}_i$, the challenger $\mathcal{C}$ performs $\mathsf{UPKE.UpdatePk}(\mathsf{pp}_{\mathsf{upke}}, \mathsf{pk}_i^{(\mathsf{u})}; \boldsymbol{\rho}_i) \rightarrow (\mathsf{pk}_{i+1}^{(\mathsf{u})}, \mathsf{up}_{i+1})$ and $\mathsf{UPKE.UpdateSk}(\mathsf{pp}_{\mathsf{upke}}, \mathsf{sk}_i^{(\mathsf{u})}, \mathsf{up}_{i+1}) \rightarrow \mathsf{sk}_{i+1}^{(\mathsf{u})}$ and increments the i to $i+1$.
- $\mathcal{O}_{\mathsf{dec}}^{\mathsf{UPKE}}(\mathsf{ct}_m)$: Given a ciphertext ct_m, if $(\mathsf{ct}_m = \mathsf{ct}_{m_b^*} \wedge \mathsf{pk}_i^{(\mathsf{u})} = \mathsf{pk}_t^{(\mathsf{u})})$ aborts, else returns $m = \mathsf{UPKE.Dec}(\mathsf{pp}_{\mathsf{upke}}, \mathsf{sk}_i^{(\mathsf{u})}, \mathsf{ct}_m)$ where $\mathsf{sk}_i^{(\mathsf{u})}$ is the secret key of the current epoch.

Reveal Phase: The challenger $\mathcal{C}$ chooses uniformly random $\boldsymbol{\rho}^*$ and then computes $\mathsf{UPKE.UpdatePk}(\mathsf{pp}_{\mathsf{upke}}, \mathsf{pk}_{t'}^{(\mathsf{u})}; \boldsymbol{\rho}^*) \rightarrow (\mathsf{pk}^*, \mathsf{up}^*)$ and $\mathsf{UPKE.UpdateSk}(\mathsf{pp}_{\mathsf{upke}}, \mathsf{sk}_{t'}^{(\mathsf{u})}, \mathsf{up}^*) \rightarrow \mathsf{sk}^*$ where t' is the current epoch and sends $(\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{up}^*)$.

Guess Phase: The adversary $\mathcal{A}$ eventually submits a bit b' to the challenger $\mathcal{C}$ and if $b = b'$ then the experiment $\mathsf{Exp}_{\mathsf{UPKE},\mathcal{A}}^{\mathsf{IND\text{-}CR\text{-}CCA}}(\lambda)$ returns 1, otherwise, it returns 0.

Fig. 4 $\mathsf{Exp}_{\mathsf{UPKE},\mathcal{A}}^{\mathsf{IND\text{-}CR\text{-}CCA}}(\lambda)$: Indistinguishability under chosen-randomness chosen-ciphertext attacks

is negligible where $\mathsf{Exp}_{\mathsf{UPKE},\mathcal{A}}^{\mathsf{IND\text{-}CR\text{-}CCA}}(\lambda)$ *is the experiment between an adversary* $\mathcal{A}$ *and a challenger* $\mathcal{C}$ *is depicted in Fig.4.*

2.3 Updatable Key Encapsulation Mechanism

Definition 25 (Updatable Key Encapsulation Mechanism)**.** *An Updatable Key Encapsulation Mechanism* (UKEM) *is a tuple of PPT algorithms* $\mathsf{UKEM} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps}, \mathsf{UpdatePk}, \mathsf{UpdateSk})$ *associated with a secret key space* $\mathcal{KS}$*, a message space* $\mathcal{MS}$*, a ciphertext space* $\mathcal{CS}$ *and a key space* $\mathcal{K}\boldsymbol{ey}$ *satisfying the following requirements*:

$\mathsf{UKEM.Setup}(\lambda) \to \mathsf{pp}_{\mathsf{ukem}}$: *A trusted party on input the security parameter* λ *outputs the public parameter* $\mathsf{pp}_{\mathsf{ukem}}$.

$\mathsf{UKEM.KeyGen}(\mathsf{pp}_{\mathsf{ukem}}) \to (\mathsf{sk}_0^{(\mathsf{u})}, \mathsf{pk}_0^{(\mathsf{u})})$: *On input the public parameter* $\mathsf{pp}_{\mathsf{ukem}}$*, a user generates a secret-public key pair* $(\mathsf{sk}_0^{(\mathsf{u})}, \mathsf{pk}_0^{(\mathsf{u})})$.

$\mathsf{UKEM.Encaps}(\mathsf{pp}_{\mathsf{ukem}}, \mathsf{pk}_i^{(\mathsf{u})}) \to (\mathsf{ct}_m, \mathsf{ek})$: *Taking as input the public parameter* $\mathsf{pp}_{\mathsf{ukem}}$*, public key* $\mathsf{pk}_i^{(\mathsf{u})}$*, this algorithm PPT returns an encapsulation* $\mathsf{hct} \in \mathcal{CS}$ *and a key* $\mathsf{ek} \in \mathcal{K}\boldsymbol{ey}$.

$\mathsf{UKEM.Decaps}(\mathsf{pp}_{\mathsf{ukem}}, \mathsf{sk}_i^{(\mathsf{u})}, \mathsf{hct}) \to \mathsf{ek}$: *On input the public parameter* $\mathsf{pp}_{\mathsf{ukem}}$*, the secret key the decrypter* $\mathsf{sk}_i^{(\mathsf{u})}$ *and a header ciphertext* hct*, this algorithm returns key* $\mathsf{ek} \in \mathcal{K}\boldsymbol{ey}$.

$\mathsf{UKEM.UpdatePk}(\mathsf{pp}_{\mathsf{ukem}}, \mathsf{pk}_i^{(\mathsf{u})}; \boldsymbol{\rho}_i) \to (\mathsf{pk}_{i+1}^{(\mathsf{u})}, \mathsf{up}_{i+1})$: *Given the public parameter* $\mathsf{pp}_{\mathsf{ukem}}$*, a public key* $\mathsf{pk}_i^{(\mathsf{u})}$ *and a random* $\boldsymbol{\rho}$*, any user can run this algorithm and produce an updated ciphertext* up_{i+1} *and a new public key* $\mathsf{pk}_{i+1}^{(\mathsf{u})}$.

$\mathsf{UKEM.UpdateSk}(\mathsf{pp}_{\mathsf{ukem}}, \mathsf{sk}_i^{(\mathsf{u})}, \mathsf{up}_{i+1}) \to \mathsf{sk}_{i+1}^{(\mathsf{u})}$: *Given the public parameter* $\mathsf{pp}_{\mathsf{ukem}}$ *and an updated ciphertext* up_{i+1} *an user with* $\mathsf{sk}^{(\mathsf{u})}$ *runs this algorithm to generate updated secret key* $\mathsf{sk}_{i+1}^{(\mathsf{u})}$.

Definition 26 (Correctness)**.** *Let* $\mathsf{pp}_{\mathsf{ukem}} \leftarrow \mathsf{UPKE.Setup}(\lambda)$*, all* $(\mathsf{sk}_0^{(\mathsf{u})}, \mathsf{pk}_0^{(\mathsf{u})}) \leftarrow \mathsf{UKEM.KeyGen}(\mathsf{pp}_{\mathsf{ukem}})$*. Let* $(\mathsf{pk}_i^{(\mathsf{u})}, \mathsf{up}_i)$ *is generated by running* $\mathsf{UKEM.UpdatePk}(\mathsf{pp}_{\mathsf{ukem}}, \mathsf{pk}_{i-1}^{(\mathsf{u})}; \boldsymbol{\rho}_{i-1})$ *and* $\mathsf{sk}_i^{(\mathsf{u})} \leftarrow \mathsf{UKEM.UpdateSk}(\mathsf{pp}_{\mathsf{ukem}}, \mathsf{sk}_{i-1}^{(\mathsf{u})}, \mathsf{up}_i)$ *for* $i \leq \ell$ *and* $\ell \in \mathbb{N}$ *. An* UKEM *scheme provides correctness if for* $\mathsf{UKEM.Encaps}(\mathsf{pp}_{\mathsf{ukem}}, \mathsf{pk}_i^{(\mathsf{u})}) \to (\mathsf{hct}, \mathsf{ek})$ *and all* $i \leq \ell$: $\Pr[\mathsf{UKEM.Decaps}(\mathsf{pp}_{\mathsf{ukem}}, \mathsf{sk}_i^{(\mathsf{u})}, \mathsf{hct}) = \mathsf{ek}] = 1$.

Definition 27 (IND-CR-CPA)**.** *An updatable key encapsulation mechanism scheme* UKEM *is secure against indistinguishability under chosen-randomness chosen-plaintext attacks* (IND-CR-CCA) *if the advantage* $\mathsf{Adv}^{\mathsf{IND\text{-}CR\text{-}CCA}}_{\mathsf{UKEM},\mathcal{A}}(\lambda)$ *of any PPT adversary* $\mathcal{A}$ *defined as*

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

is negligible where the experiment $\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{UKEM},\mathcal{A}}(\lambda)$ *between an adversary* $\mathcal{A}$ *and a challenger* $\mathcal{C}$ *is depicted in Fig. 5.*

Definition 28 (IND-CR-CCA)**.** *An updatable key encapsulation mechanism scheme* UKEM *is secure against indistinguishability under chosen-randomness chosen-ciphertext attacks* (IND-CR-CCA) *if the advantage* $\mathsf{Adv}^{\mathsf{IND\text{-}CR\text{-}CCA}}_{\mathsf{UKEM},\mathcal{A}}(\lambda)$ *of any PPT*

Setup: The challenger $\mathcal{C}$ computes public parameter $\mathsf{pp}_{\mathsf{ukem}} \leftarrow \mathsf{UKEM.Setup}(\lambda)$ and secret-public key pair $(\mathsf{sk}_0^{(\mathsf{u})}, \mathsf{pk}_0^{(\mathsf{u})}) \leftarrow \mathsf{UKEM.KeyGen}(\mathsf{pp}_{\mathsf{ukem}})$. It forwards $\mathsf{pp}_{\mathsf{ukem}}$ and $\mathsf{pk}_0^{(\mathsf{u})}$ to the adversary $\mathcal{A}$ while keeps $\mathsf{sk}^{(\mathsf{u})}$ secret to itself. It sets epoch $i = 0$.

Pre-challenge Query Phase: The adversary $\mathcal{A}$ issues polynomially many adaptive queries to the oracle $\mathcal{O}_{\mathsf{up}}^{\mathsf{UKEM}}(\cdot)$.

– $\mathcal{O}_{\mathsf{up}}^{\mathsf{UKEM}}(\boldsymbol{\rho}_i)$: Upon receiving a query on the $\boldsymbol{\rho}_i$, the challenger $\mathcal{C}$ performs $\mathsf{UKEM.UpdatePk}(\mathsf{pp}_{\mathsf{ukem}}, \mathsf{pk}_i^{(\mathsf{u})}; \boldsymbol{\rho}_i) \rightarrow (\mathsf{pk}_{i+1}^{(\mathsf{u})}, \mathsf{up}_{i+1})$ and $\mathsf{UKEM.UpdateSk}(\mathsf{pp}_{\mathsf{ukem}}, \mathsf{sk}_i^{(\mathsf{u})}, \mathsf{up}_{i+1}) \rightarrow \mathsf{sk}_{i+1}^{(\mathsf{u})}$ and increments the i to $i + 1$.

Challenge Phase: The challenger $\mathcal{C}$ sets $\mathsf{ek}_0^* \xleftarrow{\$} \mathcal{K}\mathbf{ey}$ and $(\mathsf{hct}^*, \mathsf{ek}_1^*) \leftarrow \mathsf{UKEM.Encaps}(\mathsf{pp}_{\mathsf{ukem}}, \mathsf{pk}_t^{(\mathsf{u})})$. Then, it uniformly samples $b \xleftarrow{\$} \{0, 1\}$ and sends $(\mathsf{hct}^*, \mathsf{ek}_b^*)$ to $\mathcal{A}$ where t is the current epoch.

Post-challenge Query Phase: $\mathcal{A}$ is allowed to make $\mathcal{O}_{\mathsf{up}}^{\mathsf{UKEM}}(\cdot)$ queries as in post-challenge query phase.

Reveal Phase: The challenger $\mathcal{C}$ chooses uniformly random $\boldsymbol{\rho}^*$ and then computes $\mathsf{UKEM.UpdatePk}(\mathsf{pp}_{\mathsf{ukem}}, \mathsf{pk}_{t'}^{(\mathsf{u})}; \boldsymbol{\rho}^*) \rightarrow (\mathsf{pk}^*, \mathsf{up}^*)$ and $\mathsf{UKEM.UpdateSk}(\mathsf{pp}_{\mathsf{ukem}}, \mathsf{sk}_{t'}^{(\mathsf{u})}, \mathsf{up}^*) \rightarrow \mathsf{sk}^*$ where t' is the current epoch and sends $(\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{up}^*)$.

Guess Phase: The adversary $\mathcal{A}$ eventually submits a bit b' to the challenger $\mathcal{C}$ and if $b = b'$ then the experiment $\mathsf{Exp}_{\mathsf{UKEM},\mathcal{A}}^{\mathsf{IND\text{-}CR\text{-}CPA}}(\lambda)$ returns 1, otherwise, it returns 0.

Fig. 5 $\mathsf{Exp}_{\mathsf{UKEM},\mathcal{A}}^{\mathsf{IND\text{-}CR\text{-}CPA}}(\lambda)$: Indistinguishability under chosen-randomness chosen-ciphertext attacks

adversary $\mathcal{A}$ defined as

$$\mathsf{Adv}_{\mathsf{UKEM},\mathcal{A}}^{\mathsf{IND\text{-}CR\text{-}CCA}}(\lambda) = \left| \Pr[\mathsf{Exp}_{\mathsf{UKEM},\mathcal{A}}^{\mathsf{IND\text{-}CR\text{-}CCA}}(\lambda) = 1] - \frac{1}{2} \right|$$

is negligible where the experiment $\mathsf{Exp}_{\mathsf{UKEM},\mathcal{A}}^{\mathsf{IND\text{-}CR\text{-}CCA}}(\lambda)$ between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ is depicted in Fig. 6.

Setup: The challenger $\mathcal{C}$ computes public parameter $\mathsf{pp}_{\mathsf{ukem}} \leftarrow \mathsf{UKEM.Setup}(\lambda)$ and secret-public key pair $(\mathsf{sk}_0^{(\mathsf{u})}, \mathsf{pk}_0^{(\mathsf{u})}) \leftarrow \mathsf{UKEM.KeyGen}(\mathsf{pp}_{\mathsf{ukem}})$. It forwards $\mathsf{pp}_{\mathsf{ukem}}$ and $\mathsf{pk}_0^{(\mathsf{u})}$ to the adversary $\mathcal{A}$ while keeps $\mathsf{sk}_0^{(\mathsf{u})}$ secret to itself. It sets epoch $i = 0$.

Pre-challenge Query Phase: The adversary $\mathcal{A}$ issues polynomially many adaptive queries to the oracle $\mathcal{O}_{\mathsf{up}}^{\mathsf{UKEM}}(\cdot)$ and $\mathcal{O}_{\mathsf{dec}}^{\mathsf{UKEM}}(\cdot)$.

– $\mathcal{O}_{\mathsf{up}}^{\mathsf{UKEM}}(\boldsymbol{\rho}_i)$: Upon receiving a query on the r_i, the challenger $\mathcal{C}$ performs $\mathsf{UKEM.UpdatePk}(\mathsf{pp}_{\mathsf{ukem}}, \mathsf{pk}_i^{(\mathsf{u})}; \boldsymbol{\rho}_i) \rightarrow (\mathsf{pk}_{i+1}^{(\mathsf{u})}, \mathsf{up}_{i+1})$ and $\mathsf{UKEM.UpdateSk}(\mathsf{pp}_{\mathsf{ukem}}, \mathsf{sk}_i^{(\mathsf{u})}, \mathsf{up}_{i+1}) \rightarrow \mathsf{sk}_{i+1}^{(\mathsf{u})}$ and increments i to $i+1$.

– $\mathcal{O}_{\mathsf{dec}}^{\mathsf{UKEM}}(\mathsf{ct}_m)$: On input ct_m from $\mathcal{A}$, the challenger $\mathcal{C}$ returns $\mathsf{ek} = \mathsf{UKEM.Dec}(\mathsf{pp}_{\mathsf{ukem}}, \mathsf{sk}_i^{(\mathsf{u})}, \mathsf{ct}_m)$ where $\mathsf{sk}_i^{(\mathsf{u})}$ is the secret key of the current epoch.

Challenge Phase: The challenger $\mathcal{C}$ sets $\mathsf{ek}_0^* \xleftarrow{\$} \mathcal{K}\mathbf{ey}$, $(\mathsf{hct}^*, \mathsf{ek}_1^*) \leftarrow \mathsf{UKEM.Encaps}(\mathsf{pp}_{\mathsf{ukem}}, \mathsf{pk}_t^{(\mathsf{u})})$. Then, it uniformly samples $b \xleftarrow{\$} \{0,1\}$ and sends $(\mathsf{hct}^*, \mathsf{ek}_b^*)$ to $\mathcal{A}$ where t is the current epoch.

Post-challenge Query Phase: The adversary $\mathcal{A}$ issues polynomially many adaptive queries to the oracle $\mathcal{O}_{\mathsf{up}}^{\mathsf{UKEM}}(\cdot)$ and $\mathcal{O}_{\mathsf{dec}}^{\mathsf{UKEM}}(\cdot)$.

– $\mathcal{O}_{\mathsf{up}}^{\mathsf{UKEM}}(\boldsymbol{\rho}_i)$: Upon receiving a query on the r_i, the challenger $\mathcal{C}$ performs $\mathsf{UKEM.UpdatePk}(\mathsf{pp}_{\mathsf{ukem}}, \mathsf{pk}_i^{(\mathsf{u})}; \boldsymbol{\rho}_i) \rightarrow (\mathsf{pk}_{i+1}^{(\mathsf{u})}, \mathsf{up}_{i+1})$ and $\mathsf{UKEM.UpdateSk}(\mathsf{pp}_{\mathsf{ukem}}, \mathsf{sk}_i^{(\mathsf{u})}, \mathsf{up}_{i+1}) \rightarrow \mathsf{sk}_{i+1}^{(\mathsf{u})}$ and increments i to $i+1$.

– $\mathcal{O}_{\mathsf{dec}}^{\mathsf{UKEM}}(\mathsf{hct})$: On input hct from $\mathcal{A}$ if $\mathsf{hct} = \mathsf{hct}^*$ and $\mathsf{pk}_i^{(\mathsf{u})} = \mathsf{pk}_t^{(\mathsf{u})}$ then the challenger $\mathcal{C}$ returns abort, else returns $\mathsf{ek} = \mathsf{UKEM.Dec}(\mathsf{pp}_{\mathsf{ukem}}, \mathsf{sk}_i^{(\mathsf{u})}, \mathsf{hct})$ where $\mathsf{sk}_i^{(\mathsf{u})}$ is the secret key of the current epoch.

Reveal Phase: The challenger $\mathcal{C}$ chooses uniformly random $\boldsymbol{\rho}^*$ and then computes $\mathsf{UKEM.UpdatePk}(\mathsf{pp}_{\mathsf{ukem}}, \mathsf{pk}_{t'}^{(\mathsf{u})}; \boldsymbol{\rho}^*) \rightarrow (\mathsf{pk}^*, \mathsf{up}^*)$ and $\mathsf{UKEM.UpdateSk}(\mathsf{pp}_{\mathsf{ukem}}, \mathsf{sk}_{t'}^{(\mathsf{u})}, \mathsf{up}^*) \rightarrow \mathsf{sk}^*$ where t' is the current epoch and sends $(\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{up}^*)$.

Guess Phase: The adversary $\mathcal{A}$ eventually submits a bit b' to the challenger $\mathcal{C}$ and if $b = b'$ then the experiment $\mathsf{Exp}_{\mathsf{UKEM},\mathcal{A}}^{\mathsf{IND\text{-}CR\text{-}CCA}}(\lambda)$ returns 1, otherwise, it returns 0.

Fig. 6 $\mathsf{Exp}_{\mathsf{UKEM},\mathcal{A}}^{\mathsf{IND\text{-}CR\text{-}CCA}}(\lambda)$: Indistinguishability under chosen-randomness chosen-ciphertext attacks

3 Security Proof of Theorem 38

In this section, we establish the f-CS + LR security of hPKE, which is essential for proving the security of the updatable public key encryption scheme.

Theorem 29. *The construction* hPKE *provides* f-CS + LR *security for* $f : [-\mu, \mu]^n \times \{0,1\}^{\mathsf{mlen}(\lambda)} \rightarrow [-\mu, \mu]^n$ *as per Definition 20 under the* CSSIDDH *assumption given in Definition 13, assuming* $\mathcal{H} = \{H_k\}_{k \in \mathcal{K}}$ *is entropy smoothing function and* KDF *is a secure key derivation function as per Definition 16.*

Proof Consider a key derivation function $\mathsf{KDF} : \mathcal{MS} = \{0,1\}^{\mathsf{mlen}(\lambda)} \to \mathcal{KS} = [-\mu,\mu]^n$. We prove hPKE is f-$\mathsf{CS}+\mathsf{LR}$ secure for $f(\mathsf{sk},\boldsymbol{\rho}) = \mathsf{sk} - \mathsf{KDF}(\boldsymbol{\rho})$ using a sequence of games. The Changes in the adversary's view across the sequence of games are described in Fig. 4. The changes are highlighted by enclosing them in a box. We define S_i as the event that $b = b'$ in Game_i for $i = 0,\ldots,6$.

$\underline{\mathsf{Game}_0}$*:* This is the original f-$\mathsf{CS}+\mathsf{LR}$ security game $\mathsf{Exp}^{f\text{-}\mathsf{CS}+\mathsf{LR}}_{\mathsf{PKE},\mathcal{A}}(\lambda)$ between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ works as follows:

Setup: The challenger $\mathcal{C}$ generates the public parameter $\mathsf{pp}_{\mathsf{hPKE}} = (p,\ell_1,\ldots,\ell_n,\mu,E_0,\mathcal{H} = \{H_k\}_{k\in\mathcal{K}}) \leftarrow \mathsf{hPKE.Setup}(\lambda)$ and a secret-public key $(\mathsf{sk} = \boldsymbol{a} \in [-\mu,\mu]^n,\ \mathsf{pk} = E_A = [\boldsymbol{a}]E_0) \leftarrow \mathsf{hPKE.KeyGen}(\mathsf{pp})$. The challenger $\mathcal{C}$ sends $\mathsf{pp}_{\mathsf{hPKE}}$ and $\mathsf{pk} = \mathsf{pk}^{\mathsf{G}_0}$ to $\mathcal{A}$ and keeps sk secret to itself.

Challenge Phase: The adversary $\mathcal{A}$ sends $(m_0^*,m_1^*) \in \{0,1\}^{\mathsf{mlen}} \times \{0,1\}^{\mathsf{mlen}}$ as challenge message pair to $\mathcal{C}$. Then $\mathcal{C}$ samples $\boldsymbol{\rho}^* \xleftarrow{\$} \{0,1\}^{\mathsf{mlen}(\lambda)}$ and $b \xleftarrow{\$} \{0,1\}$, computes $\mathbf{r}^* = \mathsf{KDF}(\boldsymbol{\rho}^*) \in [-\mu,\mu]^n$ and sends $\mathbf{z} = f(\mathsf{sk},\boldsymbol{\rho}^*) = \mathsf{sk} - \mathbf{r}^* = \boldsymbol{a} - \mathbf{r}^*$, $\mathsf{ct}^{\mathsf{G}_0}_{m_b^*} \leftarrow \mathsf{hPKE.Enc}(\mathsf{pp}_{\mathsf{hPKE}},\mathsf{pk},m_b^*)$ with $\mathsf{ct}^{\mathsf{G}_0}_{m_b^*} = (\mathsf{ct}^{\mathsf{G}_0,1}_{m_b^*} = [\mathbf{b}]E_0,\ \mathsf{ct}^{\mathsf{G}_0,2}_{m_b^*} = m_b^* \oplus H_k(M_C([\mathbf{b}]E_A)))$ and $\mathsf{ct}^{\mathsf{G}_0}_{\boldsymbol{\rho}^*} \leftarrow \mathsf{hPKE.Enc}(\mathsf{pp}_{\mathsf{hPKE}},\mathsf{pk},\boldsymbol{\rho}^*)$ with $\mathsf{ct}^{\mathsf{G}_0}_{\boldsymbol{\rho}^*} = (\mathsf{ct}^{\mathsf{G}_0,1}_{\boldsymbol{\rho}^*} = [\mathbf{b}']E_0,\ \mathsf{ct}^{\mathsf{G}_0,2}_{\boldsymbol{\rho}^*} = \boldsymbol{\rho}^* \oplus H_k(M_C([\mathbf{b}']E_A)))$ to $\mathcal{A}$ where $\mathbf{b},\mathbf{b}' \xleftarrow{\$} [-\mu,\mu]^n$.

Guess Phase: The adversary $\mathcal{A}$ eventually submits a bit b' to the challenger $\mathcal{C}$ and if $b = b'$ then the experiment $\mathsf{Exp}^{f\text{-}\mathsf{CS}+\mathsf{LR}}_{\mathsf{hPKE},\mathcal{A}}(\lambda)$ returns 1, otherwise, it returns 0.

$\underline{\mathsf{Game}_1}$*:* In this game, we modify the generation of the public key and m_b^* and $\boldsymbol{\rho}^*$ are encrypted using the updated public key.

Setup: The challenger $\mathcal{C}$ generates the public parameter $\mathsf{pp}_{\mathsf{hPKE}} = (p,\ell_1,\ldots,\ell_n,\mu,E_0,\mathcal{H} = \{H_k\}_{k\in\mathcal{K}})$ and samples $\boldsymbol{\rho}^* \xleftarrow{\$} \{0,1\}^{\mathsf{mlen}(\lambda)}$, computes $\mathbf{r}^* = \mathsf{KDF}(\boldsymbol{\rho}^*) \in [-\mu,\mu]^n$. The challenger $\mathcal{C}$ sends $\mathsf{pp}_{\mathsf{hPKE}}$ and $\boxed{\mathsf{pk}^{\mathsf{G}_1} = [\mathbf{r}^*]E_0}$ to $\mathcal{A}$.

Challenge Phase: The adversary $\mathcal{A}$ also sends (m_0^*,m_1^*) as a challenge message to $\mathcal{C}$. Then $\mathcal{C}$ uniformly samples $b \xleftarrow{\$} \{0,1\}$ and sends $\mathbf{z} = \mathsf{sk} - \mathbf{r}^* = \boldsymbol{a} - \mathbf{r}^*$, $\mathsf{ct}^{\mathsf{G}_1}_{m_b^*} \leftarrow \mathsf{hPKE.Enc}(\mathsf{pp}_{\mathsf{hPKE}},\mathsf{pk}^{\mathsf{G}_1},m_b^*)$ with $\mathsf{ct}^{\mathsf{G}_1}_{m_b^*} = (\mathsf{ct}^{\mathsf{G}_1,1}_{m_b^*} = [\mathbf{b}]E_0,\ \boxed{\mathsf{ct}^{\mathsf{G}_1,2}_{m_b^*} = m_b^* \oplus H_k(M_C([\mathbf{b}][\mathbf{r}^*]E_0))})$ and $\mathsf{ct}^{\mathsf{G}_1}_{\boldsymbol{\rho}^*} = \mathsf{hPKE.Enc}(\mathsf{pp}_{\mathsf{hPKE}},\mathsf{pk}^{\mathsf{G}_1},\boldsymbol{\rho}^*)$ with $\mathsf{ct}^{\mathsf{G}_1}_{\boldsymbol{\rho}^*} = (\mathsf{ct}^{\mathsf{G}_1,1}_{\boldsymbol{\rho}^*} = [\mathbf{b}']E_0,\ \boxed{\mathsf{ct}^{\mathsf{G}_1,2}_{\boldsymbol{\rho}^*} = \boldsymbol{\rho}^* \oplus H_k(M_C([\mathbf{b}'][\mathbf{r}^*]E_0))})$ to $\mathcal{A}$ where $\mathbf{b},\mathbf{b}' \xleftarrow{\$} [-\mu,\mu]^n$.

Guess Phase: The adversary $\mathcal{A}$ eventually submits a bit b' to the challenger $\mathcal{C}$ and if $b = b'$ then the experiment $\mathsf{Exp}^{f\text{-}\mathsf{CS}+\mathsf{LR}}_{\mathsf{hPKE},\mathcal{A}}(\lambda)$ returns 1, otherwise, it returns 0.

Note that in the setup phase, we change $\mathsf{pk}^{\mathsf{G}_0} = [\mathsf{sk}]E_0$ in Game_0 to $\mathsf{pk}^{\mathsf{G}_1} = [\mathbf{r}^*]E_0$ in Game_1 and in challenge phase $\mathsf{ct}^{\mathsf{G}_0}_{m_b^*} = (\mathsf{ct}^{\mathsf{G}_0,1}_{m_b^*} = [\mathbf{b}]E_0,\ \mathsf{ct}^{\mathsf{G}_0,2}_{m_b^*} = m_b^* \oplus H_k(M_C([\mathbf{b}]E_A))), \mathsf{ct}^{\mathsf{G}_0}_{\boldsymbol{\rho}^*} = (\mathsf{ct}^{\mathsf{G}_0,1}_{\boldsymbol{\rho}^*} = [\mathbf{b}']E_0,\ \mathsf{ct}^{\mathsf{G}_0,2}_{\boldsymbol{\rho}^*} = \boldsymbol{\rho}^* \oplus H_k(M_C([\mathbf{b}']E_A)))$ in Game_0 are replaced to $\mathsf{ct}^{\mathsf{G}_1}_{m_b^*} = (\mathsf{ct}^{\mathsf{G}_1,1}_{m_b^*} = [\mathbf{b}]E_0,\ \mathsf{ct}^{\mathsf{G}_1,2}_{m_b^*} = m_b^* \oplus H_k(M_C([\mathbf{b}][\mathbf{r}^*]E_0))), \mathsf{ct}^{\mathsf{G}_1}_{\boldsymbol{\rho}^*} = (\mathsf{ct}^{\mathsf{G}_1,1}_{\boldsymbol{\rho}^*} = [\mathbf{b}']E_0,\ \mathsf{ct}^{\mathsf{G}_1,2}_{\boldsymbol{\rho}^*} = \boldsymbol{\rho}^* \oplus H_k(M_C([\mathbf{b}'][\mathbf{r}^*]E_0)))$ in Game_1. As $\mathsf{pk}^{\mathsf{G}_0}$ and $\mathsf{pk}^{\mathsf{G}_1}$ are identically distributed, the adversary's view in Game_0 and Game_1 is indistinguishable. Consequently, we have $|\Pr[S_0] - \Pr[S_1]| \le \epsilon_1(\lambda)$ where $\epsilon_1(\lambda)$ negligible function in λ.

Claim 1. $|\Pr[S_0] - \Pr[S_1]| \le \epsilon_1$

Game	Transition of the game
Game_0	The original f-CS + LR security game $\mathsf{pp}_{\mathsf{hPKE}} = (p, \ell_1, \ldots, \ell_n, \mu,\ E_0, \mathcal{H} = \{H_k\}_{k\in\mathcal{K}}), \mathsf{pk}^{\mathsf{G}_0} = [\mathsf{sk}]E_0,\ \mathbf{z} = \mathsf{sk} - \mathbf{r}^*,$ $\mathsf{ct}^{\mathsf{G}_0}_{m_b^*} = (\mathsf{ct}^{\mathsf{G}_0,1}_{m_b^*} = [\mathbf{b}]E_0,\ \mathsf{ct}^{\mathsf{G}_0,2}_{m_b^*} = m_b^* \oplus H_k(M_C([\mathbf{b}]E_A))),$ $\mathsf{ct}^{\mathsf{G}_0}_{\boldsymbol{\rho}^*} = (\mathsf{ct}^{\mathsf{G}_0,1}_{\boldsymbol{\rho}^*} = [\mathbf{b}']E_0,\ \mathsf{ct}^{\mathsf{G}_0,2}_{\boldsymbol{\rho}^*} = \boldsymbol{\rho}^* \oplus H_k(M_C([\mathbf{b}']E_A)))$ for $\mathbf{r}^* = \mathsf{KDF}(\boldsymbol{\rho}^*), \mathbf{b}, \mathbf{b}' \xleftarrow{\$} [-\mu, \mu]^n, \boldsymbol{\rho}^* \xleftarrow{\$} \{0,1\}^{\mathsf{mlen}(\lambda)}$
Game_1	$\mathsf{pp}_{\mathsf{hPKE}} = (p, \ell_1, \ldots, \ell_n, \mu,\ E_0, \mathcal{H} = \{H_k\}_{k\in\mathcal{K}}), \boxed{\mathsf{pk}^{\mathsf{G}_1} = [\mathbf{r}^*]E_0},\ \mathbf{z} = \mathsf{sk} - \mathbf{r}^*,$ $\mathsf{ct}^{\mathsf{G}_1}_{m_b^*} = (\mathsf{ct}^{\mathsf{G}_1,1}_{m_b^*} = [\mathbf{b}]E_0,\ \boxed{\mathsf{ct}^{\mathsf{G}_1,2}_{m_b^*} = m_b^* \oplus H_k(M_C([\mathbf{b}][\mathbf{r}^*]E_0))}),$ $\mathsf{ct}^{\mathsf{G}_1}_{\boldsymbol{\rho}^*} = (\mathsf{ct}^{\mathsf{G}_1,1}_{\boldsymbol{\rho}^*} = [\mathbf{b}']E_0,\ \boxed{\mathsf{ct}^{\mathsf{G}_1,2}_{\boldsymbol{\rho}^*} = \boldsymbol{\rho}^* \oplus H_k(M_C([\mathbf{b}'][\mathbf{r}^*]E_0))})$ for $\mathbf{b}, \mathbf{b}' \xleftarrow{\$} [-\mu, \mu]^n, \boldsymbol{\rho}^* \xleftarrow{\$} \{0,1\}^{\mathsf{mlen}(\lambda)}$
Game_2	$\mathsf{pp}_{\mathsf{hPKE}} = (p, \ell_1, \ldots, \ell_n, \mu,\ E_0, \mathcal{H} = \{H_k\}_{k\in\mathcal{K}}), \mathsf{pk}^{\mathsf{G}_2} = [\mathbf{r}^*]E_0,\ \boxed{\mathbf{u}},$ $\mathsf{ct}^{\mathsf{G}_2}_{m_b^*} = (\mathsf{ct}^{\mathsf{G}_2,1}_{m_b^*} = [\mathbf{b}]E_0,\ \mathsf{ct}^{\mathsf{G}_2,2}_{m_b^*} = m_b^* \oplus H_k(M_C([\mathbf{b}][\mathbf{r}^*]E_0))),$ $\mathsf{ct}^{\mathsf{G}_2}_{\boldsymbol{\rho}^*} = (\mathsf{ct}^{\mathsf{G}_2,1}_{\boldsymbol{\rho}^*} = [\mathbf{b}']E_0,\ \mathsf{ct}^{\mathsf{G}_2,2}_{\boldsymbol{\rho}^*} = \boldsymbol{\rho}^* \oplus H_k(M_C([\mathbf{b}'][\mathbf{r}^*]E_0)))$ for $\mathbf{u}, \mathbf{b}, \mathbf{b}' \xleftarrow{\$} [-\mu, \mu]^n, \boldsymbol{\rho}^* \xleftarrow{\$} \{0,1\}^{\mathsf{mlen}(\lambda)}$
Game_3	$\mathsf{pp}_{\mathsf{hPKE}} = (p, \ell_1, \ldots, \ell_n, \mu,\ E_0, \mathcal{H} = \{H_k\}_{k\in\mathcal{K}}), \mathsf{pk}^{\mathsf{G}_3} = [\mathbf{r}^*]E_0,\ \mathbf{u},$ $\mathsf{ct}^{\mathsf{G}_3}_{m_b^*} = (\mathsf{ct}^{\mathsf{G}_3,1}_{m_b^*} = [\mathbf{b}]E_0,\ \boxed{\mathsf{ct}^{\mathsf{G}_3,2}_{m_b^*} = m_b^* \oplus H_k(M_C([\tilde{\mathbf{b}}]E_0))}),$ $\mathsf{ct}^{\mathsf{G}_3}_{\boldsymbol{\rho}^*} = (\mathsf{ct}^{\mathsf{G}_3,1}_{\boldsymbol{\rho}^*} = [\mathbf{b}']E_0,\ \mathsf{ct}^{\mathsf{G}_3,2}_{\boldsymbol{\rho}^*} = \boldsymbol{\rho}^* \oplus H_k(M_C([\mathbf{b}'][\mathbf{r}^*]E_0)))$ for $\mathbf{u}, \mathbf{b}, \mathbf{b}', \tilde{\mathbf{b}} \xleftarrow{\$} [-\mu, \mu]^n, \boldsymbol{\rho}^* \xleftarrow{\$} \{0,1\}^{\mathsf{mlen}(\lambda)}$
Game_4	$\mathsf{pp}_{\mathsf{hPKE}} = (p, \ell_1, \ldots, \ell_n, \mu,\ E_0, \mathcal{H} = \{H_k\}_{k\in\mathcal{K}}), \mathsf{pk}^{\mathsf{G}_4} = [\mathbf{r}^*]E_0,\ \mathbf{u},$ $\mathsf{ct}^{\mathsf{G}_4}_{m_b^*} = (\mathsf{ct}^{\mathsf{G}_4,1}_{m_b^*} = [\mathbf{b}]E_0,\ \boxed{\mathsf{ct}^{\mathsf{G}_4,2}_{m_b^*} = m_b^* \oplus h_1}),$ $\mathsf{ct}^{\mathsf{G}_4}_{\boldsymbol{\rho}^*} = (\mathsf{ct}^{\mathsf{G}_4,1}_{\boldsymbol{\rho}^*} = [\mathbf{b}']E_0,\ \mathsf{ct}^{\mathsf{G}_4,2}_{\boldsymbol{\rho}^*} = \boldsymbol{\rho}^* \oplus H_k(M_C([\mathbf{b}'][\mathbf{r}^*]E_0)))$ for $\mathbf{u}, \mathbf{b}, \mathbf{b}' \xleftarrow{\$} [-\mu, \mu]^n, h_1 \xleftarrow{\$} \{0,1\}^{\mathsf{mlen}(\lambda)}$
Game_5	$\mathsf{pp}_{\mathsf{hPKE}} = (p, \ell_1, \ldots, \ell_n, \mu,\ E_0, \mathcal{H} = \{H_k\}_{k\in\mathcal{K}}), \mathsf{pk}^{\mathsf{G}_5} = [\mathbf{r}^*]E_0,\ \mathbf{u},$ $\mathsf{ct}^{\mathsf{G}_5}_{m_b^*} = (\mathsf{ct}^{\mathsf{G}_5,1}_{m_b^*} = [\mathbf{b}]E_0,\ \mathsf{ct}^{\mathsf{G}_5,2}_{m_b^*} = m_b^* \oplus h_1),$ $\mathsf{ct}^{\mathsf{G}_5}_{\boldsymbol{\rho}^*} = (\mathsf{ct}^{\mathsf{G}_5,1}_{\boldsymbol{\rho}^*} = [\mathbf{b}']E_0,\ \boxed{\mathsf{ct}^{\mathsf{G}_5,2}_{\boldsymbol{\rho}^*} = \boldsymbol{\rho}^* \oplus H_k(M_C([\tilde{\mathbf{b}'}]E_0))})$ for $\mathbf{u}, \mathbf{b}, \mathbf{b}', \tilde{\mathbf{b}'} \xleftarrow{\$} [-\mu, \mu]^n, h_1 \xleftarrow{\$} \{0,1\}^{\mathsf{mlen}(\lambda)}$
Game_6	$\mathsf{pp}_{\mathsf{hPKE}} = (p, \ell_1, \ldots, \ell_n, \mu,\ E_0, \mathcal{H} = \{H_k\}_{k\in\mathcal{K}}), \mathsf{pk}^{\mathsf{G}_6} = [\mathbf{r}^*]E_0,\ \mathbf{u},$ $\mathsf{ct}^{\mathsf{G}_6}_{m_b^*} = (\mathsf{ct}^{\mathsf{G}_6,1}_{m_b^*} = [\mathbf{b}]E_0,\ \mathsf{ct}^{\mathsf{G}_6,2}_{m_b^*} = m_b^* \oplus h_1),$ $\mathsf{ct}^{\mathsf{G}_6}_{\boldsymbol{\rho}^*} = (\mathsf{ct}^{\mathsf{G}_6,1}_{\boldsymbol{\rho}^*} = [\mathbf{b}']E_0,\ \boxed{\mathsf{ct}^{\mathsf{G}_6,2}_{\boldsymbol{\rho}^*} = \boldsymbol{\rho}^* \oplus h_2})$ for $\mathbf{u}, \mathbf{b}, \mathbf{b}' \xleftarrow{\$} [-\mu, \mu]^n, h_1, h_2 \xleftarrow{\$} \{0,1\}^{\mathsf{mlen}(\lambda)}$

Table 4 Changes in the adversary's view across the sequence of games.

$\underline{\mathsf{Game}_2}$: The Game_2 is identical to Game_1 except that the leakage $\mathbf{z} = f(\mathsf{sk}, \boldsymbol{\rho}^*) = \mathsf{sk} - \mathbf{r}^* \in [-\mu, \mu]^n$ where $\mathbf{r}^* = \mathsf{KDF}(\boldsymbol{\rho}^*)$ is replaced by a random element $\mathbf{u} \xleftarrow{\$} [-\mu, \mu]^n$ in the challenge phase.
Since $\mathsf{pk}^{\mathsf{G}_2} = \mathsf{pk}^{\mathsf{G}_1} = [\mathbf{r}^*]E_0,\ \mathsf{ct}^{\mathsf{G}_2}_{m_b^*} = \mathsf{ct}^{\mathsf{G}_1}_{m_b^*} = ([\mathbf{b}]E_0,\ m_b^* \oplus H_k(M_C([\mathbf{b}][\mathbf{r}^*]E_0)))$ and $\mathsf{ct}^{\mathsf{G}_2}_{\boldsymbol{\rho}^*} = ([\mathbf{b}']E_0,\ \boldsymbol{\rho}^* \oplus H_k(M_C([\mathbf{b}'][\mathbf{r}^*]E_0)))$ doe not contain any information about sk do not depend on sk and $\mathbf{z} = \mathsf{sk} - \mathbf{r}^*$ in Game_1 and $\mathbf{u}$ in Game_2 both are uniformly random in $[-\mu, \mu]^n$, it follows that Game_1 and Game_2 are statistically indistinguishable from adversaries point of view. Hence, $|\Pr[S_1] - \Pr[S_2]| \leq \epsilon_2(\lambda)$ where $\epsilon_2(\lambda)$ negligible function in λ.

Claim 2. $|\Pr[S_1] - \Pr[S_2]| \leq \epsilon_2(\lambda)$ *where* $\epsilon_2(\lambda)$ *negligible function in* λ.

$\underline{\mathsf{Game}_3}$: The game Game_2 is transformed into Game_3 by replacing

$$\mathsf{ct}^{\mathsf{G}_2}_{m_b^*} = (\mathsf{ct}^{\mathsf{G}_2,1}_{m_b^*} = [\mathbf{b}]E_0,\ \mathsf{ct}^{\mathsf{G}_2,2}_{m_b^*} = m_b^* \oplus H_k(M_C([\mathbf{b}][\mathbf{r}^*]E_0)))$$

in Game_2 to

$$\mathsf{ct}^{\mathsf{G}_3}_{m_b^*} = (\mathsf{ct}^{\mathsf{G}_3,1}_{m_b^*} = [\mathbf{b}]E_0,\ \boxed{\mathsf{ct}^{\mathsf{G}_3,2}_{m_b^*} = m_b^* \oplus H_k(M_C([\tilde{\mathbf{b}}]E_0))})$$

in Game_3 in challenge phase where $\mathbf{b}, \tilde{\mathbf{b}} \xleftarrow{\$} [-\mu, \mu]^n$.

Claim 3. $|\Pr[S_2] - \Pr[S_3]| \leq \epsilon_3$ *where* $\epsilon_3(\lambda)$ *negligible function in* λ.

Proof We show below that if the adversary $\mathcal{A}$ can distinguish between Game_3 and Game_2 then we can construct a distinguisher $\mathcal{D}$ for the CSSIDDH problem. Let the CSSIDDH challenger $\mathcal{C}$ provides an instance of CSSIDDH problem $\mathsf{IN} = (E_0, [\boldsymbol{\alpha}]E_0, [\boldsymbol{\beta}]E_0, [\boldsymbol{\gamma}]E_0)$ to the distinguisher $\mathcal{D}$.

Setup: The distinguisher $\mathcal{D}$ generates the public parameter $\mathsf{pp}_{\mathsf{hPKE}} = (p, \ell_1, \ldots, \ell_n, \mu, E_0, \mathcal{H} = \{H_k\}_{k \in \mathcal{K}}) \leftarrow \mathsf{hPKE.Setup}(\lambda)$ and sets $\mathsf{pk} = [\boldsymbol{\alpha}]E_0$. It sends $\mathsf{pp}_{\mathsf{hPKE}}$ along with pk to the adversary $\mathcal{A}$.

Challenge Phase: After receiving the challenge plaintext pair (m_0^*, m_1^*) from $\mathcal{A}$, the distinguisher $\mathcal{D}$ Uniformly samples $b \xleftarrow{\$} \{0,1\}$, computes $\mathsf{ct}^{(1)}_{m_b^*} = [\boldsymbol{\beta}]E_0,\ \mathsf{ct}^{(2)}_{m_b^*} = m_b^* \oplus H_k(M_C([\boldsymbol{\gamma}]E_0))$ and sends the ciphertext $\mathsf{ct}_{m_b^*} = (\mathsf{ct}^{(1)}_{m_b^*}, \mathsf{ct}^{(2)}_{m_b^*})$ to $\mathcal{A}$.

Guess Phase: Upon receiving $b' \in \{0,1\}$ from $\mathcal{A}$, the distinguisher $\mathcal{D}$ submits 1 to the CSSIDDH challenger $\mathcal{C}$ if $b = b'$, otherwise, submits 0 to $\mathcal{C}$.

We have the following two cases:

- **Case I:** If the input IN to $\mathcal{D}$ is of the form $(E_0, [\boldsymbol{\alpha}]E_0, [\boldsymbol{\beta}]E_0, [\boldsymbol{\beta}][\boldsymbol{\alpha}]E_0)$ then computation proceeds just as in Game_2 and therefore $\Pr\Big[\mathcal{D}(E_0, [\boldsymbol{\alpha}]E_0, [\boldsymbol{\beta}]E_0, [\beta][\boldsymbol{\alpha}]E_0) = 1 \,|\, [\boldsymbol{\alpha}], [\boldsymbol{\beta}], [\boldsymbol{\gamma}] \xleftarrow{\$} \mathsf{CI}(\mathcal{O})\Big] = \Pr[S_2]$
- **Case II:** If the input IN to $\mathcal{D}$ is of the form $(E_0, [\boldsymbol{\alpha}]E_0, [\boldsymbol{\beta}]E_0, [\boldsymbol{\gamma}]E_0)$ then computation proceeds just as in Game_3 and therefore $\Pr\Big[\mathcal{D}(E_0, [\boldsymbol{\alpha}]E_0, [\boldsymbol{\beta}]E_0, [\beta][\boldsymbol{\alpha}]E_0) = 1 \,|\, [\boldsymbol{\alpha}], [\boldsymbol{\beta}], [\boldsymbol{\gamma}] \xleftarrow{\$} \mathsf{CI}(\mathcal{O})\Big] = \Pr[S_3]$

Thus $|\Pr[S_2] - \Pr[S_3]| \leq \epsilon_3$ where ϵ_3 is the advantage $\mathsf{Adv}^{\mathsf{CSSIDDH}}_{\mathcal{D}}(\lambda)$ of $\mathcal{D}$ in solving the CSSIDDH problem which is negligible under the CSSIDDH assumption. □

$\underline{\mathsf{Game}_4}$*:* We now transform Game_3 into Game_4 by computing $H_k(M_C([\tilde{\mathbf{b}}]E_0))$ by simply choosing $h_1 \xleftarrow{\$} \{0,1\}^{\mathsf{mlen}(\lambda)}$, rather than as a hash, i.e, replacing $\mathsf{ct}_{m_b^*}^{\mathsf{G}_3} = (\mathsf{ct}_{m_b^*}^{\mathsf{G}_3,1} = [\mathbf{b}]E_0,\ \mathsf{ct}_{m_b^*}^{\mathsf{G}_3,2} = m_b^* \oplus H_k(M_C([\tilde{\mathbf{b}}]E_0)))$ with

$$\mathsf{ct}_{m_b^*}^{\mathsf{G}_4} = (\mathsf{ct}_{m_b^*}^{\mathsf{G}_4,1} = [\mathbf{b}]E_0,\ \boxed{\mathsf{ct}_{m_b^*}^{\mathsf{G}_4,2} = m_b^* \oplus h_1})$$

where $\mathbf{b} \xleftarrow{\$} [-\mu,\mu]^n$ and $h_1 \xleftarrow{\$} \{0,1\}^{\mathsf{mlen}(\lambda)}$.

Claim 4. $|\Pr[S_3] - \Pr[S_4]| \leq \epsilon_4$ *where* $\epsilon_4(\lambda)$ *negligible function in* λ.

Proof We show below if the adversary $\mathcal{A}$ can distinguish between Game_3 and Game_4 then we can construct a distinguisher $\mathcal{D}$ for the *entropy smoothing* problem of the family of keyed hash functions $\mathcal{H} = \{H_k\}_{k\in\mathcal{K}}$.
Let the entropy smoothing challenger $\mathcal{C}$ provides an instance of entropy smoothing hash function $\mathsf{IN} = (k \in \mathcal{K}, h_1 \in \{0,1\}^{\mathsf{mlen}(\lambda)})$ to $\mathcal{D}$.

Setup: The distinguisher $\mathcal{D}$ generates the public parameter $\mathsf{pp}_{\mathsf{hPKE}} = (p, \ell_1, \ldots, \ell_n, \mu, E_0, \mathcal{H} = \{H_k\}_{k\in\mathcal{K}})$ and computes $\mathsf{pk} = [\mathbf{r}^*]E_0$ where $\mathbf{r}^* \xleftarrow{\$} [-\mu,\mu]^n$ and sends $\mathsf{pp}_{\mathsf{hPKE}}$ along with pk to the adversary $\mathcal{A}$.

Challenge Phase: After receiving the challenge plaintext pair (m_0^*, m_1^*) from $\mathcal{A}$, the distinguisher $\mathcal{D}$ uniformly samples $b \xleftarrow{\$} \{0,1\}$, computes $\mathsf{ct}_{m_b^*}^{(1)} = [\mathbf{b}]E_0,\ \mathsf{ct}_{m_b^*}^{(2)} = m_b^* \oplus h_1$ and sends the ciphertext $\mathsf{ct}_{m_b^*} = (\mathsf{ct}_{m_b^*}^{(1)}, \mathsf{ct}_{m_b^*}^{(2)})$ to $\mathcal{A}$.

Guess Phase: Upon receiving $b' \in \{0,1\}$ from $\mathcal{A}$, the distinguisher $\mathcal{D}$ submits 1 to the entropy smoothing challenger $\mathcal{C}$ if $b = b'$, otherwise, submits 0 to $\mathcal{C}$.

We have the following two cases:

- **Case I:** If the input IN to $\mathcal{D}$ is of the form $(k \in \mathcal{K}, h_1 = H_k(g)) \in \{0,1\}^{\mathsf{mlen}(\lambda)})$ for some $g \in \mathbb{F}_p$ then computation proceeds just as in Game_3 and therefore
$$\Pr\left[\mathcal{D}(k, h_1 = H_k(g)) = 1 \mid k \xleftarrow{\$} \mathcal{K}, g \xleftarrow{\$} \mathbb{F}_p\right] = \Pr[S_3]$$
- **Case II:** If the input IN to $\mathcal{D}$ is of the form $(k \in \mathcal{K}, h_1 \xleftarrow{\$} \{0,1\}^{\mathsf{mlen}(\lambda)})$ then computation proceeds just as in Game_4 and therefore
$$\Pr\left[\mathcal{D}(k, h_1) = 1 \mid k \xleftarrow{\$} \mathcal{K}, h_1 \xleftarrow{\$} \{0,1\}^{\mathsf{mlen}(\lambda)}\right] = \Pr[S_4]$$

Thus

$$|\Pr[S_3] - \Pr[S_4]| \leq \epsilon_4$$

where ϵ_4 is the entropy smoothing advantage $\mathsf{Adv}_{\mathcal{D}}^{\mathsf{ES}}(\lambda)$ of $\mathcal{D}$ which is negligible if the family of keyed hash functions $\mathcal{H} = \{H_k\}_{k\in\mathcal{K}}$ is *entropy smoothing* for any PPT distinguisher $\mathcal{D}$. □

$\underline{\mathsf{Game}_5}$*:* The game Game_5 is identical to Game_4 except that $\mathsf{ct}_{\boldsymbol{\rho}^*}^{\mathsf{G}_4} = (\mathsf{ct}_{\boldsymbol{\rho}^*}^{\mathsf{G}_4,1} = [\mathbf{b}']E_0,\ \mathsf{ct}_{m_b^*}^{\mathsf{G}_4,2} = r \oplus H_k(M_C([\mathbf{b}'][\mathbf{r}^*]E_0)))$ is replaced with

$$\mathsf{ct}_{\boldsymbol{\rho}^*}^{\mathsf{G}_5} = (\mathsf{ct}_{\boldsymbol{\rho}^*}^{\mathsf{G}_5,1} = [\mathbf{b}]E_0,\ \boxed{\mathsf{ct}_{\boldsymbol{\rho}^*}^{\mathsf{G}_5,2} = r \oplus H_k(M_C([\tilde{\mathbf{b}'}]E_0))})$$

where $\mathbf{b}, \tilde{\mathbf{b}'} \xleftarrow{\$} [-\mu,\mu]^n$.
We can show that if the adversary $\mathcal{A}$ can distinguish between Game_4 and Game_5 then we can construct a distinguisher $\mathcal{D}$ for the CSSIDDH problem. The proof is similar to that of **Claim** 3 distinguishing between Game_2 and Game_3. Thus the **Claim** 5 below holds under the CSSIDDH assumption.

Claim 5. $|\Pr[S_4] - \Pr[S_5]| \leq \epsilon_3$ *where* $\epsilon_3(\lambda)$ *negligible function in* λ.

$\underline{\mathsf{Game}_6}$*:* We now transition from Game_5 to Game_6 by modifying the computation of $H_k(M_C([\tilde{\mathbf{b}}']E_0))$. Instead of evaluating it as a hash function, we replace it with a uniformly random value h_2 drawn from $\{0,1\}^{\mathsf{mlen}(\lambda)}$. Thus, the ciphertext in Game_5

$$\mathsf{ct}_{\boldsymbol{\rho}^*}^{\mathsf{G}_5} = \left(\mathsf{ct}_{\boldsymbol{\rho}^*}^{\mathsf{G}_5,1} = [\mathbf{b}']E_0,\ \mathsf{ct}_{m_b^*}^{\mathsf{G}_5,2} = r \oplus H_k(M_C([\tilde{\mathbf{b}}']E_0))\right)$$

in the challenge phase is replaced in Game_6 by

$$\mathsf{ct}_{\boldsymbol{\rho}^*}^{\mathsf{G}_6} = \left(\mathsf{ct}_{\boldsymbol{\rho}^*}^{\mathsf{G}_6,1} = [\mathbf{b}']E_0,\ \boxed{\mathsf{ct}_{\boldsymbol{\rho}^*}^{\mathsf{G}_6,2} = r \oplus h_2}\right)$$

where, $\mathbf{b}', \tilde{\mathbf{b}}'$ are sampled uniformly at random from $[-\mu,\mu]^n$ and h_2 is drawn from $\{0,1\}^{\mathsf{mlen}(\lambda)}$.
Similar to the proof of the **Claim** 3 of distinguishing between Game_3 and Game_4, we can show that if the adversary $\mathcal{A}$ can distinguish between Game_5 and Game_6 then we can construct a distinguisher $\mathcal{D}$ for the *entropy smoothing* problem of the family of keyed hash functions $\mathcal{H} = \{H_k\}_{k\in\mathcal{K}}$. Assuming the family of keyed hash functions $\mathcal{H} = \{H_k\}_{k\in\mathcal{K}}$ is *entropy smoothing* claim follows.

Claim 6. $|\Pr[S_5] - \Pr[S_6]| \leq \epsilon_4$ *where* $\epsilon_4(\lambda)$ *negligible function in* λ.

Finally, in Game_6, the adversary $\mathcal{A}$'s input is

$$\mathsf{pk}^{\mathsf{G}_6} = [\mathbf{r}^*]E_0,\ \mathbf{u},\ \mathsf{ct}_{m_b^*}^{\mathsf{G}_6} = (\mathsf{ct}_{m_b^*}^{\mathsf{G}_6,1} = [\mathbf{b}]E_0,\ \mathsf{ct}_{m_b^*}^{\mathsf{G}_6,2} = m_b^* \oplus h_1),$$
$$\mathsf{ct}_{\boldsymbol{\rho}^*}^{\mathsf{G}_6} = (\mathsf{ct}_{\boldsymbol{\rho}^*}^{\mathsf{G}_6,1} = [\mathbf{b}']E_0,\ \mathsf{ct}_{\boldsymbol{\rho}^*}^{\mathsf{G}_6,2} = \boldsymbol{\rho}^* \oplus h_2)$$

where $\mathbf{u}, \mathbf{b}, \mathbf{b}', r \xleftarrow{\$} [-\mu,\mu]^n, h_1, h_2 \xleftarrow{\$} \{0,1\}^{\mathsf{mlen}(\lambda)}$ and $\mathbf{r}^* = \mathsf{KDF}(\boldsymbol{\rho}^*)$. Here, h_1, h_2 act like one-time pad in Game_6, so probability of correctly guessing $b = b'$ is $\Pr[S_6] = \frac{1}{2}$. Therefore,

$$\begin{aligned}
&|\Pr[S_0] - \Pr[S_6]| \\
&\leq |\Pr[S_0] - \Pr[S_1]| + |\Pr[S_1] - \Pr[S_2]| + |\Pr[S_2] - \Pr[S_3]| \\
&\quad + |\Pr[S_3] - \Pr[S_4]| + |\Pr[S_4] - \Pr[S_5]| + |\Pr[S_5] - \Pr[S_6]| \\
&< \epsilon_1 + \epsilon_2 + 2\epsilon_3 + 2\epsilon_4
\end{aligned}$$

Hence, $|\Pr[S_0] - \frac{1}{2}| < \epsilon$ where $\epsilon = \epsilon_1 + \epsilon_2 + 2\epsilon_3 + 2\epsilon_4$, i.e.

$$\mathsf{Adv}^{f\text{-}\mathsf{CS+LR}}_{\mathsf{Hased\text{-}PKE},\mathcal{A}}(\lambda) = |\Pr[\mathsf{Exp}^{f\text{-}\mathsf{CS+LR}}_{\mathsf{Hased\text{-}PKE},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}| < \epsilon$$

for $f(\mathsf{sk}, \boldsymbol{\rho}^*) = \mathsf{sk} - \mathsf{KDF}(\boldsymbol{\rho}^*)$.

□

4 Security Proof of Theorem 41

In this section, we establish the f-$\mathsf{CS} + \mathsf{LR}$ security of SimS, as stated in Theorem 41 which is essential for proving the security of the updatable public key encryption scheme.

Theorem 30. *The scheme* SimS *is* f-$\mathsf{CS}+\mathsf{LR}$ *secure for* $f : [-\mu,\mu]^n \times \mathbb{Z}_{2^{q-2}} \to [-\mu,\mu]^n$ *as per Definition 20 under the assumption that* $R_{E_{AB}}$ *satisfies the second property of a randomizing function (see Definition 3),* KDF *is a secure key derivation function (see Definition 16) and the* $\mathsf{CSSIDDH}$ *assumption given in Definition 13 holds.*

Proof We prove SimS is f-$\mathsf{CS}+\mathsf{LR}$ secure for $f(\mathsf{sk},\boldsymbol{\rho}^*) = \mathsf{sk} - \mathsf{KDF}(\boldsymbol{\rho}^*)$ using a sequence of games between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ where $\mathsf{KDF} : \mathcal{MS} = \mathbb{Z}_{2^{q-2}} \to \mathcal{KS} = [-\mu,\mu]^n$ is a key derivation function. The adversary's view across the sequence of games is described in Fig. 5 where the changes are highlighted by enclosing them in a box. We define S_i as the event that $b = b'$ in Game_i for $i = 0,\ldots,6$.

$\underline{\mathsf{Game}_0}$*:* This is the original f-$\mathsf{CS}+\mathsf{LR}$ security game $\mathsf{Exp}^{f\text{-}\mathsf{CS}+\mathsf{LR}}_{\mathsf{PKE},\mathcal{A}}(\lambda)$ between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ that works as follows:

Setup: The challenger $\mathcal{C}$ generates the public parameter $\mathsf{pp}_{\mathsf{SimS}} = (p,q,\ell_1,\ldots,\ell_n,\mu,E_0,R_E)$ by running $\mathsf{SimS.Setup}(\lambda)$ and a secret-public key pair $(\mathsf{sk} = \boldsymbol{a} \in [-\mu,\mu]^n,\ \mathsf{pk} = E_A = [\boldsymbol{a}]E_0)$ is generated by running $\mathsf{SimS.KeyGen}(\mathsf{pp}_{\mathsf{SimS}})$. The challenger $\mathcal{C}$ sends $\mathsf{pp}_{\mathsf{SimS}}$ and $\mathsf{pk}^{\mathsf{G}_0} = \mathsf{pk}$ to $\mathcal{A}$ and keeps sk secret to itself.

Challenge Phase: The adversary $\mathcal{A}$ sends $(m_0^*, m_1^*) \in \mathbb{Z}_{2^{q-2}} \times \mathbb{Z}_{2^{q-2}}$ as challenge message pair to $\mathcal{C}$. Then $\mathcal{C}$ samples $\boldsymbol{\rho}^* \xleftarrow{\$} \mathbb{Z}_{2^{q-2}}$ and $b \xleftarrow{\$} \{0,1\}$, computes $\mathbf{r}^* = \mathsf{KDF}(\boldsymbol{\rho}^*) \in [-\mu,\mu]^n$ and sends to $\mathcal{A}$ the tuple $(\mathbf{z}, \mathsf{ct}^{\mathsf{G}_0}_{m_b^*}, \mathsf{ct}^{\mathsf{G}_0}_{\boldsymbol{\rho}^*})$ where $\mathbf{z} = \mathsf{sk} - \mathbf{r}^* = \boldsymbol{a} - \mathbf{r}^*$, $\mathsf{ct}^{\mathsf{G}_0}_{m_b^*} \leftarrow \mathsf{SimS.Enc}(\mathsf{pp}_{\mathsf{SimS}}, \mathsf{pk}, m_b^*)$ with $\mathsf{ct}^{\mathsf{G}_0}_{m_b^*} = (\mathsf{ct}^{\mathsf{G}_0,1}_{m_b^*} = [\mathbf{b}]E_0,\ \mathsf{ct}^{\mathsf{G}_0,2}_{m_b^*} = R_{[\mathbf{b}]E_A}(x([2m_b^*+1]P_{[\mathbf{b}]E_A})))$ and $\mathsf{ct}^{\mathsf{G}_0}_{\boldsymbol{\rho}^*} \leftarrow \mathsf{SimS.Enc}(\mathsf{pp}_{\mathsf{SimS}}, \mathsf{pk}, \boldsymbol{\rho}^*)$ with $\mathsf{ct}^{\mathsf{G}_0}_{\boldsymbol{\rho}^*} = (\mathsf{ct}^{\mathsf{G}_0,1}_{\boldsymbol{\rho}^*} = [\mathbf{b}']E_0,\ \mathsf{ct}^{\mathsf{G}_0,2}_{\boldsymbol{\rho}^*} = R_{[\mathbf{b}']E_A}(x([2\boldsymbol{\rho}^*+1]P_{[\mathbf{b}']E_A})))$ to $\mathcal{A}$ and $\mathbf{b},\mathbf{b}' \xleftarrow{\$} [-\mu,\mu]^n$.

Guess Phase: The adversary $\mathcal{A}$ eventually submits a bit b' to the challenger $\mathcal{C}$ and if $b = b'$ then the experiment $\mathsf{Exp}^{f\text{-}\mathsf{CS}+\mathsf{LR}}_{\mathsf{SimS},\mathcal{A}}(\lambda)$ returns 1, otherwise, it returns 0.

$\underline{\mathsf{Game}_1}$*:* In this game, we modify the generation of the public key and m_b^* and $\boldsymbol{\rho}^*$ are encrypted using the updated public key.

Setup: The challenger $\mathcal{C}$ generates the public parameter $\mathsf{pp}_{\mathsf{SimS}} = (p,q,\ell_1,\ldots,\ell_n,\mu,E_0,R_E) \leftarrow \mathsf{SimS.Setup}(\lambda)$ and samples $\boldsymbol{\rho}^* \xleftarrow{\$} \mathbb{Z}_{2^{q-2}}$, computes $\mathbf{r}^* = \mathsf{KDF}(\boldsymbol{\rho}^*) \in [-\mu,\mu]^n$. The challenger $\mathcal{C}$ sends $\mathsf{pp}_{\mathsf{SimS}}$ and $\boxed{\mathsf{pk}^{\mathsf{G}_1} = [\mathbf{r}^*]E_0}$ to $\mathcal{A}$.

Challenge Phase: The adversary $\mathcal{A}$ sends $(m_0^*, m_1^*) \in \mathbb{Z}_{2^{q-2}} \times \mathbb{Z}_{2^{q-2}}$ as a challenge message pair to $\mathcal{C}$. Then $\mathcal{C}$ uniformly samples $b \xleftarrow{\$} \{0,1\}$ and sends to $\mathcal{A}$ the tuple $(\mathbf{z}, \mathsf{ct}^{\mathsf{G}_1}_{m_b^*}, \mathsf{ct}^{\mathsf{G}_1}_{\boldsymbol{\rho}^*})$ where $\mathbf{z} = f(\mathsf{sk},\boldsymbol{\rho}^*) = \mathsf{sk} - \mathbf{r}^*$, $\mathsf{ct}^{\mathsf{G}_1}_{m_b^*} \leftarrow \mathsf{SimS.Enc}(\mathsf{pp}_{\mathsf{SimS}}, \mathsf{pk}^{\mathsf{G}_1}, m_b^*)$ and $\mathsf{ct}^{\mathsf{G}_1}_{\boldsymbol{\rho}^*} \leftarrow \mathsf{SimS.Enc}(\mathsf{pp}_{\mathsf{SimS}}, \mathsf{pk}^{\mathsf{G}_1}, \boldsymbol{\rho}^*)$ with $\mathsf{ct}^{\mathsf{G}_1}_{m_b^*} = (\mathsf{ct}^{\mathsf{G}_1,1}_{m_b^*} = [\mathbf{b}]E_0,\ \boxed{\mathsf{ct}^{\mathsf{G}_1,2}_{m_b^*} = R_{[\mathbf{b}][\mathbf{r}^*]E_0}(x([2m_b^*+1]P_{[\mathbf{b}][\mathbf{r}^*]E_0}))})$, $\mathsf{ct}^{\mathsf{G}_1}_{\boldsymbol{\rho}^*} = (\mathsf{ct}^{\mathsf{G}_1,1}_{\boldsymbol{\rho}^*} = [\mathbf{b}']E_0,\ \boxed{\mathsf{ct}^{\mathsf{G}_1,2}_{\boldsymbol{\rho}^*} = R_{[\mathbf{b}'][\mathbf{r}^*]E_0}(x([2\boldsymbol{\rho}^*+1]P_{[\mathbf{b}'][\mathbf{r}^*]E_0}))})$ and $\mathbf{b},\mathbf{b}' \xleftarrow{\$} [-\mu,\mu]^n$.

Guess Phase: The adversary $\mathcal{A}$ eventually submits a bit b' to the challenger $\mathcal{C}$ and if $b = b'$ then the experiment $\mathsf{Exp}^{f\text{-}\mathsf{CS}+\mathsf{LR}}_{\mathsf{SimS},\mathcal{A}}(\lambda)$ returns 1, otherwise, it returns 0.

Note that, $\mathsf{pk}^{\mathsf{G}_0} = [\mathsf{sk}]E_0$ in Game_0 is replaced by $\mathsf{pk}^{\mathsf{G}_1} = [\mathbf{r}^*]E_0$ in Game_1 in the setup phase and

$$\mathsf{ct}^{\mathsf{G}_0}_{m_b^*} = (\mathsf{ct}^{\mathsf{G}_0,1}_{m_b^*} = [\mathbf{b}]E_0,\ \mathsf{ct}^{\mathsf{G}_0,2}_{m_b^*} = R_{[\mathbf{b}]E_A}(x([2m_b^*+1]P_{[\mathbf{b}]E_A})))$$

Game	Transition of the game
Game_0	The original f-$\mathsf{CS}+\mathsf{LR}$ security game $\mathsf{pp}_{\mathsf{SimS}} = (p, q, \ell_1, \ldots, \ell_n, \mu, E_0, R_E), \mathsf{pk}^{\mathsf{G}_0} = [\mathsf{sk}]E_0 = E_A,\ \mathbf{z} = \mathsf{sk} - \mathbf{r}^*,$ $\mathsf{ct}^{\mathsf{G}_0}_{m_b^*} = (\mathsf{ct}^{\mathsf{G}_0,1}_{m_b^*} = [\mathbf{b}]E_0,\ \mathsf{ct}^{\mathsf{G}_0,2}_{m_b^*} = R_{[\mathbf{b}]E_A}(x([2m_b^*+1]P_{[\mathbf{b}]E_A}))),$ $\mathsf{ct}^{\mathsf{G}_0}_{\boldsymbol{\rho}^*} = (\mathsf{ct}^{\mathsf{G}_0,1}_{\boldsymbol{\rho}^*} = [\mathbf{b}']E_0,\ \mathsf{ct}^{\mathsf{G}_0,2}_{\boldsymbol{\rho}^*} = R_{[\mathbf{b}']E_A}(x([2\boldsymbol{\rho}^*+1]P_{[\mathbf{b}']E_A})))$ with $\mathbf{r}^* = \mathsf{KDF}(\boldsymbol{\rho}^*), \mathbf{b}, \mathbf{b}' \xleftarrow{\$} [-\mu, \mu]^n$ and $\boldsymbol{\rho}^* \xleftarrow{\$} \mathbb{Z}_{2^{q-2}}$
Game_1	$\mathsf{pp}_{\mathsf{SimS}} = (p, q, \ell_1, \ldots, \ell_n, \mu, E_0, R_E), \boxed{\mathsf{pk}^{\mathsf{G}_1} = [\mathbf{r}^*]E_0},\ \mathbf{z} = \mathsf{sk} - \mathbf{r}^*,$ $\mathsf{ct}^{\mathsf{G}_1}_{m_b^*} = (\mathsf{ct}^{\mathsf{G}_1,1}_{m_b^*} = [\mathbf{b}]E_0,\ \boxed{\mathsf{ct}^{\mathsf{G}_1,2}_{m_b^*} = R_{[\mathbf{b}][\mathbf{r}^*]E_0}(x([2m_b^*+1]P_{[\mathbf{b}][\mathbf{r}^*]E_0}))})$ and $\mathsf{ct}^{\mathsf{G}_1}_{\boldsymbol{\rho}^*} = (\mathsf{ct}^{\mathsf{G}_1,1}_{\boldsymbol{\rho}^*} = [\mathbf{b}']E_0,\ \boxed{\mathsf{ct}^{\mathsf{G}_1,2}_{\boldsymbol{\rho}^*} = R_{[\mathbf{b}'][\mathbf{r}^*]E_0}(x([2\boldsymbol{\rho}^*+1]P_{[\mathbf{b}'][\mathbf{r}^*]E_0}))})$ with $\mathbf{b}, \mathbf{b}' \xleftarrow{\$} [-\mu, \mu]^n$ and $\boldsymbol{\rho}^* \xleftarrow{\$} \mathbb{Z}_{2^{q-2}}$
Game_2	$\mathsf{pp}_{\mathsf{SimS}} = (p, q, \ell_1, \ldots, \ell_n, \mu, E_0, R_E), \mathsf{pk}^{\mathsf{G}_2} = [\mathbf{r}^*]E_0,\ \boxed{\mathbf{u}},$ $\mathsf{ct}^{\mathsf{G}_2}_{m_b^*} = (\mathsf{ct}^{\mathsf{G}_2,1}_{m_b^*} = [\mathbf{b}]E_0,\ \mathsf{ct}^{\mathsf{G}_2,2}_{m_b^*} = R_{[\mathbf{b}][\mathbf{r}^*]E_0}(x([2m_b^*+1]P_{[\mathbf{b}][\mathbf{r}^*]E_0})))$ and $\mathsf{ct}^{\mathsf{G}_2}_{\boldsymbol{\rho}^*} = (\mathsf{ct}^{\mathsf{G}_2,1}_{\boldsymbol{\rho}^*} = [\mathbf{b}']E_0,\ \mathsf{ct}^{\mathsf{G}_2,2}_{\boldsymbol{\rho}^*} = R_{[\mathbf{b}'][\mathbf{r}^*]E_0}(x([2\boldsymbol{\rho}^*+1]P_{[\mathbf{b}'][\mathbf{r}^*]E_0})))$ with $\mathbf{u}, \mathbf{b}, \mathbf{b}' \xleftarrow{\$} [-\mu, \mu]^n$ and $\boldsymbol{\rho}^* \xleftarrow{\$} \mathbb{Z}_{2^{q-2}}$
Game_3	$\mathsf{pp}_{\mathsf{SimS}} = (p, q, \ell_1, \ldots, \ell_n, \mu, E_0, R_E), \mathsf{pk}^{\mathsf{G}_3} = [\mathbf{r}^*]E_0,\ \mathbf{u},$ $\mathsf{ct}^{\mathsf{G}_3}_{m_b^*} = (\mathsf{ct}^{\mathsf{G}_3,1}_{m_b^*} = [\mathbf{b}]E_0,\ \boxed{\mathsf{ct}^{\mathsf{G}_3,2}_{m_b^*} = R_{[\tilde{\mathbf{b}}]E_0}(x([2m_b^*+1]P_{[\tilde{\mathbf{b}}]E_0}))})$ and $\mathsf{ct}^{\mathsf{G}_3}_{\boldsymbol{\rho}^*} = (\mathsf{ct}^{\mathsf{G}_3,1}_{\boldsymbol{\rho}^*} = [\mathbf{b}']E_0,\ \mathsf{ct}^{\mathsf{G}_3,2}_{\boldsymbol{\rho}^*} = R_{[\mathbf{b}'][\mathbf{r}^*]E_0}(x([2\boldsymbol{\rho}^*+1]P_{[\mathbf{b}'][\mathbf{r}^*]E_0})))$ with $\mathbf{u}, \mathbf{b}, \mathbf{b}', \tilde{\mathbf{b}} \xleftarrow{\$} [-\mu, \mu]^n$ and $\boldsymbol{\rho}^* \xleftarrow{\$} \mathbb{Z}_{2^{q-2}}$
Game_4	$\mathsf{pp}_{\mathsf{SimS}} = (p, q, \ell_1, \ldots, \ell_n, \mu, E_0, R_E), \mathsf{pk}^{\mathsf{G}_4} = [\mathbf{r}^*]E_0,\ \mathbf{u},$ $\mathsf{ct}^{\mathsf{G}_4}_{m_b^*} = (\mathsf{ct}^{\mathsf{G}_4,1}_{m_b^*} = [\mathbf{b}]E_0,\ \boxed{\mathsf{ct}^{\mathsf{G}_4,2}_{m_b^*} = h_1})$ and $\mathsf{ct}^{\mathsf{G}_4}_{\boldsymbol{\rho}^*} = (\mathsf{ct}^{\mathsf{G}_4,1}_{\boldsymbol{\rho}^*} = [\mathbf{b}']E_0,\ \mathsf{ct}^{\mathsf{G}_4,2}_{\boldsymbol{\rho}^*} = R_{[\mathbf{b}'][\mathbf{r}^*]E_0}(x([2\boldsymbol{\rho}^*+1]P_{[\mathbf{b}'][\mathbf{r}^*]E_0})))$ with $\mathbf{u}, \mathbf{b}, \mathbf{b}' \xleftarrow{\$} [-\mu, \mu]^n$ and $h_1 \xleftarrow{\$} \mathbb{F}_p$
Game_5	$\mathsf{pp}_{\mathsf{SimS}} = (p, q, \ell_1, \ldots, \ell_n, \mu, E_0, R_E), \mathsf{pk}^{\mathsf{G}_5} = [\mathbf{r}^*]E_0,\ \mathbf{u},$ $\mathsf{ct}^{\mathsf{G}_5}_{m_b^*} = (\mathsf{ct}^{\mathsf{G}_5,1}_{m_b^*} = [\mathbf{b}]E_0,\ \mathsf{ct}^{\mathsf{G}_5,2}_{m_b^*} = h_1)$ and $\mathsf{ct}^{\mathsf{G}_5}_{\boldsymbol{\rho}^*} = (\mathsf{ct}^{\mathsf{G}_5,1}_{\boldsymbol{\rho}^*} = [\mathbf{b}']E_0,\ \boxed{\mathsf{ct}^{\mathsf{G}_5,2}_{\boldsymbol{\rho}^*} = R_{[\tilde{\mathbf{b}}']E_0}(x([2\boldsymbol{\rho}^*+1]P_{[\tilde{\mathbf{b}}']E_0}))})$ with $\mathbf{u}, \mathbf{b}, \mathbf{b}', \tilde{\mathbf{b}}' \xleftarrow{\$} [-\mu, \mu]^n$ and $h_1 \xleftarrow{\$} \mathbb{F}_p$
Game_6	$\mathsf{pp}_{\mathsf{SimS}} = (p, q, \ell_1, \ldots, \ell_n, \mu, E_0, R_E), \mathsf{pk}^{\mathsf{G}_6} = [\mathbf{r}^*]E_0,\ \mathbf{u},$ $\mathsf{ct}^{\mathsf{G}_6}_{m_b^*} = (\mathsf{ct}^{\mathsf{G}_6,1}_{m_b^*} = [\mathbf{b}]E_0,\ \mathsf{ct}^{\mathsf{G}_6,2}_{m_b^*} = h_1)$ and $\mathsf{ct}^{\mathsf{G}_6}_{\boldsymbol{\rho}^*} = (\mathsf{ct}^{\mathsf{G}_6,1}_{\boldsymbol{\rho}^*} = [\mathbf{b}']E_0,\ \boxed{\mathsf{ct}^{\mathsf{G}_6,2}_{\boldsymbol{\rho}^*} = h_2})$ with $\mathbf{u}, \mathbf{b}, \mathbf{b}' \xleftarrow{\$} [-\mu, \mu]^n$ and $h_1, h_2 \xleftarrow{\$} \mathbb{F}_p$

Table 5 Changes in the adversary's view across the sequence of games.

$$\mathsf{ct}^{\mathsf{G}_0}_{\boldsymbol{\rho}^*} = (\mathsf{ct}^{\mathsf{G}_0,1}_{\boldsymbol{\rho}^*} = [\mathbf{b}']E_0,\ \mathsf{ct}^{\mathsf{G}_0,2}_{\boldsymbol{\rho}^*} = R_{[\mathbf{b}']E_A}(x([2\boldsymbol{\rho}^* + 1]P_{[\mathbf{b}']E_A})))$$

in Game_0 are replaced by

$$\mathsf{ct}^{\mathsf{G}_1}_{m^*_b} = (\mathsf{ct}^{\mathsf{G}_1,1}_{m^*_b} = [\mathbf{b}]E_0,\ \mathsf{ct}^{\mathsf{G}_1,2}_{m^*_b} = R_{[\mathbf{b}][\mathbf{r}^*]E_0}(x([2m^*_b + 1]P_{[\mathbf{b}][\mathbf{r}^*]E_0})))$$

$$\mathsf{ct}^{\mathsf{G}_1}_{\boldsymbol{\rho}^*} = (\mathsf{ct}^{\mathsf{G}_1,1}_{\boldsymbol{\rho}^*} = [\mathbf{b}']E_0,\ \mathsf{ct}^{\mathsf{G}_1,2}_{\boldsymbol{\rho}^*} = R_{[\mathbf{b}'][\mathbf{r}^*]E_0}(x([2\boldsymbol{\rho}^* + 1]P_{[\mathbf{b}'][\mathbf{r}^*]E_0})))$$

in Game_1 in challenge phase. As $\mathsf{pk}^{\mathsf{G}_0}$ and $\mathsf{pk}^{\mathsf{G}_1}$ are identically distributed, the adversary's view in Game_0 and Game_1 is indistinguishable. Consequently, we have **Claim** 1 below.

Claim 1. $|\Pr[S_0] - \Pr[S_1]| \leq \epsilon_1(\lambda)$ *where* $\epsilon_1(\lambda)$ *is a negligible function in* λ.

$\underline{\mathsf{Game}_2}$: This game identical to Game_1 except the leakage $\mathbf{z} = f(\mathsf{sk}, \boldsymbol{\rho}) = \mathsf{sk} - \mathbf{r}^*$ where $\mathbf{r}^* = \mathsf{KDF}(\boldsymbol{\rho}^*)$ in Game_1 is replaced by a random element $\mathbf{u} \xleftarrow{\$} [-\mu, \mu]^n$ in Game_2.
Since the public key remains unchanged, i.e., $\mathsf{pk}^{\mathsf{G}_2} = \mathsf{pk}^{\mathsf{G}_1} = [\mathbf{r}^*]E_0$, the ciphertexts are also identical with $\mathsf{ct}^{\mathsf{G}_2}_{m^*_b} = \mathsf{ct}^{\mathsf{G}_1}_{m^*_b} = (\mathsf{ct}^{\mathsf{G}_1,1}_{m^*_b} = [\mathbf{b}]E_0, \mathsf{ct}^{\mathsf{G}_1,2}_{m^*_b} = R_{[\mathbf{b}][\mathbf{r}^*]E_0}(x([2m^*_b + 1]P_{[\mathbf{b}][\mathbf{r}^*]E_0})))$ and $\mathsf{ct}^{\mathsf{G}_2}_{\boldsymbol{\rho}^*} = \mathsf{ct}^{\mathsf{G}_1}_{\boldsymbol{\rho}^*} = (\mathsf{ct}^{\mathsf{G}_1,1}_{\boldsymbol{\rho}^*} = [\mathbf{b}']E_0, \mathsf{ct}^{\mathsf{G}_1,2}_{\boldsymbol{\rho}^*} = R_{[\mathbf{b}'][\mathbf{r}^*]E_0}(x([2\boldsymbol{\rho}^* + 1]P_{[\mathbf{b}'][\mathbf{r}^*]E_0})))$.
Given that $\mathbf{z} = f(\mathsf{sk}, \boldsymbol{\rho}^*) = \mathsf{sk} - \mathbf{r}^*$ and $\mathbf{r}^* = \mathsf{KDF}(\boldsymbol{\rho}^*)$, the security of the key derivation function KDF guarantees that $\mathsf{KDF}(\boldsymbol{\rho}^*)$ is randomly distributed over $[-\mu, \mu]^n$. Consequently, the term $\mathsf{sk} - \mathbf{r}^* = \mathsf{sk} - \mathsf{KDF}(\boldsymbol{\rho}^*)$ is uniformly random within $[-\mu, \mu]^n$. Both $\mathsf{sk} - \mathbf{r}^*$ and $\mathbf{u}$ are uniformly distributed over the same domain, making their distributions indistinguishable. Consequently, the adversary's advantage in distinguishing Game_1 from Game_2 is negligible, implying that Game_1 and Game_2 are statistically close and we have the following claim.

Claim 2. $|\Pr[S_1] - \Pr[S_2]| \leq \epsilon_2(\lambda)$ *where* $\epsilon_2(\lambda)$ *is a negligible function in* λ.

$\underline{\mathsf{Game}_3}$: In Game_2, the adversary $\mathcal{A}$'s view is $\mathsf{pp}_{\mathsf{SimS}}, \mathsf{pk}^{\mathsf{G}_2} = [\mathbf{r}^*]E_0$, $\mathbf{u}$, $\mathsf{ct}^{\mathsf{G}_2}_{m^*_b} = (\mathsf{ct}^{\mathsf{G}_2,1}_{m^*_b} = [\mathbf{b}]E_0,\ \mathsf{ct}^{\mathsf{G}_2,2}_{m^*_b} = R_{[\mathbf{b}][\mathbf{r}^*]E_0}(x([2m^*_b + 1]P_{[\mathbf{b}][\mathbf{r}^*]E_0})))$ and $\mathsf{ct}^{\mathsf{G}_2}_{\boldsymbol{\rho}^*} = (\mathsf{ct}^{\mathsf{G}_2,1}_{\boldsymbol{\rho}^*} = [\mathbf{b}']E_0,\ \mathsf{ct}^{\mathsf{G}_2,2}_{\boldsymbol{\rho}^*} = R_{[\mathbf{b}'][\mathbf{r}^*]E_0}(x([2\boldsymbol{\rho}^* + 1]P_{[\mathbf{b}'][\mathbf{r}^*]E_0})))$ where $\mathbf{u}, \mathbf{b}, \mathbf{b}' \xleftarrow{\$} [-\mu, \mu]^n, \boldsymbol{\rho}^* \xleftarrow{\$} \mathbb{Z}_{2^{q-2}}$.
Now we transform Game_2 into Game_3 by changing the ciphertext

$$\mathsf{ct}^{\mathsf{G}_2}_{m^*_b} = (\mathsf{ct}^{\mathsf{G}_2,1}_{m^*_b} = [\mathbf{b}]E_0,\ \mathsf{ct}^{\mathsf{G}_2,2}_{m^*_b} = R_{[\mathbf{b}][\mathbf{r}^*]E_0}(x([2m^*_b + 1]P_{[\mathbf{b}][\mathbf{r}^*]E_0})))$$

in Game_2 to

$$\mathsf{ct}^{\mathsf{G}_3}_{m^*_b} = (\mathsf{ct}^{\mathsf{G}_3,1}_{m^*_b} = [\mathbf{b}]E_0,\ \boxed{\mathsf{ct}^{\mathsf{G}_3,2}_{m^*_b} = R_{[\tilde{\mathbf{b}}]E_0}(x([2m^*_b + 1]P_{[\tilde{\mathbf{b}}]E_0}))})$$

in Game_3 where $\tilde{\mathbf{b}} \xleftarrow{\$} [-\mu, \mu]^n$.

Claim 3. $|\Pr[S_2] - \Pr[S_3]| \leq \epsilon_3(\lambda)$ *where* $\epsilon_3(\lambda) = \mathsf{Adv}^{\mathsf{CSSIDDH}}_{\mathcal{D}}(\lambda)$ *is a negligible function in* λ.

Proof We will show that if the adversary $\mathcal{A}$ can distinguish between Game_3 and Game_2 then we can construct a distinguisher $\mathcal{D}$ for the $\mathsf{CSSIDDH}$ problem. Let the challenger $\mathcal{C}$ provides an instance $IN = (E_0, [\boldsymbol{\alpha}]E_0, [\boldsymbol{\beta}]E_0, [\boldsymbol{\gamma}]E_0)$ of $\mathsf{CSSIDDH}$ problem to the distinguisher $\mathcal{D}$.

Setup: The distinguisher $\mathcal{D}$ generates the public parameter $\mathsf{pp}_{\mathsf{SimS}} = (p, q, \ell_1, \ldots, \ell_n, \mu, E_0, R_E) \leftarrow \mathsf{SimS.Setup}(\lambda)$ and sends $\mathsf{pp}_{\mathsf{SimS}}$ along with $\mathsf{pk} = [\boldsymbol{\alpha}]E_0$ to the adversary $\mathcal{A}$.
Simulation of the Challenge Phase: After receiving the challenge plaintext pair (m_0^*, m_1^*) from $\mathcal{A}$, the distinguisher $\mathcal{D}$ uniformly samples $b \xleftarrow{\$} \{0,1\}$, computes $\mathsf{ct}^{(1)} = [\boldsymbol{\beta}]E_0$, $\mathsf{ct}^{(2)} = R_{[\boldsymbol{\gamma}]E_0}(x([2m_b^* + 1]P_{[\boldsymbol{\gamma}]E_0}))$ and sends the ciphertext $\mathsf{ct} = (\mathsf{ct}^{(1)}, \mathsf{ct}^{(2)})$ to $\mathcal{A}$.
Guess Phase: Upon receiving b' from $\mathcal{A}$, the distinguisher $\mathcal{D}$ submits 1 to $\mathcal{C}$ if $b = b'$, otherwise, submits 0 to $\mathcal{C}$.
We have the following two cases:

- **Case I:** If the input IN to $\mathcal{D}$ is of the form $(E_0, [\boldsymbol{\alpha}]E_0, [\boldsymbol{\beta}]E_0, [\boldsymbol{\beta}][\boldsymbol{\alpha}]E_0)$ then computation proceeds just as in Game_2 and therefore $\Pr\left[\mathcal{D}(E_0, [\boldsymbol{\alpha}]E_0, [\boldsymbol{\beta}]E_0, [\beta][\boldsymbol{\alpha}]E_0) = 1 \mid [\boldsymbol{\alpha}], [\boldsymbol{\beta}], [\boldsymbol{\gamma}] \xleftarrow{\$} \mathsf{CI}(\mathcal{O})\right] = \Pr[S_2]$.
- **Case II:** If the input IN to $\mathcal{D}$ is of the form $([\boldsymbol{\alpha}]E_0, [\boldsymbol{\beta}]E_0, [\boldsymbol{\gamma}]E_0)$ then computation proceeds just as in Game_3 and therefore $\Pr[S_3] = \Pr\left[\mathcal{D}(E_0, [\boldsymbol{\alpha}]E_0, [\boldsymbol{\beta}]E_0, [\beta][\boldsymbol{\alpha}]E_0) = 1 \mid [\boldsymbol{\alpha}], [\boldsymbol{\beta}], [\boldsymbol{\gamma}] \xleftarrow{\$} \mathsf{CI}(\mathcal{O})\right]$.

Thus

$$|\Pr[S_2] - \Pr[S_3]| \leq \epsilon_3$$

where ϵ_3 is the advantage $\mathsf{Adv}_{\mathcal{D}}^{\mathsf{CSSIDDH}}(\lambda)$ of $\mathcal{D}$ in solving the $\mathsf{CSSIDDH}$ problem (see Definition 13) which is negligible under the $\mathsf{CSSIDDH}$ assumption.

□

$\underline{\mathsf{Game}_4}$*:* The game Game_3 and Game_4 are exactly same except that the term $R_{[\tilde{\mathbf{b}}]}([\tilde{\mathbf{b}}]E_0)$ is replaced by $h_1 \xleftarrow{\$} \mathbb{F}_p$ i.e, the ciphertext

$$\mathsf{ct}_{m_b^*}^{\mathsf{G}_3} = (\mathsf{ct}_{m_b^*}^{\mathsf{G}_3,1} = [\mathbf{b}]E_0,\ \mathsf{ct}_{m_b^*}^{\mathsf{G}_3,2} = R_{[\tilde{\mathbf{b}}]E_0}(x([2m_b^* + 1]P_{[\tilde{\mathbf{b}}]E_0})))$$

in Game_3 is replaced by

$$\mathsf{ct}_{m_b^*}^{\mathsf{G}_4} = (\mathsf{ct}_{m_b^*}^{\mathsf{G}_4,1} = [\mathbf{b}]E_0,\ \boxed{\mathsf{ct}_{m_b^*}^{\mathsf{G}_4,2} = h_1})$$

in Game_4 in the challenge phase.
The second property of the randomizing function (see Definition 3) states that for any $x \in \mathbb{F}_p$, an adversary without access to x and E, cannot distinguish $R_E(x)$ from a random element in $\mathbb{F}_p$. Given that $[\tilde{\mathbf{b}}]E_0$ is unknown to the adversary, the value $R_{[\tilde{\mathbf{b}}]E_0}(x([2m_b^* + 1]P_{[\tilde{\mathbf{b}}]E_0}))$ is uniformly distributed over $\mathbb{F}_p$. Since both $R_{[\tilde{\mathbf{b}}]E_0}(x([2m_b^* + 1]P_{[\tilde{\mathbf{b}}]E_0}))$ and h_1 are uniform in $\mathbb{F}_p$, the adversary cannot distinguish Game_4 from Game_3. Consequently, the statistical distance between these games is bounded and **Claim** 4 follows.

Claim 4. $|\Pr[S_3] - \Pr[S_4]| \leq \epsilon_4(\lambda)$ *where* $\epsilon_4(\lambda)$ *is a negligible function in* λ.

$\underline{\mathsf{Game}_5}$*:* Next we transform Game_4 into Game_5 by changing the challenge ciphertext

$$\mathsf{ct}_{\boldsymbol{\rho}^*}^{\mathsf{G}_4} = (\mathsf{ct}_{\boldsymbol{\rho}^*}^{\mathsf{G}_4,1} = [\mathbf{b}']E_0,\ \mathsf{ct}_{\boldsymbol{\rho}^*}^{\mathsf{G}_4,2} = R_{[\mathbf{b}'][\mathbf{r}^*]E_0}(x([2\boldsymbol{\rho}^* + 1]P_{[\mathbf{b}'][\mathbf{r}^*]E_0})))$$

in Game_4 to

$$\mathsf{ct}_{\boldsymbol{\rho}^*}^{\mathsf{G}_5} = (\mathsf{ct}_{\boldsymbol{\rho}^*}^{\mathsf{G}_5,1} = [\mathbf{b}']E_0,\ \boxed{\mathsf{ct}_{m_b^*}^{\mathsf{G}_3,2} = R_{[\tilde{\mathbf{b}'}]E_0}(x([2\boldsymbol{\rho}^* + 1]P_{[\tilde{\mathbf{b}'}]E_0}))})$$

in Game_5 where $\tilde{\mathbf{b}'} \xleftarrow{\$} [-\mu, \mu]^n$.
We can show that if the adversary $\mathcal{A}$ can distinguish between Game_4 and Game_5 then we can construct a distinguisher $\mathcal{D}$ for the CSSIDDH problem. Hence, the claim follows. The proof is similar to distinguishing between Game_3 and Game_4.

Claim 5. $|\Pr[S_4] - \Pr[S_5]| \leq \epsilon_3(\lambda)$ *where* $\epsilon_3(\lambda) = \mathsf{Adv}_{\mathcal{D}}^{\mathsf{CSSIDDH}}(\lambda)$ *is a negligible function in* λ.

$\underline{\mathsf{Game}_6}$*:* The game Game_5 and Game_6 exactly same except that the term $R_{[\tilde{\mathbf{b}'}]}([\tilde{\mathbf{b}'}]E_0)$ s replaced by $h_2 \xleftarrow{\$} \mathbb{F}_p$ i.e, replacing the ciphertext

$$\mathsf{ct}_{\boldsymbol{\rho}^*}^{\mathsf{G}_5} = (\mathsf{ct}_{\boldsymbol{\rho}^*}^{\mathsf{G}_5,1} = [\mathbf{b}']E_0,\ \mathsf{ct}_{\boldsymbol{\rho}^*}^{\mathsf{G}_5,2} = R_{[\tilde{\mathbf{b}'}]E_0}(x([2\boldsymbol{\rho}^* + 1]P_{[\tilde{\mathbf{b}'}]E_0})))$$

in Game_3 s replaced by

$$\mathsf{ct}_{\boldsymbol{\rho}^*}^{\mathsf{G}_6} = (\mathsf{ct}_{\boldsymbol{\rho}^*}^{\mathsf{G}_6,1} = [\mathbf{b}]E_0,\ \boxed{\mathsf{ct}_{\boldsymbol{\rho}^*}^{\mathsf{G}_6,2} = h_2})$$

in Game_4 in the challenge phase.
Since R_E is a randomizing function, Game_5 is indistinguishable from Game_6. Hence, **Claim** 6 follows.

Claim 6. $|\Pr[S_5] - \Pr[S_6]| \leq \epsilon_4(\lambda)$ *where* $\epsilon_4(\lambda)$ *is a negligible function in* λ.

Finally, in Game_6, the adversary $\mathcal{A}$'s view is

$$\mathsf{pp}_{\mathsf{SimS}} = (p, q, \ell_1, \ldots, \ell_n, \mu, E_0, R_E), \mathsf{pk}^{\mathsf{G}_6} = [\mathbf{r}^*]E_0,\ \mathbf{u},\ \mathsf{ct}_{m_b^*}^{\mathsf{G}_6} = (\mathsf{ct}_{m_b^*}^{\mathsf{G}_6,1}$$
$$= [\mathbf{b}]E_0,\ \mathsf{ct}_{m_b^*}^{\mathsf{G}_6,2} = h_1), \mathsf{ct}_{\boldsymbol{\rho}^*}^{\mathsf{G}_6} = (\mathsf{ct}_{\boldsymbol{\rho}^*}^{\mathsf{G}_6,1} = [\mathbf{b}']E_0,\ \mathsf{ct}_{\boldsymbol{\rho}^*}^{\mathsf{G}_6,2} = h_2)$$

where $\mathbf{u}, \mathbf{b}, \mathbf{b}' \xleftarrow{\$} [-\mu, \mu]^n$ and $h_1, h_2 \xleftarrow{\$} \mathbb{F}_p$. Consequently, the probability of correctly guessing $b = b'$ is $\Pr[S_6] = \frac{1}{2}$.
Therefore,

$$\begin{aligned}
&|\Pr[S_0] - \Pr[S_6]| \\
&\leq |\Pr[S_0] - \Pr[S_1]| + |\Pr[S_1] - \Pr[S_2]| + |\Pr[S_2] - \Pr[S_3]| \\
&\quad + |\Pr[S_3] - \Pr[S_4]| + |\Pr[S_4] - \Pr[S_5]| + |\Pr[S_5] - \Pr[S_6]| \\
&< \epsilon_1 + \epsilon_2 + 2\epsilon_3 + 2\epsilon_4
\end{aligned}$$

Hence, $\mathsf{Adv}_{\mathsf{SimS},\mathcal{A}}^{f\text{-}\mathsf{CS+LR}}(\lambda) = |\Pr[S_0] - \frac{1}{2}| < \epsilon$ where $\epsilon = \epsilon_1 + \epsilon_2 + 2\epsilon_3 + 2\epsilon_4$.

□

5 Our Protocols for Updatable PKE

5.1 Construction 1 : UhPKE

Our first construction of updatable public key encryption $\mathsf{UhPKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{UpdatePk}, \mathsf{UpdateSk})$ is associated with a message space $\mathcal{MS} = \{0,1\}^{\mathsf{mlen}(\lambda)}$

and a randomness space $\mathcal{RS} = \{0,1\}^{\mathsf{mlen}(\lambda)}$. It is based on $\mathsf{hPKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ recalled in Appendix A.1 with two additional PPT algorithms $\mathsf{UpdatePk}$ and $\mathsf{UpdateSk}$ and works as follows.

$\mathsf{UhPKE.Setup}(\lambda) \rightarrow \mathsf{pp}_{\mathsf{UhPKE}} = (\mathsf{pp}_{\mathsf{hPKE}} = (p, \ell_1, \ldots, \ell_n, \mu, E_0, \mathcal{H} = \{H_k\}_{k\in\mathcal{K}}), \mathsf{KDF})$: A trusted party runs this algorithm on the input a security parameter λ and generates the public parameter $\mathsf{pp}_{\mathsf{UhPKE}} = (\mathsf{pp}_{\mathsf{hPKE}} = (p, \ell_1, \ldots, \ell_n, \mu, E_0, \mathcal{H} = \{H_k\}_{k\in\mathcal{K}}), \mathsf{KDF})$ where $\mathsf{pp}_{\mathsf{hPKE}} \leftarrow \mathsf{UhPKE.Setup}(\lambda)$ and $\mathsf{KDF} : \{0,1\}^{\mathsf{mlen}(\lambda)} \rightarrow [-\mu, \mu]^n$ is a key derivation function. More specifically, the trusted parties execute the following steps:

i. Chooses a prime $p = 4\Pi_{i=1}^{n}\ell_i - 1$ where ℓ_i's are distinct odd primes. Sets a base curve $E_0 : y^2 = x^3 + x \in \mathsf{Ell}_p(\mathcal{O})$ over $\mathbb{F}_p$ with $\mathcal{O} = \mathbb{Z}[\sqrt{-p}]$ and picks an integer μ such that $(2\mu+1)^n \geq \#\mathsf{Cl}(\mathcal{O})$.
ii. Selects a key derivation function $\mathsf{KDF} : \{0,1\}^{\mathsf{mlen}(\lambda)} \rightarrow [-\mu, \mu]^n$.
iii. Samples a family of keyed hash function $\mathcal{H} = \{H_k\}_{k\in\mathcal{K}}$ where $H_k : \mathbb{F}_p \rightarrow \{0,1\}^{\mathsf{mlen}(\lambda)}$ for each $k \in \mathcal{K}$, $\mathcal{K}$ being key space.
iv. Outputs the public parameter $\mathsf{pp}_{\mathsf{UhPKE}} = (\mathsf{pp}_{\mathsf{hPKE}} = (p, \ell_1, \ldots, \ell_n,\ \mu, E_0, \mathcal{H} = \{H_k\}_{k\in\mathcal{K}}), \mathsf{KDF})$.

$\mathsf{UhPKE.KeyGen}(\mathsf{pp}_{\mathsf{UhPKE}}) \rightarrow (\mathsf{sk}_0^{(\mathsf{u})} = \boldsymbol{a}_0, \mathsf{pk}_0^{(\mathsf{u})} = E_{A_0})$: On input the public parameter $\mathsf{pp}_{\mathsf{UhPKE}} = (\mathsf{pp}_{\mathsf{hPKE}} = (p, \ell_1, \ldots, \ell_n, \mu, E_0,\ \mathcal{H} = \{H_k\}_{k\in\mathcal{K}}), \mathsf{KDF})$, a user generates its secret-public key pair $(\mathsf{sk}_0^{(\mathsf{u})}, \mathsf{pk}_0^{(\mathsf{u})})$ exactly the same way as in $\mathsf{hPKE.KeyGen}(\mathsf{pp}_{\mathsf{hPKE}})$ in Appendix A.1.

i. Samples $\boldsymbol{a}_0 = (a_1^0, \ldots, a_n^0) \xleftarrow{\$} [-\mu, \mu]^n$ and defines $[\boldsymbol{a}_0] = [\mathfrak{l}_1^{a_1^0} \cdots \mathfrak{l}_n^{a_n^0}] \in \mathsf{Cl}(\mathcal{O})$ where $\mathfrak{l}_i = \langle \ell_i, \pi - 1\rangle$ and π is the Frobenius endomorphism defined as $\pi(x, y) = (x^p, y^p)$ with $\pi(O) = O$.
ii. Computes $E_{A_0} = [\boldsymbol{a}_0]E_0$.
iii. Sets the public key $\mathsf{pk}_0^{(\mathsf{u})} = E_{A_0}$ and the secret key $\mathsf{sk}_0^{(\mathsf{u})} = \boldsymbol{a}_0$. Publish $\mathsf{pk}_0^{(\mathsf{u})}$ and keeps $\mathsf{sk}_0^{(\mathsf{u})}$ secret to itself.

$\mathsf{UhPKE.Enc}(\mathsf{pp}_{\mathsf{UhPKE}}, \mathsf{pk}_i^{(\mathsf{u})} = E_{A_i},\ m) \rightarrow \mathsf{ct}_m$: This algorithm is exactly same as $\mathsf{hPKE.Enc}(\mathsf{pp}_{\mathsf{hPKE}}, \mathsf{pk}_i^{(\mathsf{u})}, m)$ described in Appendix A.1. Taking input the public parameter $\mathsf{pp}_{\mathsf{UhPKE}} = (\mathsf{pp}_{\mathsf{hPKE}} = (p, \ell_1, \ldots, \ell_n, \mu, E_0, \mathcal{H} = \{H_k\}_{k\in\mathcal{K}}), \mathsf{KDF})$, public key $\mathsf{pk}_i^{(\mathsf{u})} = E_{A_i}$ and a message $m \in \mathcal{MS}$, an encrypter proceeds to compute ciphertext as follows:

i. Randomly samples $\mathbf{b} = (b_1, \ldots, b_n) \xleftarrow{\$} [-\mu, \mu]^n$ and defines $[\mathbf{b}] = [\mathfrak{l}_1^{b_1} \cdots \mathfrak{l}_n^{b_n}] \in \mathsf{Cl}(\mathcal{O})$.
ii. Computes $\mathsf{ct}_m^{(1)} = [\mathbf{b}]E_0$ and $\mathsf{ct}_m^{(2)} = m \oplus H_k(M_C([\mathbf{b}]E_{A_i}))$ where $M_C([\mathbf{b}]E_{A_i})$ is the Montgomery coefficient of $[\mathbf{b}]E_{A_i}$.
iii. Sets the ciphertext $\mathsf{ct}_m = (\mathsf{ct}_m^{(1)}, \mathsf{ct}_m^{(2)})$.

$\mathsf{UhPKE.Dec}(\mathsf{pp}_{\mathsf{UhPKE}}, \mathsf{sk}_i^{(\mathsf{u})} = \boldsymbol{a}_i, \mathsf{ct}_m = (\mathsf{ct}_m^{(1)}, \mathsf{ct}_m^{(2)})) \rightarrow m/\ \perp$: Similar to $\mathsf{hPKE.Dec}(\mathsf{pp}_{\mathsf{hPKE}}, \mathsf{sk}_i^{(\mathsf{u})}, \mathsf{ct}_m)$ decrypter with its secret key $\mathsf{sk}_i^{(\mathsf{u})}$ runs this deterministic algorithm on input the public parameter $\mathsf{pp}_{\mathsf{UhPKE}} = (\mathsf{pp}_{\mathsf{hPKE}} = (p, \ell_1, \ldots, \ell_n, \mu, E_0, M_C, \mathcal{H} = \{H_k\}_{k\in\mathcal{K}}), \mathsf{KDF})$, ciphertext ct_m and its secret key $\mathsf{sk}_i^{(\mathsf{u})} = \boldsymbol{a}_i$ and recovers the plaintext m by performing the following steps:

i. Computes $h = H_k(M_C([\boldsymbol{a}_i]\mathsf{ct}_m^{(1)}))$ where $M_C([\boldsymbol{a}_i]\mathsf{ct}_m^{(1)})$ is the Montgomery coefficient of the elliptic curve $[\boldsymbol{a}_i]\mathsf{ct}_m^{(1)}$.

ii. Decrypts the message $\mathsf{ct}_m^{(2)} \oplus h$.

$\mathsf{UhPKE.UpdatePk}(\mathsf{pp}_{\mathsf{UhPKE}}, \mathsf{pk}_i^{(\mathsf{u})}, \boldsymbol{\rho}_i) \to (\mathsf{pk}_{i+1}^{(\mathsf{u})}, \mathsf{up}_{i+1})$: Given the public parameter $\mathsf{pp}_{\mathsf{UhPKE}} = (\mathsf{pp}_{\mathsf{hPKE}} = (p, \ell_1, \ldots, \ell_n, \mu, E_0, \mathcal{H} = \{H_k\}_{k\in\mathcal{K}}), \mathsf{KDF})$, a public key $\mathsf{pk}_i^{(\mathsf{u})} = E_{A_i}$ and a random $\boldsymbol{\rho}_i \in \{0,1\}^{\mathsf{mlen}(\lambda)}$, a user runs this algorithm and proceeds to compute an updated ciphertext up_{i+1} and a updated public key $\mathsf{pk}_{i+1}^{(\mathsf{u})}$ in the following way:

i. Define $\mathbf{r}_i = (r_1, \ldots, r_n) = \mathsf{KDF}(\boldsymbol{\rho}_i)$ and $[\mathbf{r}_i] = [\mathfrak{l}_1^{r_1} \cdots \mathfrak{l}_n^{r_n}] \in \mathsf{Cl}(\mathcal{O})$ where $\mathfrak{l}_i = \langle \ell_i, \pi - 1\rangle$.

ii. Computes $E_{A_{i+1}} = [-\mathbf{r}_i]\mathsf{pk}_i^{(\mathsf{u})}$ and updated ciphertext $\mathsf{up}_{i+1} \leftarrow \mathsf{UhPKE.Enc}(\mathsf{pp}_{\mathsf{UhPKE}}, \mathsf{pk}_i^{(\mathsf{u})}, \boldsymbol{\rho}_i)$ with $\mathsf{up}_{i+1} = (\mathsf{ct}_{\boldsymbol{\rho}_i}^{(1)} = [\mathbf{b}']E_0,\ \mathsf{ct}_{\boldsymbol{\rho}_i}^{(2)} = \boldsymbol{\rho}_i \oplus H_k(M_C([\mathbf{b}']\mathsf{pk}_i^{(\mathsf{u})})))$ for some $\mathbf{b}' \xleftarrow{\$} [-\mu, \mu]^n$.

iii. Returns updated public key $\mathsf{pk}_{i+1}^{(\mathsf{u})} = E_{A_{i+1}}$ and updated ciphertext $\mathsf{up}_{i+1} = (\mathsf{ct}_{\boldsymbol{\rho}_i}^{(1)}, \mathsf{ct}_{\boldsymbol{\rho}_i}^{(2)})$.

$\mathsf{UhPKE.UpdateSk}(\mathsf{pp}_{\mathsf{UhPKE}}, \mathsf{sk}_i^{(\mathsf{u})}, \mathsf{up}_{i+1}) \to \mathsf{sk}_{i+1}^{(\mathsf{u})}$: On the input of the public parameter $\mathsf{pp}_{\mathsf{UhPKE}} = (\mathsf{pp}_{\mathsf{hPKE}} = (p, \ell_1, \ldots, \ell_n, \mu, E_0, \mathcal{H} = \{H_k\}_{k\in\mathcal{K}}), \mathsf{KDF})$ and an updated ciphertext up_{i+1}, a user u with its secret key $\mathsf{sk}_i^{(\mathsf{u})} = \boldsymbol{a}_i$ runs this algorithm to generate updated secret key $\mathsf{sk}_{i+1}^{(\mathsf{u})} = \boldsymbol{a}_{i+1}$ by executing the following steps:

i. Decrypts the updated ciphertext $\boldsymbol{\rho}_i = \mathsf{UhPKE.Dec}(\mathsf{pp}_{\mathsf{UhPKE}}, \mathsf{sk}_i^{(\mathsf{u})}, \mathsf{up}_{i+1} = (\mathsf{ct}_{\boldsymbol{\rho}_i}^{(1)} = [\mathbf{b}']E_0, \mathsf{ct}_{\boldsymbol{\rho}_i}^{(2)} = \boldsymbol{\rho}_i \oplus H_k(M_C([\mathbf{b}']\mathsf{pk}_i^{(\mathsf{u})})))$ where $\boldsymbol{\rho}_i \in \{0,1\}^{\mathsf{mlen}(\lambda)}$ and $\mathbf{b}' \xleftarrow{\$} [-\mu, \mu]^n$.

ii. Computes updated secret key $\mathsf{sk}_{i+1}^{(\mathsf{u})} = \boldsymbol{a}_{i+1} = \mathsf{sk}_i^{(\mathsf{u})} - \mathsf{KDF}(\boldsymbol{\rho}_i) = \boldsymbol{a}_i - \mathsf{KDF}(\boldsymbol{\rho}_i)$.

iii. Returns updated secret key $\mathsf{sk}_{i+1}^{(\mathsf{u})} = \boldsymbol{a}_{i+1}$.

Correctness. Note that, $(\mathsf{sk}_0^{(\mathsf{u})} = \boldsymbol{a}_0, \mathsf{pk}_0^{(\mathsf{u})} = E_{A_0})$ is generated by running $\mathsf{UhPKE.KeyGen}(\mathsf{pp}_{\mathsf{UhPKE}})$ satisfying $\mathsf{pk}_0^{(\mathsf{u})} = E_{A_0} = [\boldsymbol{a}_0]E_0 = [\mathsf{sk}_0^{(\mathsf{u})}]E_0$. Also note that, $\mathsf{sk}_{i+1}^{(\mathsf{u})} = \mathsf{sk}_i^{(\mathsf{u})} - \mathsf{KDF}(\boldsymbol{\rho}_i)$ and $\mathsf{pk}_{i+1}^{(\mathsf{u})} = [-\mathsf{KDF}(\boldsymbol{\rho}_i)]\mathsf{pk}_i^{(\mathsf{u})}$. We will prove $\mathsf{pk}_i^{(\mathsf{u})} = [\mathsf{sk}_i^{(\mathsf{u})}]E_0$ for all $i \geq 0$ by mathematical induction:

For $i = 0$ we have $\mathsf{pk}_0^{(\mathsf{u})} = [\mathsf{sk}_0^{(\mathsf{u})}]E_0$. Let $\mathsf{pk}_i^{(\mathsf{u})} = [\mathsf{sk}_i^{(\mathsf{u})}]E_0$ then $[\mathsf{sk}_{i+1}^{(\mathsf{u})}]E_0 = [\mathsf{sk}_i^{(\mathsf{u})} - \mathsf{KDF}(\boldsymbol{\rho}_i)]E_0 = [\mathsf{sk}_i^{(\mathsf{u})}][-\mathsf{KDF}(\boldsymbol{\rho}_i)]E_0 = [\mathsf{sk}_i^{(\mathsf{u})}][-\mathsf{KDF}(\boldsymbol{\rho}_i)]E_0 = [-\mathsf{KDF}(\boldsymbol{\rho}_i)][\mathsf{sk}_i^{(\mathsf{u})}]E_0 = [-\mathsf{KDF}(\boldsymbol{\rho}_i)]\mathsf{pk}_i^{(\mathsf{u})} = \mathsf{pk}_{i+1}^{(\mathsf{u})}$. Hence, $\mathsf{pk}_i^{(\mathsf{u})} = [\mathsf{sk}_i^{(\mathsf{u})}]E_0$ for all $i \geq 0$. By the correctness of hPKE (as described in Appendix A.1), it follows that $\Pr[\mathsf{UhPKE.Dec}(\mathsf{pp}_{\mathsf{UhPKE}}, \mathsf{sk}_i^{(\mathsf{u})}, \mathsf{UhPKE.Enc}(\mathsf{pp}_{\mathsf{UhPKE}}, \mathsf{pk}_i^{(\mathsf{u})}, m)) = m] = 1$ ensuring the correctness of UhPKE is established.

For the security proof of our updatable public key encryption scheme based on hPKE, we need to establish that hPKE is $f\text{-}\mathsf{CS} + \mathsf{LR}$ secure. This is formally stated in Theorem 38 with the corresponding proof presented in Appendix 3.

Theorem 31. *The construction* hPKE *provides* $f\text{-}\mathsf{CS} + \mathsf{LR}$ *security for* $f : [-\mu, \mu]^n \times \{0,1\}^{\mathsf{mlen}(\lambda)} \to [-\mu, \mu]^n$ *under the* $\mathsf{CSSIDDH}$ *assumption given in Definition 13,*

assuming $\mathcal{H} = \{H_k\}_{k\in\mathcal{K}}$ *is entropy smoothing function and* KDF *is a secure key derivation function as per Definition 16.*

Theorem 32. *If* hPKE *described in Appendix A.1 is circular secure and leakage-resilient* (f-$\mathsf{CS}+\mathsf{LR}$) *secure as per Definition 20 for* $f : [-\mu,\mu]^n \times \{0,1\}^{\mathsf{mlen}(\lambda)} \to [-\mu,\mu]^n$ *then the isogeny based* UhPKE *construction is* $\mathsf{IND\text{-}CR\text{-}CPA}$ *secure as per Definition 23.*

Proof If an adversary $\mathcal{A}$ wins the experiment $\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{UhPKE},\mathcal{A}}(\lambda)$ (see Fig. 3) then we will show how to construct an adversary $\mathcal{B}$ that uses the adversary $\mathcal{A}$ as a subroutine and wins the experiment $\mathsf{Exp}^{f\text{-}\mathsf{CS}+\mathsf{LR}}_{\mathsf{hPKE},\mathcal{B}}(\lambda)$ (see Fig. 2) with $\mathsf{Adv}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{UhPKE},\mathcal{A}}(\lambda) = \mathsf{Adv}^{f\text{-}\mathsf{CS}+\mathsf{LR}}_{\mathsf{hPKE},\mathcal{B}}(\lambda)$.

In the experiment $\mathsf{Exp}^{f\text{-}\mathsf{CS}+\mathsf{LR}}_{\mathsf{hPKE},\mathcal{B}}(\lambda)$, the hPKE challenger $\mathcal{C}$ generates the public parameter $\mathsf{pp}_{\mathsf{hPKE}} = (p, \ell_1, \ldots, \ell_n, \mu, E_0, \mathcal{H} = \{H_k\}_{k\in\mathcal{K}})$ by running $\mathsf{hPKE.Setup}(\lambda)$ and a secret-public key pair ($\mathsf{sk} = \boldsymbol{a}$, $\mathsf{pk} = E_A = [\boldsymbol{a}]E_0$) by running $\mathsf{hPKE.KeyGen}(\mathsf{pp}_{\mathsf{hPKE}})$. It provides $\mathsf{pp}_{\mathsf{hPKE}}$ and pk to the adversary $\mathcal{B}$. The hPKE adversary $\mathcal{B}$ proceeds to simulate the experiment $\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{UhPKE},\mathcal{A}}(\lambda)$ for the UhPKE adversary $\mathcal{A}$ in the following manner.

Setup: The adversary $\mathcal{B}$ initializes a list $\mathsf{rList} = \phi$ and epoch $i = 0$ and also takes a key derivation function $\mathsf{KDF} : \{0,1\}^{\mathsf{mlen}(\lambda)} \to [-\mu,\mu]^n$. It sends $\mathsf{pp}_{\mathsf{UhPKE}} = (\mathsf{pp}_{\mathsf{hPKE}}, \mathsf{KDF})$ with $\mathsf{pp}_{\mathsf{hPKE}} = (p, \ell_1, \ldots, \ell_n, \mu, E_0, \mathcal{H} = \{H_k\}_{k\in\mathcal{K}})$ and $\mathsf{pk}_0^{(\mathsf{u})} = \mathsf{pk}$ to the adversary $\mathcal{A}$.

Simulation of the pre-challenge oracle query: The adversary $\mathcal{B}$ issues polynomially many adaptive queries to the oracle $\mathcal{O}^{\mathsf{UhPKE}}_{\mathsf{up}}(\cdot)$ which are simulated by $\mathcal{B}$ perfectly for $\mathcal{A}$ as follows.

– $\mathcal{O}^{\mathsf{UhPKE}}_{\mathsf{up}}(\boldsymbol{\rho}_i)$: Upon receiving a query on $\boldsymbol{\rho}_i \in \{0,1\}^{\mathsf{mlen}(\lambda)}$ at the i-th epoch, the adversary $\mathcal{B}$ computes $\mathbf{r}_i = \mathsf{KDF}(\boldsymbol{\rho}_i)$, updates the list $\mathsf{rList} = \mathsf{rList} \cup \{(i, \mathbf{r}_i)\}$, runs $\mathsf{UhPKE.UpdatePk}(\mathsf{pp}_{\mathsf{UhPKE}}, \mathsf{pk}_i^{(\mathsf{u})}; \boldsymbol{\rho}_i) \to (\mathsf{pk}_{i+1}^{(\mathsf{u})} = [-\mathbf{r}_i]\mathsf{pk}_i^{(\mathsf{u})}, \mathsf{up}_{i+1})$ with $\mathsf{up}_{i+1} = (\mathsf{ct}^{(1)}_{\boldsymbol{\rho}_i} = [\mathbf{b}']E_0,\ \mathsf{ct}^{(2)}_{\boldsymbol{\rho}_i} = \boldsymbol{\rho}_i \oplus H_k(M_C([\mathbf{b}']\mathsf{pk}_i^{(\mathsf{u})})))$ $\leftarrow \mathsf{UhPKE.Enc}(\mathsf{pp}_{\mathsf{UhPKE}}, \mathsf{pk}_i^{(\mathsf{u})}, \boldsymbol{\rho}_i)$ choosing some $\mathbf{b}' \in [-\mu,\mu]^n$. It increments the counter i to $i+1$. Note that, $\mathsf{pk}_i^{(\mathsf{u})} = E_{A_i}$ implies $\mathsf{pk}_{i+1}^{(\mathsf{u})} = [-\mathbf{r}_i]\mathsf{pk}_i^{(\mathsf{u})} = [-\sum_{j=0}^{i-1}\mathbf{r}_j]\mathsf{pk}_0^{(\mathsf{u})} = [-\sum_{j=0}^{i-1}\mathbf{r}_j][\boldsymbol{a}]E_0 = [\boldsymbol{a} - \sum_{j=0}^{i-1}\mathbf{r}_j]E_0 = [\boldsymbol{a}_i]E_0 = E_{A_{i+1}}$ implicitly setting $\boldsymbol{a}_i = \boldsymbol{a} - \sum_{j=0}^{i-1}\mathbf{r}_j$.

Simulation of the challenge phase: After receiving a challenge message pair (m_0^*, m_1^*) from the UhPKE adversary $\mathcal{A}$, the hPKE adversary $\mathcal{B}$ also forwards (m_0^*, m_1^*) to its own challenger $\mathcal{C}$. In response, the challenger $\mathcal{C}$ performs the following steps

i. Uniformly samples $\boldsymbol{\rho}^* \in \{0,1\}^{\mathsf{mlen}}$, computes $\mathbf{r}^* = \mathsf{KDF}(\boldsymbol{\rho}^*)$ and sets $\mathbf{z} = \mathsf{sk} - \mathbf{r}^* = \boldsymbol{a} - \mathbf{r}^*$.
ii. Computes $\mathsf{ct}_{m_b^*} \leftarrow \mathsf{hPKE.Enc}(\mathsf{pp}_{\mathsf{hPKE}}, \mathsf{pk}_0^{(\mathsf{u})} = \mathsf{pk}, m_b^*)$ with $\mathsf{ct}_{m_b^*} = (\mathsf{ct}^{(1)}_{m_b^*} = [\mathbf{b}]E_0,\ \mathsf{ct}^{(2)}_{m_b^*} = m_b^* \oplus H_k(M_C([\mathbf{b}]E_A)))$ for some uniformly element $\mathbf{b} \in [-\mu,\mu]^n$.
iii. Computes $\mathsf{ct}_{\boldsymbol{\rho}^*} = \mathsf{hPKE.Enc}(\mathsf{pp}_{\mathsf{hPKE}}, \mathsf{pk}_0^{(\mathsf{u})} = \mathsf{pk}, \boldsymbol{\rho}^*)$ with $\mathsf{ct}_{\boldsymbol{\rho}^*} = (\mathsf{ct}^{(1)}_{\boldsymbol{\rho}^*} = [\mathbf{b}']E_0,\ \mathsf{ct}^{(2)}_{\boldsymbol{\rho}^*} = \boldsymbol{\rho}^* \oplus H_k(M_C([\mathbf{b}']E_A)))$ for some uniformly element $\mathbf{b}' \in [-\mu,\mu]^n$.
iv. Sends $(\mathbf{z}, \mathsf{ct}_{m_b^*}, \mathsf{ct}_{\boldsymbol{\rho}^*})$ to $\mathcal{B}$.

To simulate a challenge ciphertext for UhPKE adversary $\mathcal{A}$ at epoch t in which $\mathcal{A}$ issued its challenge messages, the adversary $\mathcal{B}$ computes $\mathbf{r} = \sum_{i=0}^{t-1}\mathbf{r}_i$ by extracting $\mathbf{r}_i$ from its maintained list rList for $i = 0, 1, \ldots, t-1$ and sets the challenge ciphertext as $\widetilde{\mathsf{ct}}_{m_b^*} = (\widetilde{\mathsf{ct}}^{(1)}_{m_b^*} = [\mathbf{r}]\mathsf{ct}^{(1)}_{m_b^*}, \widetilde{\mathsf{ct}}^{(2)}_{m_b^*} = \mathsf{ct}^{(2)}_{m_b^*})$ by extracting $\mathsf{ct}^{(1)}_{m_b^*} = [\mathbf{b}]E_0$ and $\mathsf{ct}^{(2)}_{m_b^*} = m_b^* \oplus H_k(M_C([\mathbf{b}]E_A))$ from

$\mathsf{ct}_{m_b^*} = (\mathsf{ct}^{(1)}_{m_b^*}, \mathsf{ct}^{(2)}_{m_b^*})$ sent by the hPKE challenger $\mathcal{C}$ to $\mathcal{B}$. The adversary $\mathcal{B}$ then forwards $\widetilde{\mathsf{ct}}_{m_b^*}$ as the challenge ciphertext to $\mathcal{A}$.

Note that $\widetilde{\mathsf{ct}}_{m_b^*}$ is a valid UhPKE encryption of m_b^* under the public key $\mathsf{pk}_t^{(\mathsf{u})}$ as $\mathsf{pk}_t^{(\mathsf{u})} = E_{A_t} = [\boldsymbol{a}_t]E_0 = [\boldsymbol{a} - \sum_{i=0}^{t-1} \mathbf{r}_i]E_0 = [-\mathbf{r}]\mathsf{pk}_0^{(\mathsf{u})}$
and $\widetilde{\mathsf{ct}}_{m_b^*} = ([\mathbf{r}]\mathsf{ct}^{(1)}_{m_b^*}, \mathsf{ct}^{(2)}_{m_b^*}) = ([\mathbf{b}+\mathbf{r}]E_0, m_b^* \oplus H_k(M_C([\mathbf{b}+\boldsymbol{a}]E_0))) = ([\mathbf{b}+\mathbf{r}]E_0, m_b^* \oplus H_k(M_C([\mathbf{b}+\mathbf{r}+\boldsymbol{a}-\mathbf{r}]E_0))) = ([\mathbf{b}+\mathbf{r}]E_0, m_b^* \oplus H_k(M_C([\mathbf{b}+\mathbf{r}][\boldsymbol{a}-\mathbf{r}]E_0))) = ([\mathbf{b}+\mathbf{r}]E_0, m_b^* \oplus H_k(M_C([\mathbf{b}+\mathbf{r}]\mathsf{pk}_t^{(\mathsf{u})}))) = ([\tilde{\mathbf{b}}]E_0, m_b^* \oplus H_k(M_C([\tilde{\mathbf{b}}]E_{A_t})))$ where $\tilde{\mathbf{b}} = \mathbf{b} + \mathbf{r}$ which remains uniformly distributed in $[-\mu, \mu]^n$ since $\mathbf{b} \xleftarrow{\$} [-\mu, \mu]^n$. Thus $\widetilde{\mathsf{ct}}_{m_b^*}$ has the same distribution generated by $\mathsf{UhPKE.Enc}(\mathsf{pp}_{\mathsf{UhPKE}}, \mathsf{pk}_t^{(\mathsf{u})}, m_b^*)$ as the real protocol, demonstrating the correctness of the simulated ciphertext under the UhPKE encryption scheme.

Simulation of the post-challenge oracle query: The hPKE adversary $\mathcal{B}$ simulates the oracle exactly in the same as in the pre-challenge oracle query to $\mathcal{O}_{\mathsf{up}}^{\mathsf{UhPKE}}(\cdot)$.

Simulation of the reveal phase: Let $\mathcal{O}_{\mathsf{up}}^{\mathsf{UhPKE}}(\boldsymbol{\rho}_{t'})$ is the final query issued by the UhPKE adversary $\mathcal{A}$. The hPKE adversary $\mathcal{B}$ is required to output the final public key, secret key and updated ciphertext $(\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{up}^*)$ where $\mathsf{sk}^* \leftarrow \mathsf{UhPKE.UpdateSk}(\mathsf{pp}_{\mathsf{UhPKE}}, \mathsf{sk}_{t'}^{(\mathsf{u})}, \mathsf{up}^*)$ and $(\mathsf{pk}^*, \mathsf{up}^*) \leftarrow \mathsf{UhPKE.UpdatePk}(\mathsf{pp}_{\mathsf{UhPKE}}, \mathsf{pk}_{t'}^{(\mathsf{u})})$ which represents the cumulative effect of all t' update queries made by $\mathcal{A}$ along with a final update whose randomness remains unknown to $\mathcal{A}$. Here, $\mathsf{pk}_{t'}^{(\mathsf{u})} = E_{A_{t'}} = [\boldsymbol{a}_{t'}]E_0 = [\boldsymbol{a} - \sum_{i=0}^{t'-1} \mathbf{r}_i]E_0 = [-\mathbf{r}'][\boldsymbol{a}]\mathsf{pk}_0^{(\mathsf{u})} = [-\mathbf{r}']E_A$ which implicitly sets $\mathsf{sk}_{t'}^{(\mathsf{u})} = \boldsymbol{a}_{t'} = \boldsymbol{a} - \sum_{i=0}^{t'-1} \mathbf{r}_i = \boldsymbol{a} - \mathbf{r}'$ with $\mathbf{r}' = \sum_{i=0}^{t'-1} \mathbf{r}_i$.

To simulate $(\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{up}^*)$, the adversary $\mathcal{B}$ employs $\mathbf{z} = \boldsymbol{a} - \mathbf{r}^*$ and the ciphertext $\mathsf{ct}_{\boldsymbol{\rho}^*} \leftarrow \mathsf{hPKE.Enc}(\mathsf{pp}_{\mathsf{hPKE}}, \mathsf{pk}_0^{(\mathsf{u})} = \mathsf{pk}, \boldsymbol{\rho}^*)$ received from its own challenger $\mathcal{C}$ with $\mathsf{ct}_{\boldsymbol{\rho}^*} = (\mathsf{ct}^{(1)}_{\boldsymbol{\rho}^*}, \mathsf{ct}^{(2)}_{\boldsymbol{\rho}^*})$, $\mathsf{ct}^{(1)}_{\boldsymbol{\rho}^*} = [\mathbf{b}']E_0$, $\mathsf{ct}^{(2)}_{\boldsymbol{\rho}^*} = \boldsymbol{\rho}^* \oplus H_k(M_C([\mathbf{b}']E_A))$, $\mathbf{b}, \mathbf{b}' \xleftarrow{\$} [-\mu, \mu]^n$ and $\mathbf{r}^* = \mathsf{KDF}(\boldsymbol{\rho}^*)$ for some $\boldsymbol{\rho}^* \in \{0,1\}^{\mathsf{mlen}}$ (unknown to $\mathcal{B}$) as follows.

The adversary $\mathcal{B}$ computes $\mathbf{r}' = \sum_{i=0}^{t'-1} \mathbf{r}_i$ by extracting $\mathbf{r}_i$ for $i = 0, 1, \ldots, t'-1$ from its maintained list rList, sets secret key sk^* at epoch $t'+1$ as $\mathsf{sk}^* = \mathbf{z} - \mathbf{r}'$ and defines the public key at epoch $t'+1$ as $\mathsf{pk}^* = [\mathsf{sk}^*]E_0$.

Note that $\mathsf{sk}^* = \mathbf{z} - \mathbf{r}' = \boldsymbol{a} - \mathbf{r}^* - \mathbf{r}' = \mathsf{sk}_{t'}^{(\mathsf{u})} - \mathbf{r}^* = \boldsymbol{a}_{t'} - \mathbf{r}^*$ which has the same distribution as that in the real protocol by executing $\mathsf{UhPKE.UpdateSk}(\mathsf{pp}_{\mathsf{UhPKE}}, \mathsf{sk}_{t'}^{(\mathsf{u})}, \mathsf{up}^*)$. Moreover, pk^* is set to be $[\mathsf{sk}^*]E_0$ which is identically distributed as in the real protocol by invoking $\mathsf{UhPKE.UpdatePk}(\mathsf{pp}_{\mathsf{UhPKE}}, \mathsf{pk}_{t'}^{(\mathsf{u})}, \boldsymbol{\rho}^*)$. Thus, $\mathbf{z} = \boldsymbol{a} - \mathbf{r}^*$ sent by $\mathcal{C}$ to $\mathcal{B}$ is sufficient for $\mathcal{B}$ to simulates sk^* and pk^* without knowing explicitly ρ^* or $\mathbf{r}^* = \mathsf{KDF}(\boldsymbol{\rho}^*)$. To compute updated ciphertext up^*, the adversary $\mathcal{B}$ sets $\mathsf{up}^* = (\widetilde{\mathsf{ct}}^{(1)}_{\boldsymbol{\rho}^*} = [\mathbf{r}']\mathsf{ct}^{(1)}_{\boldsymbol{\rho}^*}, \widetilde{\mathsf{ct}}^{(2)}_{\boldsymbol{\rho}^*} = \mathsf{ct}^{(2)}_{\boldsymbol{\rho}^*})$ by extracting $\mathsf{ct}^{(1)}_{\boldsymbol{\rho}^*} = ([\mathbf{b}']E_0$ and $\mathsf{ct}^{(2)}_{\boldsymbol{\rho}^*} = \boldsymbol{\rho}^* \oplus H_k([\mathbf{b}' + \mathbf{r}' + \boldsymbol{a} - \mathbf{r}']E_0)$.
Finally, $(\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{up}^*)$ is forwarded to $\mathcal{A}$. We now establish that up^* is a valid encryption of $\boldsymbol{\rho}^*$ under $\mathsf{pk}_{t'}^{(\mathsf{u})}$. Note that, $\mathsf{up}^* = (\widetilde{\mathsf{ct}}^{(1)}_{\boldsymbol{\rho}^*} \widetilde{\mathsf{ct}}^{(2)}_{\boldsymbol{\rho}^*}) = ([\mathbf{r}']\mathsf{ct}^{(1)}_{\boldsymbol{\rho}^*}, \mathsf{ct}^{(2)}_{\boldsymbol{\rho}^*}) = ([\mathbf{b}' + \mathbf{r}']E_0, \boldsymbol{\rho}^* \oplus H_k(M_C([\mathbf{b}' + \boldsymbol{a}]E_0))) = ([\mathbf{b}' + \mathbf{r}']E_0, \boldsymbol{\rho}^* \oplus H_k(M_C([\mathbf{b}' + \mathbf{r}' + \boldsymbol{a} - \mathbf{r}']E_0))) = ([\mathbf{b}' + \mathbf{r}']E_0, \boldsymbol{\rho}^* \oplus H_k(M_C([\mathbf{b}' + \mathbf{r}']\mathsf{pk}_{t'}^{(\mathsf{u})})))$. Defining $\tilde{\mathbf{b}}' = \mathbf{b}' + \mathbf{r}'$, we observe that $\mathbf{b}' \xleftarrow{\$} [-\mu, \mu]^n$, it follows that $\tilde{\mathbf{b}}'$ remains uniformly distributed in $[-\mu, \mu]^n$, ensuring up^* has identical distribution as that in the real protocol generated by algorithm $\mathsf{UhPKE.Enc}(\mathsf{pp}_{\mathsf{UhPKE}}, \mathsf{pk}_{t'}^{(\mathsf{u})}, \boldsymbol{\rho}^*)$. Thus,

$\mathsf{ct}_{m_b^*} = (\mathsf{ct}_{m_b^*}^{(1)} = [\mathbf{b}]E_0, \mathsf{ct}_{m_b^*}^{(2)} = m_b^* \oplus H_k(M_C([\mathbf{b}]E_A)))$ sent by $\mathcal{C}$ to $\mathcal{B}$ is sufficient for $\mathcal{B}$ to simulates up^* without any explicit knowledge of $\boldsymbol{\rho}^*$ or $\mathbf{r}^* = \mathsf{KDF}(\boldsymbol{\rho}^*)$. This validates the correctness of the simulated updated ciphertext up^*.

Guess Phase: Upon receiving $b' \in \{0,1\}$ from the adversary $\mathcal{A}$, the adversary $\mathcal{B}$ submits b' to its own challenger $\mathcal{C}$.

Observe that $\mathcal{B}$ perfectly simulates $\mathcal{A}$'s challenger in the IND-CR-CPA security game of Fig. 3 and succeeds whenever $\mathcal{A}$ does. Thus

$$\mathsf{Adv}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{UhPKE},\mathcal{A}}(\lambda) = \left|\Pr[\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{UhPKE},\mathcal{A}}(\lambda) = 1] - \tfrac{1}{2}\right|$$

$$= \left|\Pr[\mathsf{Exp}^{f\text{-}\mathsf{CS}+\mathsf{LR}}_{\mathsf{hPKE},\mathcal{B}}(\lambda) = 1] - \tfrac{1}{2}\right| = \mathsf{Adv}^{f\text{-}\mathsf{CS}+\mathsf{LR}}_{\mathsf{hPKE},\mathcal{B}}(\lambda).$$

Hence, Theorem 32 follows. □

5.2 Construction 2: USimS

Our second construction of updatable public key encryption $\mathsf{USimS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{UpdatePk}, \mathsf{UpdateSk})$ associated with a message space $\mathcal{MS} = \mathbb{Z}_{2^{q-2}}$ is based on $\mathsf{SimS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ satisfying the following requirements:

$\mathsf{USimS.Setup}(\lambda) \to \mathsf{pp}_{\mathsf{USimS}} = (\mathsf{pp}_{\mathsf{SimS}} = (p, q, \ell_1, \ldots, \ell_n, \mu, E_0, R_E), \mathsf{KDF})$: A trusted party runs this algorithm on the input the security parameter λ and generates the public parameter $\mathsf{pp}_{\mathsf{USimS}} = (\mathsf{pp}_{\mathsf{SimS}} = (p, q, \ell_1, \ldots, \ell_n, \mu, E_0, R_E), \mathsf{KDF})$ where $\mathsf{KDF} : \mathbb{Z}_{2^q}^{\times} \to [-\mu, \mu]^n$ and $\mathsf{pp}_{\mathsf{SimS}} \leftarrow \mathsf{SimS.Setup}(\lambda)$ is a key derivation function. More specifically, the following steps are executed.

i. Chooses a prime $p = 2^q \Pi_{i=1}^n \ell_i - 1$ such that $\lambda + 2 \le q \le \frac{1}{2}\log p$ and ℓ_i's are small distinct odd primes.
ii. Sets a base curve $E_0 : y^2 = x^3 + x \in \mathsf{Ell}_p(\mathcal{O})$ over $\mathbb{F}_p$ with $\mathcal{O} = \mathbb{Z}[\sqrt{-p}]$ and picks an integer μ such that $(2\mu+1)^n \ge \#\mathsf{Cl}(\mathcal{O})$.
iii. Consider the function $R_E : \mathbb{F}_p \to \mathbb{F}_p$ defined as $R_E(x) = \mathsf{int}(\mathsf{bin}(x) \oplus \mathsf{bin}(M_C(E)))$ where $\mathsf{bin}(\cdot)$ and $\mathsf{int}(\cdot)$ represent the operations that convert an element of $\mathbb{F}_p$ into its binary representation and back, respectively. Here, $M_C(E)$ denotes the Montgomery coefficient associated with the elliptic curve E.
iv. Consider a key derivation function $\mathsf{KDF} : \mathbb{Z}_{2^q}^{\times} \to [-\mu, \mu]^n$.
v. Outputs the public parameter $\mathsf{pp}_{\mathsf{USimS}} = (\mathsf{pp}_{\mathsf{SimS}} = (p, q, \ell_1, \ldots, \ell_n, \mu, E_0, R_E), \mathsf{KDF})$.

$\mathsf{USimS.KeyGen}(\mathsf{pp}_{\mathsf{USimS}}) \to (\mathsf{sk}_0^{(\mathsf{u})} = \boldsymbol{a}_0, \mathsf{pk}_0^{(\mathsf{u})} = E_{A_0})$: Similar to $\mathsf{SimS.KeyGen}(\mathsf{pp}_{\mathsf{SimS}})$ on input the public parameter $\mathsf{pp}_{\mathsf{USimS}} = (\mathsf{pp}_{\mathsf{SimS}} = (p, q, \ell_1, \ldots, \ell_n, \mu, E_0, R_E), \mathsf{KDF})$, a user generates a secret-public key pair $(\mathsf{sk}_0^{(\mathsf{u})}, \mathsf{pk}_0^{(\mathsf{u})})$ in the following manner:

i. Samples $\boldsymbol{a}_0 = (a_1^0, \ldots, a_n^0) \xleftarrow{\$} [-\mu, \mu]^n$ and define $[\boldsymbol{a}_0] = [\mathfrak{l}_1^{a_1^0} \cdots \mathfrak{l}_n^{a_n^0}] \in \mathsf{Cl}(\mathcal{O})$ where $\mathfrak{l}_i = \langle \ell_i, \pi - 1 \rangle$. Computes $E_{A_0} = [\boldsymbol{a}_0]E_0$.
ii. Returns public key $\mathsf{pk}_0^{(\mathsf{u})} = E_{A_0}$ and keeps secret $\mathsf{sk}_0^{(\mathsf{u})} = \boldsymbol{a}_0$ to itself.

$\mathsf{USimS.Enc}(\mathsf{pp}_{\mathsf{USimS}}, \mathsf{pk}_i^{(\mathsf{u})}, m) \to \mathsf{ct}_m$: Taking input the public parameter $\mathsf{pp}_{\mathsf{USimS}} = (p, q, \ell_1, \ldots, \ell_n, \mu, E_0, R_E, \mathsf{KDF})$, public key $\mathsf{pk}_i^{(\mathsf{u})} = E_{A_i}$ and a message $m \in \mathcal{MS}$, an encrypter proceeds to compute ciphertext as in the $\mathsf{SimS.Enc}(\mathsf{pp}_{\mathsf{SimS}}, \mathsf{pk}_i^{(\mathsf{u})}, m)$.

i. Embeds $m \in \mathbb{Z}_{2^q}^{\times}$ via $m \to 2m+1$. Randomly samples $\mathbf{b} = (b_1, \ldots, b_n) \xleftarrow{\$} [-\mu, \mu]^n$ and define $[\mathbf{b}] = [\mathfrak{l}_1^{b_1} \cdots \mathfrak{l}_n^{b_n}] \in \mathsf{Cl}(\mathcal{O})$.
ii. Computes $E_B = [\mathbf{b}]E_0$, $E_{AB} = [\mathbf{b}]E_{A_i}$ and $P_{AB} = [2m+1]P_{E_{AB}}$. Here, $P_{E_{AB}}$ is a point on E_{AB} of order 2^q, which is determined using Algorithm 3.
iii. Returns a ciphertext $\mathsf{ct}_m = (E_B, R_{E_{AB}}(x(P_{AB})))$ with $R_{E_{AB}}(x(P_{AB}))) = \mathsf{int}(\mathsf{bin}(x(P_{AB})) \oplus \mathsf{bin}(M_C(E_{AB})))$.

$\mathsf{USimS.Dec}(\mathsf{pp}_{\mathsf{USimS}}, \mathsf{sk}_i^{(\mathsf{u})} = \boldsymbol{a}_i, \mathsf{ct}_m = (\mathsf{ct}_m^{(1)}, \mathsf{ct}_m^{(2)})) \to m/\perp$: Similar to $\mathsf{SimS.Dec}(\mathsf{pp}_{\mathsf{SimS}}, \mathsf{sk}_i^{(\mathsf{u})}, \mathsf{ct}_m)$ decrypter runs this deterministic algorithm on input the public parameter $\mathsf{pp}_{\mathsf{USimS}} = (\mathsf{pp}_{\mathsf{SimS}} = (p, q, \ell_1, \ldots, \ell_n, \mu, E_0, R_E), \mathsf{KDF})$, ciphertext $\mathsf{ct}_m = (E_B, x')$ and it's secret key $\mathsf{sk}_i^{(\mathsf{u})} = \boldsymbol{a}_i$ to recover the plaintext m in as follows:
i. Verifies that E_B is a supersingular curve,
ii. Computes $E_{BA} = [\boldsymbol{a}_i]E_B$ and a point $P_{E_{BA}}$ on E_{BA} of order 2^q by using Algorithm 3.
iii. Computes $R_{E_{BA}}(x')$ and if $R_{E_{BA}}(x')$ is not the x-coordinate of a 2^q-torsion point on the curve E_{BA} then aborts.
iv. Solves the discrete logarithm instance between $P_{BA} = (R_{E_{BA}}(x'), \cdot)$ and $P_{E_{BA}}$ using the Pohlig-Hellman algorithm given in Algorithm 2. Let $m' \in \mathbb{Z}_{2^q}^{\times}$ be the solution of this computation. If $2^{q-1} < m'$ then it changes m' to $2^q - m'$ and returns the plaintext $\frac{(m'-1)}{2}$.

$\mathsf{USimS.UpdatePk}(\mathsf{pp}_{\mathsf{USimS}}, \mathsf{pk}_i^{(\mathsf{u})}, \boldsymbol{\rho}_i \in \mathbb{Z}_{2^{q-2}}) \to (\mathsf{pk}_{i+1}^{(\mathsf{u})}, \mathsf{up}_{i+1})$: Given the public parameter $\mathsf{pp}_{\mathsf{USimS}} = (\mathsf{pp}_{\mathsf{SimS}} = (p, q, \ell_1, \ldots, \ell_n, \mu, E_0, R_E), \mathsf{KDF})$ and a public key $\mathsf{pk}_i^{(\mathsf{u})}$, any user can run this algorithm and proceeds to compute an updated ciphertext up_{i+1} and a new public key $\mathsf{pk}_i^{(\mathsf{u})}$ in the following way:
i. Define $\mathsf{KDF}(\boldsymbol{\rho}_i) = \mathbf{r}_i = (r_1, \ldots, r_n)$ and $[\mathbf{r}_i] = [\mathfrak{l}_1^{r_1} \cdots \mathfrak{l}_n^{r_n}] \in \mathsf{Cl}(\mathcal{O})$ where $\mathfrak{l}_i = \langle \ell_i, \pi - 1 \rangle$.
ii. Computes $\mathsf{pk}_{i+1}^{(\mathsf{u})} = [-\mathbf{r}_i]\mathsf{pk}_i^{(\mathsf{u})}$ and updated ciphertext $\mathsf{up}_{i+1} \leftarrow \mathsf{USimS.Enc}(\mathsf{pp}_{\mathsf{USimS}}, \mathsf{pk}_i^{(\mathsf{u})}, \boldsymbol{\rho}_i)$.
iii. Returns updated public key $\mathsf{pk}_{i+1}^{(\mathsf{u})}$ and updated ciphertext up_{i+1}.

$\mathsf{USimS.UpdateSk}(\mathsf{pp}_{\mathsf{USimS}}, \mathsf{sk}_i^{(\mathsf{u})}, \mathsf{up}_{i+1}) \to \mathsf{sk}_{i+1}^{(\mathsf{u})}$: Given the public parameter $\mathsf{pp}_{\mathsf{USimS}} = (\mathsf{pp}_{\mathsf{SimS}} = (p, q, \ell_1, \ldots, \ell_n, \mu, E_0, R_E), \mathsf{KDF})$ and an updated ciphertext up_{i+1} an user with $\mathsf{sk}_i^{(\mathsf{u})}$ computes $\boldsymbol{\rho}_i = \mathsf{USimS.Dec}(\mathsf{pp}_{\mathsf{USimS}}, \mathsf{sk}_i^{(\mathsf{u})}, \mathsf{up}_{i+1})$ and $\mathsf{sk}_{i+1}^{(\mathsf{u})} = \mathsf{sk}_i^{(\mathsf{u})} - \mathsf{KDF}(\boldsymbol{\rho}_i)$ and returns updated secret key $\mathsf{sk}_{i+1}^{(\mathsf{u})}$.

Correctness. We can show that $[\mathsf{sk}_{i+1}^{(\mathsf{u})}]E_0 = \mathsf{pk}_{i+1}^{(\mathsf{u})}$ as in the correctness of UhPKE. By the correctness of SimS described in Section A.2 implies $\Pr[\mathsf{USimS.Dec}(\mathsf{pp}_{\mathsf{USimS}}, \mathsf{sk}_i^{(\mathsf{u})}, \mathsf{USimS.Enc}(\mathsf{pp}_{\mathsf{USimS}}, \mathsf{pk}_i^{(\mathsf{u})}, m)) = m] = 1$.

To prove the security of our updatable public key encryption scheme based on SimS, we first establish that SimS is f-CS + LR secure, as stated in Theorem 41 and proven in Appendix 4.

Theorem 33. *If* SimS *described in Appendix A.2 is circular secure and leakage-resilient* (f-CS + LR) *secure as per Definition 20 for* $f : [-\mu, \mu]^n \times \mathbb{Z}_{2^{q-2}} \to [-\mu, \mu]^n$

then the isogeny based UPKE *construction* USimS *described in Section 5.2 provides* IND-CR-CPA *security.*

Proof We will demonstrate that if the adversary $\mathcal{A}$ wins in $\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{USimS},\mathcal{A}}(\lambda)$ then we can construct an adversary $\mathcal{B}$ using the adversary $\mathcal{A}$ that can win in $\mathsf{Exp}^{f\text{-}\mathsf{CS+LR}}_{\mathsf{SimS},\mathcal{B}}(\lambda)$ and $\mathsf{Adv}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{USimS},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{f\text{-}\mathsf{CS+LR}}_{\mathsf{SimS},\mathcal{B}}(\lambda)$.

In the experiment $\mathsf{Exp}^{f\text{-}\mathsf{CS+LR}}_{\mathsf{SimS},\mathcal{B}}(\lambda)$, the SimS challenger $\mathcal{C}$ generates the public parameter $\mathsf{pp}_{\mathsf{SimS}} = (p, q, \ell_1, \ldots, \ell_n, \mu, E_0)$ by running $\mathsf{SimS.Setup}(\lambda)$ and a secret-public key pair $(\mathsf{sk} = \boldsymbol{a}, \mathsf{pk}^{(\mathsf{u})} = E_A = [\boldsymbol{a}]E_0)$ by running $\mathsf{SimS.KeyGen}(\mathsf{pp}_{\mathsf{SimS}})$. It provides $\mathsf{pp}_{\mathsf{USimS}}$ and $\mathsf{pk}^{(\mathsf{u})}$ to the adversary $\mathcal{B}$. The adversary $\mathcal{B}$ proceeds to simulate the experiment $\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{USimS},\mathcal{A}}(\lambda)$ in the following manner.

Setup: The adversary $\mathcal{B}$ initializes a list $\mathsf{rList} = \phi$ and epoch $i = 0$ and also takes a key derivation function $\mathsf{KDF} : \mathbb{Z}_{2^{q-2}} \to [-\mu, \mu]^n$. It sends $\mathsf{pp}_{\mathsf{USimS}} = (\mathsf{pp}_{\mathsf{SimS}}, \mathsf{KDF}) = (p, \ell_1, \ldots, \ell_n, \mu, E_0, \mathsf{KDF})$ and $\mathsf{pk}^{(\mathsf{u})}_0 = \mathsf{pk}^{(\mathsf{u})}$ to the adversary $\mathcal{A}$.

Simulation of the pre-challenge oracle query: The adversary $\mathcal{B}$ issues polynomially many adaptive queries to the oracle $\mathcal{O}^{\mathsf{UhPKE}}_{\mathsf{up}}(\cdot)$. In the following, we will demonstrate that $\mathcal{B}$ simulates the following oracle perfectly for $\mathcal{A}$.

– $\mathcal{O}^{\mathsf{UhPKE}}_{\mathsf{up}}(\boldsymbol{\rho}_i)$: Upon receiving a query on $\boldsymbol{\rho}_i \in \{0,1\}^{\mathsf{mlen}(\lambda)}$ at the i-th epoch, the adversary $\mathcal{B}$ computes $\mathbf{r}_i = \mathsf{KDF}(\boldsymbol{\rho}_i)$, updates the list $\mathsf{rList} = \mathsf{rList} \cup \{(i, \mathbf{r}_i)\}$, runs $\mathsf{USimS.UpdatePk}(\mathsf{pp}_{\mathsf{USimS}}, \mathsf{pk}^{(\mathsf{u})}_i, \boldsymbol{\rho}_i) \to (\mathsf{pk}^{(\mathsf{u})}_{i+1} = [-\mathbf{r}_i]\mathsf{pk}^{(\mathsf{u})}_i, \mathsf{up}_{i+1} \leftarrow \mathsf{USimS.Enc}(\mathsf{pp}_{\mathsf{USimS}}, \mathsf{pk}^{(\mathsf{u})}_i, \boldsymbol{\rho}_i))$ with $\mathsf{up}_{i+1} = (\mathsf{ct}^{(1)}_{\boldsymbol{\rho}_i} = [\mathbf{b}']E_0,\ \mathsf{ct}^{(2)}_{\boldsymbol{\rho}_i} = R_{E_{[\mathbf{b}']\mathsf{pk}^{(\mathsf{u})}_i}}(x([2\boldsymbol{\rho}_i + 1]P_{[\mathbf{b}']\mathsf{pk}^{(\mathsf{u})}_i})))$ for some $\mathbf{b}' \in [-\mu, \mu]^n$ and increments the counter i to $i+1$. Note that if $\mathsf{pk}^{(\mathsf{u})}_i = E_{A_i}$ then

$$\mathsf{pk}^{(\mathsf{u})}_{i+1} = [-\mathbf{r}_i]\mathsf{pk}^{(\mathsf{u})}_i = [-\sum_{j=0}^{i-1}\mathbf{r}_j]\mathsf{pk}^{(\mathsf{u})}_0 = [-\sum_{j=0}^{i-1}\mathbf{r}_j][\boldsymbol{a}]E_0$$

$$= [\boldsymbol{a} - \sum_{j=0}^{i-1}\mathbf{r}_j]E_0 = [\boldsymbol{a}_i]E_0 = E_{A_{i+1}}$$

implicitly, $\mathsf{sk}_i = \boldsymbol{a}_i = \boldsymbol{a} - \sum_{j=0}^{i-1}\mathbf{r}_j$

Simulation of the challenge phase: After receiving (m^*_0, m^*_1) from the adversary $\mathcal{A}$, the adversary $\mathcal{B}$ also forwards (m^*_0, m^*_1) to its challenger $\mathcal{C}$. In response, the challenger $\mathcal{C}$ computes in the following way

i. Samples $\mathbf{r}^* = \mathsf{KDF}(\boldsymbol{\rho}^*)$ and computes $\boldsymbol{\rho}^* \in \mathbb{Z}_{2^{q-2}}$ and $\mathbf{z} = \mathsf{sk} - \mathbf{r}^* = \boldsymbol{a} - \mathbf{r}^*$

ii. Computes $\mathsf{ct}_{m^*_b} = \mathsf{SimS.Enc}(\mathsf{pp}_{\mathsf{SimS}}, \mathsf{pk}^{(\mathsf{u})}_0 = \mathsf{pk}, m^*_b)$ with $\mathsf{ct}_{m^*_b} = (\mathsf{ct}^{(1)}_{m^*_b} = [\mathbf{b}]E_0,\ \mathsf{ct}^{(2)}_{m^*_b} = R_{[\mathbf{b}]E_A}(x([2m^*_b + 1]P_{[\mathbf{b}]E_A})))$ for some uniform element $\mathbf{b} \in [-\mu, \mu]^n$.

iii. Calculates $\mathsf{ct}_{\boldsymbol{\rho}^*} = \mathsf{SimS.Enc}(\mathsf{pp}_{\mathsf{SimS}}, \mathsf{pk}^{(\mathsf{u})}_0 = \mathsf{pk}, \boldsymbol{\rho}^*)$ with $\mathsf{ct}_{\boldsymbol{\rho}^*} = (\mathsf{ct}^{(1)}_{\boldsymbol{\rho}^*} = [\mathbf{b}']E_0,\ \mathsf{ct}^{(2)}_{\boldsymbol{\rho}^*} = R_{[\mathbf{b}']E_A}(x([2\boldsymbol{\rho}^* + 1]P_{[\mathbf{b}']E_A})))$ for some uniform element $\mathbf{b}' \in [-\mu, \mu]^n$.

iv. Sends the tuple $(\mathbf{z}, \mathsf{ct}_{m^*_b}, \mathsf{ct}_{\boldsymbol{\rho}^*})$ to $\mathcal{B}$.

To simulate a challenge ciphertext for the adversary $\mathcal{A}$ at the epoch t, in which $\mathcal{A}$ issued its challenge messages, the adversary $\mathcal{B}$ computes $\mathbf{r} = \sum_{i=0}^{t-1}\mathbf{r}_i$ extracting $\mathbf{r}_i$ for $i = 0, 1, \ldots, t-1$ from its maintained list rList. Subsequently, the challenge ciphertext is constructed as

$$\widetilde{\mathsf{ct}}_{m_b^*} = (\widetilde{\mathsf{ct}}^{(1)}_{m_b^*} = [\mathbf{r}]\mathsf{ct}^{(1)}_{m_b^*}, \widetilde{\mathsf{ct}}^{(2)}_{m_b^*} = \mathsf{ct}^{(2)}_{m_b^*}),$$

by using $\mathsf{ct}^{(1)}_{m_b^*} = [\mathbf{b}]E_0$, and $\mathsf{ct}^{(2)}_{m_b^*} = R_{[\mathbf{b}]E_A}(x([2m_b^* + 1]P_{[\mathbf{b}]E_A}))$ which is sent by the SimS challenger $\mathcal{C}$ to $\mathcal{B}$.
The adversary $\mathcal{B}$ then forwards $\widetilde{\mathsf{ct}}_{m_b^*}$ as the challenge ciphertext to $\mathcal{A}$.

Note that $\widetilde{\mathsf{ct}}_{m_b^*}$ is a valid USimS encryption of m_b^* under the public key $\mathsf{pk}_t^{(\mathsf{u})}$. Specifically, the public key at epoch t is

$$\mathsf{pk}_t^{(\mathsf{u})} = E_{A_t} = [\boldsymbol{a}_t]E_0 = [\boldsymbol{a} - \sum_{i=0}^{t-1} \mathbf{r}_i]E_0 = [-\mathbf{r}]\mathsf{pk}_0^{(\mathsf{u})}$$

and the ciphertext is

$$\begin{aligned}
\widetilde{\mathsf{ct}}_{m_b^*} &= ([\mathbf{r}]\mathsf{ct}^{(1)}_{m_b^*}, \mathsf{ct}^{(2)}_{m_b^*}) \\
&= ([\mathbf{b} + \mathbf{r}]E_0, R_{[\mathbf{b}+\boldsymbol{a}]E_0}(x([2m_b^* + 1]P_{[\mathbf{b}+\boldsymbol{a}]E_0}))) \\
&= ([\mathbf{b} + \mathbf{r}]E_0, R_{[\mathbf{b}+\mathbf{r}+\boldsymbol{a}-\mathbf{r}]E_0}(x([2m_b^* + 1]P_{[\mathbf{b}+\mathbf{r}+\boldsymbol{a}-\mathbf{r}]E_0}))) \\
&= ([\mathbf{b} + \mathbf{r}]E_0, R_{[\mathbf{b}+\mathbf{r}][\boldsymbol{a}-\mathbf{r}]E_0}(x([2m_b^* + 1]P_{[\mathbf{b}+\mathbf{r}][\boldsymbol{a}-\mathbf{r}]E_0}))) \\
&= ([\mathbf{b} + \mathbf{r}]E_0, m_b^* \oplus R_{[\mathbf{b}+\mathbf{r}]\mathsf{pk}_t^{(\mathsf{u})}}(x([2m_b^* + 1]P_{[\mathbf{b}+\mathbf{r}]\mathsf{pk}_t^{(\mathsf{u})}})) \\
&= ([\mathbf{b}]E_0, m_b^* \oplus R_{[\tilde{\mathbf{b}}]\mathsf{pk}_t^{(\mathsf{u})}}(x([2m_b^* + 1]P_{[\tilde{\mathbf{b}}]\mathsf{pk}_t^{(\mathsf{u})}})))
\end{aligned}$$

where $\tilde{\mathbf{b}} = \mathbf{b}+\mathbf{r}$ remains uniformly distributed in $[-\mu, \mu]^n$ as $\mathbf{b} \xleftarrow{\$} [-\mu, \mu]^n$. Thus $\widetilde{\mathsf{ct}}_{m_b^*}$ has the same distribution as the real protocol generated by$\widetilde{\mathsf{ct}}_{m_b^*} \leftarrow \mathsf{USimS.Enc}(\mathsf{pp}_{\mathsf{USimS}}, \mathsf{pk}_t^{(\mathsf{u})}, m_b^*)$, demonstrating the correctness of the simulated ciphertext under the USimS encryption scheme.

Simulation of the post-challenge oracle query: The USimS adversary $\mathcal{B}$ simulates the oracle exactly the same as in the pre-challenge oracle query to $\mathcal{O}_{\mathsf{up}}^{\mathsf{USimS}}(\cdot)$.

Simulation of the reveal phase: Let $\mathcal{O}_{\mathsf{up}}^{\mathsf{USimS}}(\boldsymbol{\rho}_{t'})$ is the final query issued by the adversary $\mathcal{A}$. The adversary $\mathcal{B}$ is required to output the final public key, secret key and updated ciphertext $(\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{up}^*)$ where $(\mathsf{pk}^*, \mathsf{up}^*) \leftarrow \mathsf{USimS.UpdatePk}(\mathsf{pp}_{\mathsf{USimS}}, \mathsf{pk}_{t'}^{(\mathsf{u})}, \cdot)$ and $\mathsf{sk}^* \leftarrow \mathsf{USimS.UpdateSk}(\mathsf{pp}_{\mathsf{USimS}}, \mathsf{sk}_{t'}^{(\mathsf{u})}, \mathsf{up}^*)$ which represents the cumulative effect of all t' update queries made by $\mathcal{A}$, along with a final update whose randomness remains unknown to $\mathcal{A}$. Here, $\mathsf{pk}_{t'}^{(\mathsf{u})} = E_{A_{t'}} = [\boldsymbol{a}_{t'}]E_0 = [\boldsymbol{a} - \sum_{i=0}^{t'-1} \mathbf{r}_i]E_0 = [-\mathbf{r}'][\boldsymbol{a}]\mathsf{pk}_0^{(\mathsf{u})} = [-\mathbf{r}']E_A$ which implicitly sets $\mathsf{sk}_{t'}^{(\mathsf{u})} = a_{t'} = \boldsymbol{a} - \sum_{i=0}^{t'-1} \mathbf{r}_i = \boldsymbol{a} - \mathbf{r}'$ where $\mathbf{r}' = \sum_{i=0}^{t'-1} \mathbf{r}_i$.

To simulate $(\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{up}^*)$, the adversary $\mathcal{B}$ employs $\mathbf{z} = \boldsymbol{a} - \mathbf{r}^*$ and the ciphertext $\mathsf{ct}_{\boldsymbol{\rho}^*} \leftarrow \mathsf{SimS.Enc}(\mathsf{pp}_{\mathsf{USimS}}, \mathsf{pk}_0^{(\mathsf{u})} = \mathsf{pk}, \boldsymbol{\rho}^*)$ with $\mathsf{ct}_{\boldsymbol{\rho}^*} = (\mathsf{ct}^{(1)}_{\boldsymbol{\rho}^*}, \mathsf{ct}^{(2)}_{\boldsymbol{\rho}^*})$, $\mathsf{ct}^{(1)}_{\boldsymbol{\rho}^*} = [\mathbf{b}']E_0$, $\mathsf{ct}^{(2)}_{\boldsymbol{\rho}^*} = R_{[\mathbf{b}']E_A}(x([2\boldsymbol{\rho}^* + 1]P_{[\mathbf{b}']E_A}))$ and $\mathbf{b}, \mathbf{b}' \xleftarrow{\$} [-\mu, \mu]^n$ with $\mathbf{r}^* = \mathsf{KDF}(\boldsymbol{\rho}^*)$ for some $\boldsymbol{\rho}^* \in \mathbb{Z}_{2^{q-2}}$ received from its own challenger $\mathcal{C}$ as follows.

The adversary $\mathcal{B}$ computes $\mathbf{r}' = \sum_{i=0}^{t'-1} \mathbf{r}_i$ by extracting r_i for $i = 0, 1, \ldots, t' - 1$ from its maintained list rList and sets secret key sk^* at epoch $t' + 1$ as $\mathsf{sk}^* = \mathbf{z} - \mathbf{r}'$ and defines the public key at epoch $t' + 1$ as $\mathsf{pk}^* = [\mathsf{sk}^*]E_0$.

Note that

$$\mathsf{sk}^* = \mathbf{z} - \mathbf{r}' = \boldsymbol{a} - \mathbf{r}^* - \mathbf{r}' = \mathsf{sk}_{t'}^{(\mathsf{u})} - \mathbf{r}^* = \boldsymbol{a}_{t'} - \mathbf{r}^*$$

which has the same distribution as the real protocol by executing $\mathsf{USimS.UpdateSk}(\mathsf{pp}_{\mathsf{USimS}}, \mathsf{sk}_{t'}^{(\mathsf{u})}, \mathsf{up}^*)$. Moreover, pk^* is set to be $[\mathsf{sk}^*]E_0$ which is identically distributed as the real protocol by executing $\mathsf{USimS.UpdatePk}(\mathsf{pp}_{\mathsf{USimS}}, \mathsf{pk}_{t'}^{(\mathsf{u})}, \boldsymbol{\rho}^*)$. Thus, $\mathbf{z} = \boldsymbol{a} - \mathbf{r}^*$ sent by $\mathcal{C}$ to $\mathcal{B}$ is sufficient for $\mathcal{B}$ to simulates sk^* and pk^* without knowing explicitly ρ^* or $\mathbf{r}^* = \mathsf{KDF}(\boldsymbol{\rho}^*)$.

To compute up^*, the adversary $\mathcal{B}$ sets

$$\mathsf{up}^* = (\widetilde{\mathsf{ct}}_{\boldsymbol{\rho}^*}^{(1)} = [\mathbf{r}']\mathsf{ct}_{\boldsymbol{\rho}^*}^{(1)}, \widetilde{\mathsf{ct}}_{\boldsymbol{\rho}^*}^{(2)} = \mathsf{ct}_{\boldsymbol{\rho}^*}^{(2)})$$

by extracting

$$\mathsf{ct}_{\boldsymbol{\rho}^*}^{(1)} = [\mathbf{b}']E_0, \text{ and } \mathsf{ct}_{\boldsymbol{\rho}^*}^{(2)} = R_{[\mathbf{b}']E_A}(x([2\boldsymbol{\rho}^* + 1]P_{[\mathbf{b}']E_A}))$$

Finally, $(\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{up}^*)$ is forwarded to $\mathcal{A}$.

We now establish that $\widetilde{\mathsf{ct}}_{\boldsymbol{\rho}^*}$ is a valid encryption of $\boldsymbol{\rho}^*$ under $\mathsf{pk}_{t'}^{(\mathsf{u})}$. Rewriting the ciphertext,

$$\begin{aligned}
\widetilde{\mathsf{ct}}_{\boldsymbol{\rho}^*} &= (\widetilde{\mathsf{ct}}_{\boldsymbol{\rho}^*}^{(1)}, \widetilde{\mathsf{ct}}_{\boldsymbol{\rho}^*}^{(2)}) \\
&= ([\mathbf{r}']\mathsf{ct}_{\boldsymbol{\rho}^*}^{(1)}, \mathsf{ct}_{\boldsymbol{\rho}^*}^{(2)}) \\
&= ([\mathbf{b}' + \mathbf{r}']E_0, R_{[\mathbf{b}'+\boldsymbol{a}]E_0}(x([2\boldsymbol{\rho}^* + 1]P_{[\mathbf{b}'+\boldsymbol{a}]E_0}))) \\
&= ([\mathbf{b}' + \mathbf{r}']E_0, R_{[\mathbf{b}'+\mathbf{r}'+\boldsymbol{a}-\mathbf{r}']E_0}(x([2\boldsymbol{\rho}^* + 1]P_{[\mathbf{b}'+\mathbf{r}'+\boldsymbol{a}-\mathbf{r}']E_0}))) \\
&= ([\mathbf{b}' + \mathbf{r}']E_0, R_{[\mathbf{b}'+\mathbf{r}']\mathsf{pk}_{t'}^{(\mathsf{u})}}(x([2\boldsymbol{\rho}^* + 1]P_{[\mathbf{b}'+\mathbf{r}']\mathsf{pk}_{t'}^{(\mathsf{u})}})))
\end{aligned}$$

Defining $\tilde{\mathbf{b}}' = \mathbf{b}' + \mathbf{r}'$ and noting that $\mathbf{b}' \xleftarrow{\$} [-\mu, \mu]^n$, it follows that $\tilde{\mathbf{b}}'$ remains uniformly distributed in $[-\mu, \mu]^n$. This confirms that up^* has an identical distribution as that in the real protocol generated by calling $\mathsf{USimS.Enc}(\mathsf{pp}_{\mathsf{USimS}}, \mathsf{pk}_{t'}^{(\mathsf{u})}, \boldsymbol{\rho}^*)$ validating the correctness of the simulated ciphertext under the USimS encryption scheme. Thus, $\mathsf{ct}_{m_b^*} = (\mathsf{ct}_{m_b^*}^{(1)} = [\mathbf{b}]E_0, \mathsf{ct}_{m_b^*}^{(2)} = R_{[\mathbf{b}]E_A}(x([2m_b^* + 1]P_{[\mathbf{b}]E_A})))$ sent by $\mathcal{C}$ to $\mathcal{B}$ is sufficient for $\mathcal{B}$ to simulates up^* without explicit knowledge of ρ^* or $\mathbf{r}^* = \mathsf{KDF}(\boldsymbol{\rho}^*)$. This validates the correctness of the simulated updated ciphertext up^* under the encryption scheme USimS.

Guess Phase: Upon receiving $b' \in \{0, 1\}$ from the adversary $\mathcal{A}$, the adversary $\mathcal{B}$ submits b' to the challenger $\mathcal{C}$.

Observe that $\mathcal{B}$ perfectly simulates the $\mathsf{IND\text{-}CR\text{-}CPA}$ security game to $\mathcal{A}$ and succeeds whenever $\mathcal{A}$ does. Thus

$$\begin{aligned}
\mathsf{Adv}_{\mathsf{USimS},\mathcal{A}}^{\mathsf{IND\text{-}CR\text{-}CPA}}(\lambda) &= \left| \Pr[\mathsf{Exp}_{\mathsf{USimS},\mathcal{A}}^{\mathsf{IND\text{-}CR\text{-}CPA}}(\lambda) = 1] - \frac{1}{2} \right| \\
&= \left| \Pr[\mathsf{Exp}_{\mathsf{SimS},\mathcal{A}}^{f\text{-}\mathsf{CS+LR}}(\lambda) = 1] - \frac{1}{2} \right| \\
&= \mathsf{Adv}_{\mathsf{SimS},\mathcal{A}}^{f\text{-}\mathsf{CS+LR}}(\lambda)
\end{aligned}$$

Theorem 33 follows. □

Theorem 34. *If the isogeny based* UPKE *construction* USimS *described in Section 5.2 is* IND-CR-CPA *secure and* CSSIKoE *assumption holds then it provides* IND-CR-CCA *security defined in Definition 24.*

Proof Let $\mathcal{A}$ can win in $\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CCA}}_{\mathsf{USimS},\mathcal{B}}(\lambda)$ and CSSIKoE assumption holds then we can construct an adversary $\mathcal{B}$ who can win in $\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{USimS},\mathcal{B}}(\lambda)$ $\mathsf{Adv}^{\mathsf{IND\text{-}CR\text{-}CCA}}_{\mathsf{USimS},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{USimS},\mathcal{B}}(\lambda)$.

In the experiment $\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{USimS},\mathcal{B}}(\lambda)$, the challenger $\mathcal{C}$ generates the public parameter $\mathsf{pp}_{\mathsf{USimS}} = (p, q, \ell_1, \ldots, \ell_n, \mu, E_0, R_E, \mathsf{KDF})$ and a secret-public key pair $(\mathsf{sk}_0^{(\mathsf{u})} = \boldsymbol{a}_0,\ \mathsf{pk}_0^{(\mathsf{u})} = E_{A_0} = [\boldsymbol{a}_0]E_0)$ by running $\mathsf{USimS.KeyGen}(\mathsf{pp}_{\mathsf{USimS}})$. It provides $\mathsf{pp}_{\mathsf{USimS}}$, $\mathsf{pk}_0^{(\mathsf{u})}$ and access to $\mathcal{O}^{\mathsf{USimS}}_{\mathsf{up}}(\cdot)$ oracle to the adversary $\mathcal{B}$. The adversary $\mathcal{B}$ proceeds to simulate the experiment $\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{USimS},\mathcal{B}}(\lambda)$ in the following manner.

Setup: The adversary $\mathcal{B}$ initializes a list $\mathsf{rList} = \phi$ and epoch $i = 0$ and sends $\mathsf{pp}_{\mathsf{USimS}}$ and $\mathsf{pk}_0^{(\mathsf{u})}$ to the adversary $\mathcal{A}$.

Simulation of the oracle query: In the following, we will demonstrate that $\mathcal{B}$ simulates the following oracle perfectly for $\mathcal{A}$.

- $\mathcal{O}^{\mathsf{USimS}}_{\mathsf{up}}(\boldsymbol{\rho}_i)$: Upon receiving a query on the $\boldsymbol{\rho}_i \in \mathbb{Z}_{2^{q-2}}$, the adversary $\mathcal{B}$ call its it own $\mathcal{O}^{\mathsf{USimS}}_{\mathsf{up}}(\boldsymbol{\rho}_i)$ oracle.
- $\mathcal{O}^{\mathsf{USimS}}_{\mathsf{dec}}(\mathsf{ct}_m)$: Given a ciphertext $\mathsf{ct}_m = ([\mathbf{b}']E_0,\ R_{[\boldsymbol{a}_0][\mathbf{b}']}(x(P)))$ where $P \in [\boldsymbol{a}_0][\mathbf{b}']E_0$ is a point of order 2^q. By CSSIKoE assumption, there exists a polynomial time algorithm $\mathcal{A}'$ that on input $\mathsf{ct}_m = ([\mathbf{b}']E_0,\ R_{[\boldsymbol{a}_0][\mathbf{b}']}(x(P)))$ outputs the tuple $([\mathbf{b}'],\ [\mathbf{b}']E_0,\ R_{[\boldsymbol{a}_0][\mathbf{b}']}(x(P)))$. From the knowledge of the ideal classes $[\mathbf{b}']$ and $[\boldsymbol{a}_0]E_0$, the adversary $\mathcal{B}$ successfully decrypts ct_m.

Simulation of the challenge phase: After receiving $(m_0^*, m_1^*) \in \mathbb{Z}_{2^{q-2}} \times \mathbb{Z}_{2^{q-2}}$ from the adversary $\mathcal{A}$, the adversary $\mathcal{B}$ also forwards $(m_0^*, m_1^*) \in \mathbb{Z}_{2^{q-2}} \times \mathbb{Z}_{2^{q-2}}$ to its challenger $\mathcal{C}$. In response, the challenger $\mathcal{C}$ sends $\mathsf{ct}_{m_b^*} = \mathsf{USimS.Enc}(\mathsf{pp}, \mathsf{pk}_t^{(\mathsf{u})}, m_b^*)$ to $\mathcal{B}$ where t is the current epoch at which $\mathcal{A}$ sent its challenge messages. The adversary $\mathcal{B}$ also forwards $\mathsf{ct}_{m_b^*}$ the adversary $\mathcal{A}$.

Simulation of the post-challenge oracle query: In the following, we will demonstrate that $\mathcal{B}$ simulates the following oracle perfectly for $\mathcal{A}$.

- $\mathcal{O}^{\mathsf{USimS}}_{\mathsf{up}}(\boldsymbol{\rho}_i)$: Upon receiving a query on the $\boldsymbol{\rho}_i \in \mathbb{Z}_{2^{q-2}}$, the adversary $\mathcal{B}$ call its it own $\mathcal{O}^{\mathsf{USimS}}_{\mathsf{up}}(\boldsymbol{\rho}_i)$ oracle.
- $\mathcal{O}^{\mathsf{USimS}}_{\mathsf{dec}}(\mathsf{ct}_m)$: Given a ciphertext $\mathsf{ct}_m = ([\mathbf{b}']E_0,\ R_{[\boldsymbol{a}_0][\mathbf{b}']}(x(P)))$ where $P \in [\boldsymbol{a}_0][\mathbf{b}']E_0$ is a point of order 2^q. If $(\mathsf{ct}_m = \mathsf{ct}_{m_b^*} \wedge \mathsf{pk}_i^{(\mathsf{u})} = \mathsf{pk}_t^{(\mathsf{u})})$ aborts. By CSSIKoE assumption, there exists a polynomial time algorithm $\mathcal{A}'$ that on input $\mathsf{ct}_m = ([\mathbf{b}']E_0,\ R_{[\boldsymbol{a}_0][\mathbf{b}']}(x(P)))$ outputs $([\mathbf{b}'],\ [\mathbf{b}']E_0,\ R_{[\boldsymbol{a}_0][\mathbf{b}']}(x(P)))$. From the knowledge of the ideal classes $[\mathbf{b}']$ and $[\boldsymbol{a}_0]E_0$, the adversary $\mathcal{B}$ successfully decrypts ct_m.

Reveal Phase: Let $\mathcal{O}^{\mathsf{USimS}}_{\mathsf{up}}(\boldsymbol{\rho}_{t'})$ is the last query asked by $\mathcal{A}$. After receiving $(\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{up}^*)$ from $\mathcal{C}$ the adversary $\mathcal{B}$ forwards $(\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{up}^*)$ to $\mathcal{A}$ where pk^*, sk^* and up^* are public key, secret key and updated ciphertext respectively at epoch t'.

Guess Phase: Upon receiving $b' \in \{0, 1\}$ from the adversary $\mathcal{A}$, the adversary $\mathcal{B}$ submits b' to the challenger $\mathcal{C}$.

Thus $\mathcal{B}$ perfectly simulates $\mathcal{A}$'s challenger in the IND-CR-CCA security game and succeeds whenever $\mathcal{A}$ does. Hence, we get $\mathsf{Adv}^{\mathsf{IND\text{-}CR\text{-}CCA}}_{\mathsf{USimS},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{USimS},\mathcal{B}}(\lambda)$.

□

6 Our Protocols for Updatable KEM

6.1 Construction 1: UKEM_1

We present below an updatable key encapsulation mechanism $\mathsf{UKEM}_1 = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps}, \mathsf{UpdatePk}, \mathsf{UpdateSk})$ based on our scheme $\mathsf{UhPKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{UpdatePk}, \mathsf{UpdateSk})$ described in Section 5.1 associated with a message space $\mathcal{MS} = \{0,1\}^{\mathsf{mlen}(\lambda)}$ and a key space $\mathcal{K}\mathbf{ey} = \{0,1\}^{\mathsf{klen}(\lambda)}$.

$\mathsf{UKEM}_1.\mathsf{Setup}(\lambda) \to \mathsf{pp}_{\mathsf{UKEM}_1}$: On the input the security parameter λ, a trusted party proceeds as follows similar to our $\mathsf{UhPKE.Setup}(\lambda)$ with one additional key derivation function.

i. Selects a prime $p = 4\Pi_{i=1}^{n}\ell_i - 1$ where ℓ_i's are small distinct odd primes. Sets a base curve $E_0 : y^2 = x^3 + x \in \mathsf{Ell}_p(\mathcal{O})$ over $\mathbb{F}_p$ with $\mathcal{O} = \mathbb{Z}[\sqrt{-p}]$ and picks an integer μ such that $(2\mu+1)^n \geq \#\mathsf{Cl}(\mathcal{O})$.
ii. Picks two key derivation function $\mathsf{KDF}_1 : \{0,1\}^{\mathsf{mlen}(\lambda)} \to [-\mu,\mu]^n$ and $\mathsf{KDF}_2 : \{0,1\}^{\mathsf{mlen}(\lambda)} \to \{0,1\}^{\mathsf{klen}(\lambda)}$.
iii. Samples a family of keyed hash function $\mathcal{H} = \{H_k\}_{k\in\mathcal{K}}$ where $H_k : \mathbb{F}_p \to \{0,1\}^{\mathsf{mlen}(\lambda)}$ for each $k \in \mathcal{K}$ where $\mathcal{K}$ is key space. .
iv. Outputs the public parameter $\mathsf{pp}_{\mathsf{UKEM}_1} = (\mathsf{pp}_{\mathsf{UhPKE}} = (p, \ell_1, \ldots, \ell_n, \mu, E_0, \mathcal{H} = \{H_k\}_{k\in\mathcal{K}}, \mathsf{KDF}_1), \mathsf{KDF}_2)$.

$\mathsf{UKEM}_1.\mathsf{KeyGen}(\mathsf{pp}_{\mathsf{UKEM}_1}) \to (\mathsf{sk}_0^{(\mathsf{u})} = \boldsymbol{a}_0, \mathsf{pk}_0^{(\mathsf{u})} = E_{A_0})$: On input the public parameter $\mathsf{pp}_{\mathsf{UKEM}_1} = (\mathsf{pp}_{\mathsf{UhPKE}} = (p, \ell_1, \ldots, \ell_n, \mu, E_0, \mathcal{H} = \{H_k\}_{k\in\mathcal{K}}, \mathsf{KDF}_1), \mathsf{KDF}_2)$, a user u generates a secret-public key pair $(\mathsf{sk}_0^{(\mathsf{u})}, \mathsf{pk}_0^{(\mathsf{u})})$ by executing the following steps exactly same way as in the $\mathsf{UhPKE.KeyGen}(\mathsf{pp}_{\mathsf{UhPKE}})$.

i. Samples $\boldsymbol{a}_0 = (a_1^0, \ldots, a_n^0) \xleftarrow{\$} [-\mu,\mu]^n$ and define $[\boldsymbol{a}_0] = [\mathfrak{l}_1^{a_1^0} \cdots \mathfrak{l}_n^{a_n^0}] \in \mathsf{Cl}(\mathcal{O})$ where $\mathfrak{l}_i = \langle \ell_i, \pi - 1\rangle$. Computes $E_{A_0} = [\boldsymbol{a}_0]E_0$.
ii. Sets the public key $\mathsf{pk}_0^{(\mathsf{u})} = E_{A_0}$ and the secret key $\mathsf{sk}_0^{(\mathsf{u})} = \boldsymbol{a}_0$.
iii. Publishes $\mathsf{pk}_0^{(\mathsf{u})} = E_{A_0}$ and keeps $\mathsf{sk}_0^{(\mathsf{u})} = \boldsymbol{a}_0$ secret to itself.

$\mathsf{UKEM}_1.\mathsf{Encaps}(\mathsf{pp}_{\mathsf{UKEM}_1}, \mathsf{pk}_i^{(\mathsf{u})} = E_{A_i}) \to (\mathsf{hct}, \mathsf{ek})$: Taking input the public parameter $\mathsf{pp}_{\mathsf{UKEM}_1} = (\mathsf{pp}_{\mathsf{UhPKE}} = (p, \ell_1, \ldots, \ell_n, \mu, E_0, \mathcal{H} = \{H_k\}_{k\in\mathcal{K}}, \mathsf{KDF}_1), \mathsf{KDF}_2)$, public key $\mathsf{pk}_i^{(\mathsf{u})} = E_{A_i}$ a user proceeds to compute a header ciphertext hct encrypting a random message $m \in \mathcal{MS}$ using $\mathsf{UhPKE.Enc}(\mathsf{pp}_{\mathsf{UhPKE}}, \mathsf{pk}_i^{(\mathsf{u})} = E_{A_i}, m)$ and generates an encapsulation key ek. More precisely, the user performs the following steps :

i. Randomly samples a message $m \xleftarrow{\$} \mathcal{MS}$, computes $\mathsf{KDF}_1(m) = \mathbf{b} = (b_1, \ldots, b_n) \in [-\mu,\mu]^n$ and $[\mathbf{b}] = [\mathfrak{l}_1^{b_1} \cdots \mathfrak{l}_n^{b_n}] \in \mathsf{Cl}(\mathcal{O})$.
ii. Computes $\mathsf{hct}^{(1)} = [\mathbf{b}]E_0$, $\mathsf{hct}^{(2)} = m \oplus H_k(M_C([\mathbf{b}]E_{A_i}))$ and $\mathsf{ek} = \mathsf{KDF}_2(m)$ where $M_C([\mathbf{b}]E_{A_i})$ is the Montgomery coefficient of the elliptic curve $[\mathbf{b}]E_{A_i}$.
iii. Publishes the header ciphertext $\mathsf{hct} = (\mathsf{hct}^{(1)}, \mathsf{hct}^{(2)})$ and keeps encapsulation key ek secret to itself.

$\mathsf{UKEM}_1.\mathsf{Decaps}(\mathsf{pp}_{\mathsf{UKEM}_1}, \mathsf{sk}_i^{(\mathsf{u})} = \boldsymbol{a}_i, \mathsf{hct} = (\mathsf{hct}^{(1)}, \mathsf{hct}^{(2)})) \to \mathsf{ek}$: A decrypter runs this deterministic algorithm on input the public parameter $\mathsf{pp}_{\mathsf{UKEM}_1} = (\mathsf{pp}_{\mathsf{UhPKE}} = (p, \ell_1, \ldots, \ell_n, \mu, E_0, \mathcal{H} = \{H_k\}_{k\in\mathcal{K}}, \mathsf{KDF}_1), \mathsf{KDF}_2)$ and header ciphertext hct using its own secret key $\mathsf{sk}_i^{(\mathsf{u})} = \boldsymbol{a}_i$ to recover ek by executing the following:

i. Computes $\alpha = H_k(M_C([\boldsymbol{a}_i]\mathsf{hct}^{(1)}))$ where $M_C([\boldsymbol{a}_i]\mathsf{hct}^{(1)})$ is the Montgomery coefficient of the elliptic curve $[\boldsymbol{a}_i]\mathsf{hct}^{(1)}$.

ii. Returns the encapsulation key $\mathsf{ek} = \mathsf{KDF}_2(\mathsf{hct}^{(2)} \oplus \alpha)$.

$\mathsf{UhKEM}.\mathsf{UpdatePk}(\mathsf{pp}_{\mathsf{UKEM}_1}, \mathsf{pk}_i^{(\mathsf{u})}; \boldsymbol{\rho}_i) \to (\mathsf{pk}_{i+1}^{(\mathsf{u})}, \mathsf{up}_{i+1})$: Silimar to $\mathsf{UhPKE}.\mathsf{UpdatePk}(\mathsf{pp}_{\mathsf{UhPKE}}, \mathsf{pk}_i^{(\mathsf{u})}; \boldsymbol{\rho}_i)$ on input of the public parameter $\mathsf{pp}_{\mathsf{UKEM}_1} = (\mathsf{pp}_{\mathsf{UhPKE}} = (p, \ell_1, \ldots, \ell_n, \mu, E_0, \mathcal{H} = \{H_k\}_{k\in\mathcal{K}}, \mathsf{KDF}_1), \mathsf{KDF}_2)$, a public key $\mathsf{pk}_i^{(\mathsf{u})} = E_{A_i}$ and a randomness $\boldsymbol{\rho}_i$, a user u can run this algorithm and proceeds to compute an updated ciphertext up_{i+1} and a new public key $\mathsf{pk}_{i+1}^{(\mathsf{u})}$ in the following way:

i. Define $\mathbf{r}_i = (r_1, \ldots, r_n) = \mathsf{KDF}_1(\boldsymbol{\rho}_i)$ and $[\mathbf{r}_i] = [\mathfrak{l}_1^{r_1} \cdots \mathfrak{l}_n^{r_n}] \in \mathsf{Cl}(\mathcal{O})$ where $\mathfrak{l}_i = \langle \ell_i, \pi - 1 \rangle$.

ii. Computes $\mathsf{pk}_{i+1}^{(\mathsf{u})} = E_{A_{i+1}} = [-\mathbf{r}_i]E_{A_i}$ and generates updated ciphertext $\mathsf{up}_{i+1} = (\mathsf{ct}_{\boldsymbol{\rho}_i}^{(1)} = [\mathbf{b}]E_0, \mathsf{ct}_{\boldsymbol{\rho}_i}^{(2)} = \boldsymbol{\rho}_i \oplus H_k([\mathbf{b}]\mathsf{pk}_i^{(\mathsf{u})}))$ by running $\mathsf{UhPKE}.\mathsf{Enc}(\mathsf{pp}_{\mathsf{UhPKE}}, \mathsf{pk}_i^{(\mathsf{u})}, \boldsymbol{\rho}_i)$ where $\mathbf{b} \xleftarrow{\$} [-\mu, \mu]^n$.

iii. Returns updated public key $\mathsf{pk}_{i+1}^{(\mathsf{u})} = E_{A_{i+1}}$ and updated ciphertext $\mathsf{up}_{i+1} = (\mathsf{ct}_{\boldsymbol{\rho}_i}^{(1)}, \mathsf{ct}_{\boldsymbol{\rho}_i}^{(2)})$.

$\mathsf{UKEM}_1.\mathsf{UpdateSk}(\mathsf{pp}_{\mathsf{UKEM}_1}, \mathsf{sk}_i^{(\mathsf{u})}, \mathsf{up}_{i+1}) \to \mathsf{sk}_{i+1}^{(\mathsf{u})}$: On input the public parameter $\mathsf{pp}_{\mathsf{UKEM}_1} = (\mathsf{pp}_{\mathsf{hPKE}} = (p, \ell_1, \ldots, \ell_n, \mu, E_0, \mathcal{H} = \{H_k\}_{k\in\mathcal{K}}, \mathsf{KDF}_1), \mathsf{KDF}_2)$ and an updated ciphertext $\mathsf{up}_{i+1} = (\mathsf{ct}_{\boldsymbol{\rho}_i}^{(1)}, \mathsf{ct}_{\boldsymbol{\rho}_i}^{(2)})$ an user with $\mathsf{sk}_i^{(\mathsf{u})} = \boldsymbol{a}_i$ runs this algorithm and generates updated secret key $\mathsf{sk}_{i+1}^{(\mathsf{u})}$ as in $\mathsf{UhPKE}.\mathsf{UpdateSk}(\mathsf{pp}_{\mathsf{UhPKE}}, \mathsf{sk}_i^{(\mathsf{u})}, \mathsf{up}_{i+1})$.

i. Decrypts the updated ciphertext $\boldsymbol{\rho}_i = \mathsf{ct}_{\boldsymbol{\rho}_i}^{(2)} \oplus H_k(M_C([\mathsf{sk}_i^{(\mathsf{u})}]\mathsf{ct}_{\boldsymbol{\rho}_i}^{(1)}))$ by calling $\mathsf{UhPKE}.\mathsf{Dec}(\mathsf{pp}_{\mathsf{UhPKE}}, \mathsf{sk}_i^{(\mathsf{u})}, \mathsf{up}_{i+1})$.

ii. Updates secret key $\mathsf{sk}_{i+1}^{(\mathsf{u})} = \mathsf{sk}_i^{(\mathsf{u})} - \mathsf{KDF}_1(\boldsymbol{\rho}_i)$ and returns $\mathsf{sk}_{i+1}^{(\mathsf{u})}$.

Correctness. The correctness of UKEM_1 follows from the correctness of UhPKE described in Section 5.1.

Theorem 35. *If* UhPKE *described in Section 5.1 is* $\mathsf{IND\text{-}CR\text{-}CPA}$ *secure (see Definition 23) and* KDF_2 *is a secure key derivation function as per Definition 16 then* UKEM_1 *provides* $\mathsf{IND\text{-}CR\text{-}CPA}$ *security as per Definition 27.*

Proof We will demonstrate that if the adversary $\mathcal{A}$ wins in $\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{UKEM}_1,\mathcal{A}}(\lambda)$ then we can construct an adversary $\mathcal{B}$ (say) using the adversary $\mathcal{A}$ that can win in $\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{UhPKE},\mathcal{B}}(\lambda)$ and thereby $\mathsf{Adv}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{UKEM}_1,\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{UhPKE},\mathcal{B}}(\lambda)$.

In the experiment $\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{UhPKE},\mathcal{B}}(\lambda)$, the UhPKE challenger $\mathcal{C}$ generates the public parameter $\mathsf{pp}_{\mathsf{UhPKE}} = (p, \ell_1, \ldots, \ell_n, \mu, E_0, \mathcal{H} = \{H_k\}_{k\in\mathcal{K}}, \mathsf{KDF}_1)$ by running $\mathsf{UhPKE.Setup}(\lambda)$ and a secret-public key pair $(\mathsf{sk}_0^{(\mathsf{u})} = \boldsymbol{a}_0, \mathsf{pk}_0^{(\mathsf{u})} = E_{A_0} = [\boldsymbol{a}_0]E_0)$ is generated by running

$\mathsf{UhPKE.KeyGen}(\mathsf{pp}_{\mathsf{UhPKE}})$. It provides $\mathsf{pp}_{\mathsf{UhPKE}}$, $\mathsf{pk}_0^{(\mathsf{u})}$ and gives $\mathcal{O}_{\mathsf{up}}^{\mathsf{UhPKE}}(\cdot)$ oracle access to the UhPKE adversary $\mathcal{B}$. The adversary $\mathcal{B}$ proceeds to simulate the experiment $\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{UKEM}_1,\mathcal{A}}(\lambda)$ to the UKEM_1 adversary $\mathcal{A}$ in the following manner.

Setup: The adversary $\mathcal{B}$ sets epoch $i = 0$ and a key derivation function $\mathsf{KDF}_2 : \{0,1\}^{\mathsf{mlen}(\lambda)} \to \{0,1\}^{\mathsf{klen}(\lambda)}$ and sends $\mathsf{pp}_{\mathsf{UKEM}_1} = (\mathsf{pp}_{\mathsf{UhPKE}}, \mathsf{KDF}_2)$ and $\mathsf{pk}_0^{(\mathsf{u})}$ to the UKEM_1 adversary $\mathcal{A}$.

Simulation of the pre-challenge oracle query: In the following, we will demonstrate that $\mathcal{B}$ simulates the following oracle perfectly for $\mathcal{A}$.

– $\mathcal{O}_{\mathsf{up}}^{\mathsf{UKEM}_1}(\boldsymbol{\rho}_i)$: Upon receiving a query on the $\boldsymbol{\rho}_i \in \{0,1\}^{\mathsf{mlen}(\lambda)}$, the adversary $\mathcal{B}$ calls its own oracle $\mathcal{O}_{\mathsf{up}}^{\mathsf{UhPKE}}(\boldsymbol{\rho}_i)$ and increments the counter i to $i+1$.

Simulation of the challenge phase: The UKEM_1 challenger $\mathcal{B}$ does the following to simulate challenge phase to UKEM_1 adversary $\mathcal{A}$.

i. $\mathcal{B}$ samples $(m_0^*, m_1^*) \xleftarrow{\$} \{0,1\}^{\mathsf{mlen}(\lambda)} \times \{0,1\}^{\mathsf{mlen}(\lambda)}$ to the UhPKE challenger $\mathcal{C}$.

ii. In response, the challenger $\mathcal{C}$ uniformly samples $b \xleftarrow{\$} \{0,1\}$ and computes $\mathsf{ct}_{m_b^*} = (\mathsf{ct}^{(1)}_{m_b^*} = [\mathbf{b}]E_0, \mathsf{ct}^{(2)}_{m_b^*} = m_b^* \oplus H_k(M_C([\mathbf{b}]\mathsf{pk}_t^{(\mathsf{u})}))) \leftarrow \mathsf{UhPKE.Enc}(\mathsf{pp}_{\mathsf{UhPKE}}, \mathsf{pk}_t^{(\mathsf{u})}, m_b^*)$ for some $\mathbf{b} \xleftarrow{\$} [-\mu, \mu]^n$ where t is the current epoch. The UhPKE challenger $\mathcal{C}$ sends $\mathsf{ct}_{m_b^*} = (\mathsf{ct}^{(1)}_{m_b^*}, \mathsf{ct}^{(2)}_{m_b^*})$ to $\mathcal{B}$.

iii. Then the adversary $\mathcal{B}$ sends $(\mathsf{hct}^* = \mathsf{ct}_{m_b^*}, \mathsf{ek}^* = \mathsf{KDF}_2(m_1^*))$ to $\mathcal{A}$

Simulation of the post-challenge oracle query: The UhPKE adversary $\mathcal{B}$ simulates the oracle exactly the same as in the pre-challenge oracle query to $\mathcal{O}_{\mathsf{up}}^{\mathsf{UKEM}_1}(\cdot)$.

Reveal Phase: Let $\mathcal{O}_{\mathsf{up}}^{\mathsf{USimS}}(\boldsymbol{\rho}_{t'})$ is the last query asked by $\mathcal{A}$. After receiving $(\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{up}^*)$ from $\mathcal{C}$ the adversary $\mathcal{B}$ forwards $(\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{up}^*)$ to $\mathcal{A}$ where pk^*, sk^* and up^* are public key, secret key and updated ciphertext respectively at epoch t'.

Guess Phase: Upon receiving $b' \in \{0,1\}$ from the adversary $\mathcal{A}$, the adversary $\mathcal{B}$ submits b' to the challenger $\mathcal{C}$.

Note that,

- If $b = 0$, the UhPKE challenger $\mathcal{C}$ transmits $\mathsf{ct}_{m_0^*}$ to the adversary $\mathcal{B}$. Subsequently, the UKEM_1 challenger $\mathcal{B}$ forwards the pair $(\mathsf{hct}^* = \mathsf{ct}_{m_0^*}, \mathsf{ek}^* = \mathsf{KDF}_2(m_1^*))$ to the adversary $\mathcal{A}$. Here, the ciphertext hct^* consists of two components:

$$\mathsf{ct}^{(1)}_{m_0^*} = [\mathbf{b}]E_0,\ \mathsf{ct}^{(2)}_{m_0^*} = m_0^* \oplus H_k(M_C([\mathbf{b}]\mathsf{pk}_t^{(\mathsf{u})}))$$

where m_0^* and m_1^* are sampled uniformly from $\{0,1\}^{\mathsf{mlen}(\lambda)}$ and $\mathbf{b}$ is sampled uniformly from $[-\mu, \mu]^n$.

Since $\mathbf{b}$ is indistinguishable from $\mathsf{KDF}_1(m_0^*)$ due to the security properties of KDF_1, we conclude that

$$\mathsf{hct}^* = \left(\mathsf{ct}^{(1)}_{m_0^*} = [\mathbf{b}]E_0, \mathsf{ct}^{(2)}_{m_0^*} = m_0^* \oplus H_k(M_C([\mathbf{b}]\mathsf{pk}_t^{(\mathsf{u})}))\right)$$

is identically distributed to

$$\left(\widetilde{\mathsf{ct}}^{(1)}_{m_0^*} = [\mathsf{KDF}_1(m_0^*)]E_0, \widetilde{\mathsf{ct}}^{(2)}_{m_0^*} = m_0^* \oplus H_k(M_C([\mathsf{KDF}_1(m_0^*)]\mathsf{pk}_t^{(\mathsf{u})}))\right)$$

Thus, $(\mathsf{hct}^*, \cdot)$ follows the same distribution as the output of $\mathsf{UhKEM.Encaps}(\mathsf{pp}_{\mathsf{UKEM}_1}, \mathsf{pk}_t^{(\mathsf{u})})$.

Furthermore, since m_1^* is uniformly sampled from $\{0,1\}^{\mathsf{mlen}(\lambda)}$, the security of KDF_2 ensures that $\mathsf{ek}^* = \mathsf{KDF}_2(m_1^*)$ is uniformly distributed over the key space $\mathcal{KS} = \{0,1\}^{\mathsf{klen}(\lambda)}$.
Consequently, the tuple $(\mathsf{hct}^*, \mathsf{ek}^*)$ follows the same distribution in the case where $b = 0$ during the challenge phase of the experiment $\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{UKEM}_1,\mathcal{A}}(\lambda)$ as described in Fig. 6.

- If $b = 1$, the UhPKE challenger $\mathcal{C}$ transmits $\mathsf{ct}_{m_1^*}$ to the adversary $\mathcal{B}$. Subsequently, the UKEM_1 challenger $\mathcal{B}$ forwards the pair $(\mathsf{hct}^* = \mathsf{ct}_{m_1^*}, \mathsf{ek}^* = \mathsf{KDF}_2(m_1^*))$ to the adversary $\mathcal{A}$.
Here, the ciphertext hct^* consists of two components

$$\mathsf{ct}^{(1)}_{m_1^*} = [\mathbf{b}]E_0,\ \mathsf{ct}^{(2)}_{m_1^*} = m_1^* \oplus H_k(M_C([\mathbf{b}]\mathsf{pk}_t^{(\mathsf{u})}))$$

where m_1^* is sampled uniformly at random from $\{0,1\}^{\mathsf{mlen}(\lambda)}$ and $\mathbf{b}$ is sampled uniformly from $[-\mu,\mu]^n$.
Since $\mathbf{b}$ is indistinguishable from $\mathsf{KDF}_1(m_1^*)$ due to the security properties of KDF_1, we conclude that

$$\mathsf{hct}^* = \left(\mathsf{ct}^{(1)}_{m_1^*} = [\mathbf{b}]E_0, \mathsf{ct}^{(2)}_{m_1^*} = m_1^* \oplus H_k(M_C([\mathbf{b}]\mathsf{pk}_t^{(\mathsf{u})}))\right)$$

is identically distributed to

$$\left(\widetilde{\mathsf{ct}}^{(1)}_{m_1^*} = [\mathsf{KDF}_1(m_1^*)]E_0, \widetilde{\mathsf{ct}}^{(2)}_{m_1^*} = m_1^* \oplus H_k(M_C([\mathsf{KDF}_1(m_1^*)]\mathsf{pk}_t^{(\mathsf{u})}))\right).$$

Thus, $(\mathsf{hct}^* = \mathsf{ct}_{m_1^*}, \mathsf{ek}^* = \mathsf{KDF}_2(m_1^*)$ follows the same distribution as the output of $\mathsf{UhKEM.Encaps}(\mathsf{pp}_{\mathsf{UKEM}_1}, \mathsf{pk}_t^{(\mathsf{u})})$.
Consequently, the tuple $(\mathsf{hct}^* = \mathsf{ct}_{m_1^*}, \mathsf{ek}^* = \mathsf{KDF}_2(m_1^*))$ follows the same distribution in the case where $b = 1$ during the challenge phase of the experiment $\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{UKEM}_1,\mathcal{A}}(\lambda)$ as described in Fig. 6.

Therefore, $\mathcal{B}$ succeeds in $\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{UhPKE},\mathcal{B}}(\lambda)$ if $\mathcal{A}$ succeeds in $\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{UKEM}_1,\mathcal{A}}(\lambda)$. This implies

$$\Pr[\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{UKEM}_1,\mathcal{A}}(\lambda) = 1] = \Pr[\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{UhPKE},\mathcal{B}}(\lambda) = 1]$$

Therefore,

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{UKEM}_1,\mathcal{A}}(\lambda) &= \left|\Pr[\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{UKEM}_1,\mathcal{A}}(\lambda) = 1] - \frac{1}{2}\right| \\
&= \left|\Pr[\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{UhPKE},\mathcal{B}}(\lambda) = 1] - \frac{1}{2}\right| \\
&= \mathsf{Adv}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{UhPKE},\mathcal{B}}(\lambda)
\end{aligned}$$

□

6.2 Construction 2: UKEM_2

Our second proposed updatable key encapsulation mechanism scheme is $\mathsf{UKEM}_2 = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps}, \mathsf{UpdatePk}, \mathsf{UpdateSk})$ associated with a message space $\mathcal{MS} = \mathbb{Z}_{2q-2}$ and a key space $\mathcal{K}\mathbf{ey} = \{0,1\}^{\mathsf{klen}(\lambda)}$ is based on $\mathsf{USimS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{UpdatePk}, \mathsf{UpdateSk})$ satisfying the following requirements:

$\mathsf{UKEM}_2.\mathsf{Setup}(\lambda) \to \mathsf{pp}_{\mathsf{UKEM}_2}$: A trusted party runs this algorithm on the input a security parameter λ and proceeds as follows:

i. Chooses a prime $p = 2^q \Pi_{i=1}^n \ell_i - 1$ such that $\lambda + 2 \leq q \leq \frac{1}{2}\log p$ and ℓ_i's are small distinct odd primes.
ii. Sets a base curve $E_0 : y^2 = x^3 + x \in \mathsf{Ell}_p(\mathcal{O})$ over $\mathbb{F}_p$ with $\mathcal{O} = \mathbb{Z}[\sqrt{-p}]$ and picks an integer μ such that $(2\mu+1)^n \geq \#\mathsf{Cl}(\mathcal{O})$.
iii. Consider two key derivation function $\mathsf{KDF}_1 : \mathbb{Z}_{2^{q-2}} \to [-\mu, \mu]^n$ and $\mathsf{KDF}_2 : \mathbb{Z}_{2^{q-2}} \to \{0,1\}^{\mathsf{klen}(\lambda)}$.
iv. Consider the function $R_E : \mathbb{F}_p \to \mathbb{F}_p$ defined as $R_E(x) = \mathsf{int}(\mathsf{bin}(x) \oplus \mathsf{bin}(M_C(E)))$ where $\mathsf{bin}(\cdot)$ and $\mathsf{int}(\cdot)$ represent the operations that convert an element of $\mathbb{F}_p$ into its binary representation and back, respectively.
v. Outputs the public parameter $\mathsf{pp}_{\mathsf{UKEM}_2} = (\mathsf{pp}_{\mathsf{USimS}} = (p, q, \ell_1, \ldots, \ell_n, \mu, E_0, R_E, \mathsf{KDF}_1), \mathsf{KDF}_2)$.

$\mathsf{UKEM}_2.\mathsf{KeyGen}(\mathsf{pp}_{\mathsf{UKEM}_2}) \to (\mathsf{sk}_0^{(\mathsf{u})} = \boldsymbol{a}_0, \mathsf{pk}_0^{(\mathsf{u})} = E_{A_0})$: On input the public parameter $\mathsf{pp}_{\mathsf{UKEM}_2} = (\mathsf{pp}_{\mathsf{USimS}} = (p, q, \ell_1, \ldots, \ell_n, \mu, E_0, R_E, \mathsf{KDF}_1), \mathsf{KDF}_2)$, a user generates a secret-public key pair $(\mathsf{sk}_0^{(\mathsf{u})}, \mathsf{pk}_0^{(\mathsf{u})})$ in the following manner:

i. Samples $\boldsymbol{a}_0 = (a_1^0, \ldots, a_n^0) \xleftarrow{\$} [-\mu, \mu]^n$ and define $[\boldsymbol{a}_0] = [\mathfrak{l}_1^{a_1^0} \cdots \mathfrak{l}_n^{a_n^0}] \in \mathsf{Cl}(\mathcal{O})$ where $\mathfrak{l}_i = \langle \ell_i, \pi - 1 \rangle$.
ii. Computes $E_{A_0} = [\boldsymbol{a}_0]E_0$.
iii. Returns public key $\mathsf{pk}_0^{(\mathsf{u})} = E_{A_0}$ and secret key $\mathsf{sk}_0^{(\mathsf{u})} = \boldsymbol{a}_0$.

$\mathsf{UKEM}_2.\mathsf{Encaps}(\mathsf{pp}_{\mathsf{UKEM}_2}, \mathsf{pk}_i^{(\mathsf{u})}) \to \mathsf{hct}$: Taking input the public parameter $\mathsf{pp}_{\mathsf{UKEM}_2} = (\mathsf{pp}_{\mathsf{USimS}} = (p, q, \ell_1, \ldots, \ell_n, \mu, E_0, R_E, \mathsf{KDF}_1), \mathsf{KDF}_2)$ and public key $\mathsf{pk}_i^{(\mathsf{u})} = E_{A_i}$, an encrypter proceeds to compute ciphertext as follows:

i. Uniformly samples $m \xleftarrow{\$} \mathbb{Z}_{2^{q-2}}$ and embeds $m \in \mathbb{Z}_{2^q}^\times$ via $m \to 2m+1$. Randomly samples $\mathbf{b} = (b_1, \ldots, b_n) \xleftarrow{\$} [-\mu, \mu]^n$ and define $[\mathbf{b}] = [\mathfrak{l}_1^{b_1} \cdots \mathfrak{l}_n^{b_n}] \in \mathsf{Cl}(\mathcal{O})$.
ii. Computes $E_B = [\mathbf{b}]E_0$, $E_{AB} = [\mathbf{b}]E_{A_i}$, $P_{AB} = [2m+1]P_{E_{AB}}$ and $\mathsf{ek} = \mathsf{KDF}_2(m)$. Here, $P_{E_{AB}}$ is a point on E_{AB} of order 2^q, which is determined using Algorithm 3.
iii. Returns $(\mathsf{hct} = (E_B, R_{E_{AB}}(x(P_{AB}))), \mathsf{ek})$.

$\mathsf{UKEM}_2.\mathsf{Decaps}(\mathsf{pp}_{\mathsf{UKEM}_2}, \mathsf{sk}_i^{(\mathsf{u})} = \boldsymbol{a}_i, \mathsf{hct} = (\mathsf{hct}^{(1)}, \mathsf{hct}^{(2)})) \to m/\perp$: A decrypter runs this deterministic algorithm on input the public parameter $\mathsf{pp}_{\mathsf{UKEM}_2} = (\mathsf{pp}_{\mathsf{USimS}} = (p, q, \ell_1, \ldots, \ell_n, \mu, E_0, R_E, \mathsf{KDF}_1), \mathsf{KDF}_2)$, ciphertext $\mathsf{hct} = (E_B, x')$ and its secret key $\mathsf{sk}_i^{(\mathsf{u})} = \boldsymbol{a}_i$ to recover the plaintext m in the following manner:

i. Verifies that E_B is a supersingular curve,
ii. Computes $E_{BA} = [\boldsymbol{a}_i]E_B$ and $P_{E_{BA}}$ a point on E_{BA} of order 2^q by using Algorithm 3.
iii. Computes $R_{E_{BA}}(x')$ and if $R_{E_{BA}}(x')$ is not the x-coordinate of a 2^q-torsion point on the curve E_{BA} then aborts.
iv. Solves the discrete logarithm instance between $P_{BA} = (R_{E_{BA}}(x'), \cdot)$ and $P_{E_{BA}}$ using the Pohlig-Hellman algorithm given in Algorithm 2. Let $m' \in \mathbb{Z}_{2^q}^\times$ be the solution of this computation. If $2^{q-1} < m'$ then it changes m' to $2^q - m'$. It returns $\mathsf{ek} = \mathsf{KDF}_2(\frac{(m'-1)}{2})$.

$\mathsf{UKEM}_2.\mathsf{UpdatePk}(\mathsf{pp}_{\mathsf{UKEM}_2}, \mathsf{pk}_i^{(\mathsf{u})}, \boldsymbol{\rho}_i) \to (\mathsf{pk}_{i+1}^{(\mathsf{u})}, \mathsf{up}_{i+1})$: Given the public parameter $\mathsf{pp}_{\mathsf{UKEM}_2} = (\mathsf{pp}_{\mathsf{USimS}} = (p, q, \ell_1, \ldots, \ell_n, \mu, E_0, R_E, \mathsf{KDF}_1), \mathsf{KDF}_2)$, a public key $\mathsf{pk}_i^{(\mathsf{u})} =$

E_{A_i} and a random $\boldsymbol{\rho}_i \in \mathbb{Z}_{2^{q-2}}$, any user can run this algorithm to compute an updated ciphertext up_{i+1} and a new public key $\mathsf{pk}_{i+1}^{(\mathsf{u})}$ in the following way:

i. Define $\mathbf{r}_i = (r_1, \ldots, r_n) = \mathsf{KDF}_1(\boldsymbol{\rho}_i)$ and $[\mathbf{r}_i] = [\mathfrak{l}_1^{r_1} \cdots \mathfrak{l}_n^{r_n}] \in \mathsf{Cl}(\mathcal{O})$ where $\mathfrak{l}_i = \langle \ell_i, \pi - 1 \rangle$.

ii. Computes $E_{A_{i+1}} = [-\mathbf{r}_i]E_{A_i}$ and updated ciphertext $\mathsf{up}_{i+1} \leftarrow \mathsf{USimS.Enc}(\mathsf{pp}_{\mathsf{UKEM}_2}, \mathsf{pk}_i^{(\mathsf{u})}, \boldsymbol{\rho}_i)$.

iii. Returns updated public key $\mathsf{pk}_{i+1}^{(\mathsf{u})} = E_{A_{i+1}}$ and ciphertext up_{i+1}.

$\mathsf{UKEM}_2.\mathsf{UpdateSk}(\mathsf{pp}_{\mathsf{UKEM}_2}, \mathsf{sk}_i^{(\mathsf{u})}, \mathsf{up}_{i+1}) \rightarrow \mathsf{sk}_{i+1}^{(\mathsf{u})}$: Given the public parameter $\mathsf{pp}_{\mathsf{UKEM}_2} = (\mathsf{pp}_{\mathsf{USimS}} = (p, q, \ell_1, \ldots, \ell_n, \mu, E_0, R_E, \mathsf{KDF}_1), \mathsf{KDF}_2)$ and an updated ciphertext up_{i+1} an user with $\mathsf{sk}_i^{(\mathsf{u})}$ computes $\boldsymbol{\rho}_i = \mathsf{UKEM}_2.\mathsf{Dec}(\mathsf{pp}_{\mathsf{UKEM}_2}, \mathsf{sk}_i^{(\mathsf{u})}, \mathsf{up}_{i+1})$ and $\mathsf{sk}_{i+1}^{(\mathsf{u})} = \mathsf{sk}_i^{(\mathsf{u})} - \mathsf{KDF}_1(\boldsymbol{\rho}_i)$ and returns updated secret key $\mathsf{sk}_{i+1}^{(\mathsf{u})}$.

Correctness. The correctness of UKEM_2 follows from the correctness of USimS described in Section 5.2.

Theorem 36. *If* USimS *described in Section 5.2 is* $\mathsf{IND\text{-}CR\text{-}CCA}$ *(see Definition 24) and* KDF_2 *is a secure key derivation function as per Definition 16 then* UKEM_2 *also provides* $\mathsf{IND\text{-}CR\text{-}CCA}$ *security as per Definition 28.*

Proof We will demonstrate that if the adversary $\mathcal{A}$ wins in $\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CCA}}_{\mathsf{UKEM}_2, \mathcal{A}}(\lambda)$ then we can construct an adversary $\mathcal{B}$ using the adversary $\mathcal{A}$ that can win in $\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CCA}}_{\mathsf{USimS}, \mathcal{B}}(\lambda)$ and thereby $\mathsf{Adv}^{\mathsf{IND\text{-}CR\text{-}CCA}}_{\mathsf{UKEM}_2, \mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathsf{IND\text{-}CR\text{-}CCA}}_{\mathsf{USimS}, \mathcal{B}}(\lambda)$.

In the experiment $\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{USimS}, \mathcal{B}}(\lambda)$, the USimS challenger $\mathcal{C}$ generates the public parameter $\mathsf{pp}_{\mathsf{USimS}} = (p, q, \ell_1, \ldots, \ell_n, \mu, E_0, R_E, \mathsf{KDF}_1)$ by running $\mathsf{USimS.Setup}(\lambda)$ and a secret-public key pair $(\mathsf{sk}_0^{(\mathsf{u})} = \boldsymbol{a}_0, \mathsf{pk}_0^{(\mathsf{u})} = E_{A_0} = [\boldsymbol{a}_0]E_0) \leftarrow \mathsf{USimS.KeyGen}(\mathsf{pp}_{\mathsf{USimS}})$. It provides $\mathsf{pp}_{\mathsf{UKEM}_2}$ and $\mathsf{pk}_0^{(\mathsf{u})}$ and gives $\mathcal{O}^{\mathsf{USimS}}_{\mathsf{up}}(\cdot)$ and $\mathcal{O}^{\mathsf{USimS}}_{\mathsf{dec}}(\cdot)$ oracle access to the USimS adversary $\mathcal{B}$. The adversary $\mathcal{B}$ proceeds to simulate the experiment $\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{UKEM}_2, \mathcal{A}}(\lambda)$ in the following manner.

Setup: The adversary $\mathcal{B}$ initializes epoch $i = 0$. It sets a key derivation function $\mathsf{KDF}_2 : \mathbb{Z}_{2^{q-2}} \rightarrow \{0,1\}^{\mathsf{klen}(\lambda)}$ and sends $\mathsf{pp}_{\mathsf{UKEM}_2} = (\mathsf{pp}_{\mathsf{USimS}}, \mathsf{KDF}_2)$ and $\mathsf{pk}_0^{(\mathsf{u})}$ to the adversary $\mathcal{A}$.

Simulation of the pre challenge oracle query: In the following, we will demonstrate that $\mathcal{B}$ simulates the following oracle perfectly for $\mathcal{A}$.

– $\mathcal{O}^{\mathsf{UKEM}_2}_{\mathsf{up}}(\boldsymbol{\rho}_i)$: Upon receiving a query on the $\boldsymbol{\rho}_i \in \mathbb{Z}_{2^{q-2}}$, the adversary $\mathcal{B}$ calls its own oracle $\mathcal{O}^{\mathsf{USimS}}_{\mathsf{up}}(\boldsymbol{\rho}_i)$ and increments the counter i to $i + 1$.

– $\mathcal{O}^{\mathsf{UKEM}_2}_{\mathsf{dec}}(\mathsf{hct})$: The adversary $\mathcal{B}$ obtains m by calling its oracle $\mathcal{O}^{\mathsf{USimS}}_{\mathsf{dec}}(\mathsf{hct})$ and returns $\mathsf{KDF}_2(m)$.

Simulation of the challenge phase: $\mathcal{B}$ does the following to simulate challenge phase to UKEM_2 adversary $\mathcal{A}$.

i. $\mathcal{B}$ samples $(m_0^*, m_1^*) \xleftarrow{\$} \mathbb{Z}_{2^{q-2}} \times \mathbb{Z}_{2^{q-2}}$ to the USimS challenger $\mathcal{C}$.

ii. In response, the challenger $\mathcal{C}$ uniformly samples $b \xleftarrow{\$} \{0,1\}$ and computes $\mathsf{ct}_{m_b^*} = (\mathsf{ct}^{(1)}_{m_b^*} = [\mathbf{b}]E_0, \mathsf{ct}^{(2)}_{m_b^*} = R_{[\mathbf{b}]\mathsf{pk}_t^{(\mathsf{u})}}(x([2m_0^*+1]P_{[\mathbf{b}]\mathsf{pk}_t^{(\mathsf{u})}}))) \leftarrow \mathsf{USimS.Enc}(\mathsf{pp}_{\mathsf{USimS}}, \mathsf{pk}_t^{(\mathsf{u})}, m_b^*)$

for some $\mathbf{b} \xleftarrow{\$} [-\mu, \mu]^n$ where t is the current epoch. The USimS challenger $\mathcal{C}$ sends $\mathsf{ct}_{m_b^*} = (\mathsf{ct}^{(1)}_{m_b^*}, \mathsf{ct}^{(2)}_{m_b^*})$ to $\mathcal{B}$.

iii. Then the adversary $\mathcal{B}$ sends $(\mathsf{hct}^* = \mathsf{ct}_{m_b^*}, \mathsf{ek}^* = \mathsf{KDF}_2(m_1^*))$ to $\mathcal{A}$

Simulation of the post challenge oracle query: The USimS adversary $\mathcal{B}$ simulates the oracle exactly the same as in the pre-challenge oracle query.

Reveal Phase: Let $\mathcal{O}_{\mathsf{up}}(\boldsymbol{\rho}_{t'})$ is the last query asked by $\mathcal{A}$. After receiving $(\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{up}^*)$ from $\mathcal{C}$ the adversary $\mathcal{B}$ forwards $(\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{up}^*)$ to $\mathcal{A}$ where pk^*, sk^* and up^* are public key, secret key and updated ciphertext respectively at epoch t'.

Guess Phase: Upon receiving $b' \in \{0,1\}$ from the adversary $\mathcal{A}$, the adversary $\mathcal{B}$ submits b' to the challenger $\mathcal{C}$.

Note that,

- If $b = 0$, the USimS challenger $\mathcal{C}$ transmits $\mathsf{ct}_{m_0^*}$ to the adversary $\mathcal{B}$. Subsequently, the UKEM_2 challenger $\mathcal{B}$ forwards the pair $(\mathsf{hct}^* = \mathsf{ct}_{m_0^*}, \mathsf{ek}^* = \mathsf{KDF}_2(m_1^*))$ to the adversary $\mathcal{A}$. Here, the ciphertext hct^* consists of two components:

$$\mathsf{ct}^{(1)}_{m_0^*} = [\mathbf{b}]E_0,\ \mathsf{ct}^{(2)}_{m_0^*} = R_{[\mathbf{b}]\mathsf{pk}_t^{(\mathsf{u})}}(x([2m_0^* + 1]P_{[\mathbf{b}]\mathsf{pk}_t^{(\mathsf{u})}}))$$

where m_0^* and m_1^* are sampled uniformly from $\{0,1\}^{\mathsf{mlen}(\lambda)}$ and $\mathbf{b}$ is sampled uniformly from $[-\mu, \mu]^n$. Since $\mathbf{b}$ is indistinguishable from $\mathsf{KDF}_2(m_0^*)$ due to the security properties of KDF_2, we conclude that

$$\mathsf{hct}^* = \left(\mathsf{ct}^{(1)}_{m_0^*} = [\mathbf{b}]E_0, \mathsf{ct}^{(2)}_{m_0^*} = R_{[\mathbf{b}]\mathsf{pk}_t^{(\mathsf{u})}}(x([2m_0^* + 1]P_{[\mathbf{b}]\mathsf{pk}_t^{(\mathsf{u})}}))\right)$$

is identically distributed to $\left(\widetilde{\mathsf{ct}}^{(1)}_{m_0^*} = [\mathsf{KDF}_1(m_0^*)]E_0, \widetilde{\mathsf{ct}}^{(2)}_{m_0^*} = R_{[\mathsf{KDF}_1(m_0^*)]\mathsf{pk}_t^{(\mathsf{u})}}(x([2m_0^* + 1]P_{[\mathsf{KDF}_1(m_0^*)]\mathsf{pk}_t^{(\mathsf{u})}}))\right)$.
Thus, $(\mathsf{hct}^*, \cdot)$ follows the same distribution as the output of $\mathsf{UKEM}_2.\mathsf{Encaps}(\mathsf{pp}_{\mathsf{UKEM}_2}, \mathsf{pk}_t^{(\mathsf{u})})$.
Furthermore, since m_1^* is uniformly sampled from $\{0,1\}^{\mathsf{mlen}(\lambda)}$, the security of KDF_2 ensures that $\mathsf{ek}^* = \mathsf{KDF}_2(m_1^*)$ is uniformly distributed over the key space $\mathcal{KS} = \{0,1\}^{\mathsf{klen}(\lambda)}$.
Consequently, the tuple $(\mathsf{hct}^*, \mathsf{ek}^*)$ follows the same distribution in the case where $b = 0$ during the challenge phase of the experiment $\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{UKEM}_2, \mathcal{A}}(\lambda)$ as described in Fig. 6.

- If $b = 1$, the USimS challenger $\mathcal{C}$ transmits $\mathsf{ct}_{m_1^*}$ to the adversary $\mathcal{B}$. Subsequently, the UKEM_2 challenger $\mathcal{B}$ forwards the pair $(\mathsf{hct}^* = \mathsf{ct}_{m_1^*}, \mathsf{ek}^* = \mathsf{KDF}_2(m_1^*))$ to the adversary $\mathcal{A}$. Here, the ciphertext hct^* consists of two components

$$\mathsf{ct}^{(1)}_{m_1^*} = [\mathbf{b}]E_0,\ \mathsf{ct}^{(2)}_{m_1^*} = R_{[\mathbf{b}]\mathsf{pk}_t^{(\mathsf{u})}}(x([2m_1^* + 1]P_{[\mathbf{b}]\mathsf{pk}_t^{(\mathsf{u})}}))$$

where m_1^* is sampled uniformly at random from $\{0,1\}^{\mathsf{mlen}(\lambda)}$ and $\mathbf{b}$ is sampled uniformly from $[-\mu, \mu]^n$. Since $\mathbf{b}$ is indistinguishable from $\mathsf{KDF}_1(m_1^*)$ due to the security properties of KDF_1, we conclude that

$$\mathsf{hct}^* = \left(\mathsf{ct}^{(1)}_{m_1^*} = [\mathbf{b}]E_0, \mathsf{ct}^{(2)}_{m_1^*} = R_{[\mathbf{b}]\mathsf{pk}_t^{(\mathsf{u})}}(x([2m_1^* + 1]P_{[\mathbf{b}]\mathsf{pk}_t^{(\mathsf{u})}}))\right)$$

is identically distributed to $\left(\widetilde{\mathsf{ct}}^{(1)}_{m_1^*} = [\mathsf{KDF}_1(m_1^*)]E_0, \widetilde{\mathsf{ct}}^{(2)}_{m_1^*} = R_{[\mathsf{KDF}_1(m_0^*)]\mathsf{pk}_t^{(u)}}(x([2m_1^* + 1]P_{[\mathsf{KDF}_1(m_0^*)]\mathsf{pk}_t^{(u)}}))\right)$.

Thus, $(\mathsf{hct}^* = \mathsf{ct}_{m_1^*}, \mathsf{ek}^* = \mathsf{KDF}_2(m_1^*))$ is identically distributed to the output of $\mathsf{UKEM}_2.\mathsf{Encaps}(\mathsf{pp}_{\mathsf{UKEM}_2}, \mathsf{pk}_t^{(u)})$.

Thus, when $b = 1$, the tuple $(\mathsf{hct}^* = \mathsf{ct}_{m_1^*}, \mathsf{ek}^* = \mathsf{KDF}_2(m_1^*))$ follows the same distribution during the challenge phase of $\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{UKEM}_2,\mathcal{A}}(\lambda)$ as described in Fig. 6.

Therefore, if $\mathcal{A}$ succeeds in $\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{UKEM}_2,\mathcal{A}}(\lambda)$ then $\mathcal{B}$ succeeds in $\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{USimS},\mathcal{B}}(\lambda)$. This implies

$$\Pr[\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CCA}}_{\mathsf{UKEM}_2,\mathcal{A}}(\lambda) = 1] = \Pr[\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CCA}}_{\mathsf{USimS},\mathcal{B}}(\lambda) = 1]$$

Therefore,

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{IND\text{-}CR\text{-}CCA}}_{\mathsf{UKEM}_2,\mathcal{A}}(\lambda) &= \left|\Pr[\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CCA}}_{\mathsf{UKEM}_2,\mathcal{A}}(\lambda) = 1] - \frac{1}{2}\right| \\
&= \left|\Pr[\mathsf{Exp}^{\mathsf{IND\text{-}CR\text{-}CCA}}_{\mathsf{USimS},\mathcal{B}}(\lambda) = 1] - \frac{1}{2}\right| \\
&= \mathsf{Adv}^{\mathsf{IND\text{-}CR\text{-}CCA}}_{\mathsf{USimS},\mathcal{B}}(\lambda)
\end{aligned}$$

□

References

[1] Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: International Conference on the Theory and Applications of Cryptographic Techniques, pp. 255–271 (2003). Springer

[2] Gentry, C., Silverberg, A.: Hierarchical id-based cryptography. In: Advances in Cryptology—ASIACRYPT 2002: 8th International Conference on the Theory and Application of Cryptology and Information Security Queenstown, New Zealand, December 1–5, 2002 Proceedings 8, pp. 548–566 (2002). Springer

[3] Horwitz, J., Lynn, B.: Toward hierarchical identity-based encryption. In: International Conference on the Theory and Applications of Cryptographic Techniques, pp. 466–481 (2002). Springer

[4] Boneh, D., Boyen, X.: Efficient selective-id secure identity-based encryption without random oracles. In: Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings 23, pp. 223–238 (2004). Springer

[5] Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical identity based encryption with constant size ciphertext. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 440–456 (2005). Springer

[6] Brakerski, Z., Lombardi, A., Segev, G., Vaikuntanathan, V.: Anonymous ibe, leakage resilience and circular security from new assumptions. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 535–564 (2018). Springer

[7] Döttling, N., Garg, S.: From selective ibe to full ibe and selective hibe. In: Theory of Cryptography Conference, pp. 372–408 (2017). Springer

[8] Jost, D., Maurer, U., Mularczyk, M.: Efficient ratcheting: Almost-optimal guarantees for secure messaging. In: 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 38, pp. 159–188 (2019). Springer

[9] Dodis, Y., Karthikeyan, H., Wichs, D.: Updatable public key encryption in the standard model. In: Theory of Cryptography: 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8–11, 2021, Proceedings, Part III 19, pp. 254–285 (2021). Springer

[10] 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, pp. 11–22 (2022)

[11] Abou Haidar, C., Passelègue, A., Stehlé, D.: Efficient updatable public-key encryption from lattices. In: International Conference on the Theory and Application of Cryptology and Information Security, pp. 342–373 (2023). Springer

[12] Asano, K., Watanabe, Y.: Updatable public key encryption with strong cca security: Security analysis and efficient generic construction. Cryptology ePrint Archive (2023)

[13] Albrecht, M.R., Benčina, B., Lai, R.W.: Hollow lwe: A new spin: Unbounded updatable encryption from lwe and pce. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 363–392 (2025). Springer

[14] Quantum Computing Growth, .Y.: Accessed: Apr. 4, 2020. [Online]. Available: https://www.statista.com/chart/17896/quantum-computing-developments/

[15] Eaton, E., Jao, D., Komlo, C., Mokrani, Y.: Towards post-quantum key-updatable public-key encryption via supersingular isogenies. In: International Conference on Selected Areas in Cryptography, pp. 461–482 (2021). Springer

[16] Duparc, M., Fouotsa, T.B., Vaudenay, S.: Silbe: an updatable public key encryption scheme from lollipop attacks. Cryptology ePrint Archive (2024)

[17] Castryck, W., Decru, T.: An efficient key recovery attack on sidh. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 423–447 (2023). Springer

[18] Maino, L., Martindale, C., Panny, L., Pope, G., Wesolowski, B.: A direct key recovery attack on sidh. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 448–471 (2023). Springer

[19] Robert, D.: Breaking sidh in polynomial time. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 472–503 (2023). Springer

[20] Kani, E.: The number of curves of genus two with elliptic differentials. (1997)

[21] Moriya, T., Onuki, H., Takagi, T.: SiGamal: A supersingular isogeny-based PKE and its application to a PRF. In: International Conference on the Theory and Application of Cryptology and Information Security, pp. 551–580 (2020). Springer

[22] Fouotsa, T.B., Petit, C.: Sims: a simplification of sigamal. In: Post-Quantum Cryptography: 12th International Workshop, PQCrypto 2021, Daejeon, South Korea, July 20–22, 2021, Proceedings 12, pp. 277–295 (2021). Springer

[23] De Feo, L.: Mathematics of isogeny based cryptography. arXiv preprint arXiv:1711.04062 (2017)

[24] Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: an efficient post-quantum commutative group action. In: International Conference on the Theory and Application of Cryptology and Information Security, pp. 395–427 (2018). Springer

[25] Waterhouse, W.C.: Abelian varieties over finite fields. In: Annales Scientifiques de l'École Normale Supérieure, vol. 2, pp. 521–560 (1969)

[26] Vélu, J.: Isogénies entre courbes elliptiques. CR Acad. Sci. Paris, Séries A **273**, 305–347 (1971)

[27] Pohlig, S., Hellman, M.: An improved algorithm for computing logarithms overgf(p)and its cryptographic significance (corresp.). IEEE Transactions on Information Theory **24**(1), 106–110 (1978) https://doi.org/10.1109/TIT.1978.1055817

A Additional Preliminaries

Algorithm 1 Vélu's formula to compute isogeny

Require: An elliptic curve E_1 given by the generalized Weierstrass equation $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$, with all $a_i \in K$ and a G finite subgroup of $E_1(\overline{K})$.

Ensure: An elliptic curve E_2 and a separable isogeny $\varphi : E_1 \to E_2$ with $G = \ker(\varphi)$ i.e. $E_2 = E_1/G$.

1: For a point $Q = (x(Q), y(Q)) \in G$ with $Q \neq O$, define

$$g_Q = 3(x(Q))^2 + 2a_2x(Q) + a_4 - a_1y(Q),\ h_Q = -2y(Q) - a_1x(Q) - a_3,$$

$$u_Q = h_Q^2 \text{ and } v_Q = \begin{cases} g_Q, & \text{if } 2Q = O \\ 2g_Q - a_1h_Q, & \text{if } 2Q \neq O \end{cases}$$

2: Let G_2 be the points of order 2 in G. Choose $R \subset G$ such that we have a disjoint union

$$G = \{O\} \cup G_2 \cup R \cup (-R).$$

3: Let $S = R \cup G_2$ and set $v = \sum_{Q \in S} v_Q,\ w = \sum_{Q \in S}(u_Q + x(Q)v_Q)$ Then E_2 has the equation $Y^2 + A_1XY + A_3Y = X^3 + A_2X^2 + A_4X + A_6$, where

$$A_1 = a_1,\ A_2 = a_2,\ A_3 = a_3,\ A_4 = a_4 - 5v,\ A_6 = a_6 - (a_1^2 + 4a_2)v - 7w$$

4: The isogeny $\varphi : E_1 \to E_2$ is given below where $(x, y) \in E_1$ and $(X, Y) \in E_2$ satisfying $(X, Y) = \varphi(x, y)$

$$X = x + \sum_{Q \in S}\left(\frac{v_Q}{x - x(Q)} + \frac{u_Q}{(x - x(Q))^2}\right)$$

$$Y = y - \sum_{Q \in S}\Big(u_Q\frac{2y + a_1x + a_3}{(x - x(Q))^3} + v_Q\frac{a_1(x - x(Q)) + y - y(Q)}{(x - x(Q))^2} + \frac{a_1u_Q - g_Qh_Q}{(x - x(Q))^2}\Big)$$

5: **return** E_2 and φ

Remark A.0.1 ([26]). *The complexity of the algorithm is computing isogeny by Vélu's formulae for a finite field K is $O(l\ \log(\#K)^2)$ and in practice, this computation is only feasible when the degree of isogeny l is relatively small (say $l < 1000$).*

A.1 hPKE: An IND-CPA Secure Encryption from CSIDH

We recall below a variant of IND-CPA secure encryption scheme hPKE proposed by Moriya et al. [21]. A hPKE is a tuple of PPT algorithms $\mathsf{hPKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ associated with a message space $\mathcal{MS} = \{0,1\}^{\mathsf{mlen}(\lambda)}$ satisfying the following requirements:

$\mathsf{hPKE.Setup}(\lambda) \rightarrow \mathsf{pp}_{\mathsf{hPKE}} = (p, \ell_1, \ldots, \ell_n, \mu, E_0, \mathcal{H} = \{H_k\}_{k\in\mathcal{K}})$: A trusted party runs this PPT algorithm on input the security parameter λ and proceeds as follows:

i. Chooses a prime $p = 4\Pi_{i=1}^{n}\ell_i - 1$ where ℓ_i's are small distinct odd primes.
ii. Sets a base curve $E_0 : y^2 = x^3 + x \in \mathsf{Ell}_p(\mathcal{O})$ over $\mathbb{F}_p$ with $\mathcal{O} = \mathbb{Z}[\sqrt{-p}]$ and picks an integer μ such that $(2\mu+1)^n \geq \#\mathsf{Cl}(\mathcal{O})$.
iii. Samples a family of keyed hash function $\mathcal{H} = \{H_k\}_{k\in\mathcal{K}}$ where $H_k : \mathbb{F}_p \rightarrow \{0,1\}^{\mathsf{mlen}(\lambda)}$ for each $k \in \mathcal{K}$ where $\mathcal{K}$ is key space.
iv. Outputs the public parameter $\mathsf{pp}_{\mathsf{hPKE}} = (p, \ell_1, \ldots, \ell_n, \mu, E_0, \mathcal{H} = \{H_k\}_{k\in\mathcal{K}})$.

$\mathsf{hPKE.KeyGen}(\mathsf{pp}_{\mathsf{hPKE}}) \rightarrow (\mathsf{sk} = \boldsymbol{a}, \mathsf{pk} = E_A)$: On input the public parameter $\mathsf{pp}_{\mathsf{hPKE}} = (p, \ell_1, \ldots, \ell_n, \mu, E_0, \mathcal{H} = \{H_k\}_{k\in\mathcal{K}})$, a user runs this PPT algorithm to generates a secret-public key pair (sk, pk) in the following manner:

i. Samples $\boldsymbol{a} = (a_1, \ldots, a_n) \xleftarrow{\$} [-\mu, \mu]^n$ and define $[\boldsymbol{a}] = [\mathfrak{l}_1^{a_1} \cdots \mathfrak{l}_n^{a_n}] \in \mathsf{Cl}(\mathcal{O})$ where $\mathfrak{l}_i = \langle \ell_i, \pi - 1\rangle$ and computes $E_A = [\boldsymbol{a}]E_0$.
ii. Sets public key $\mathsf{pk} = E_A$ and secret key $\mathsf{sk} = \boldsymbol{a}$.

$\mathsf{hPKE.Enc}(\mathsf{pp}_{\mathsf{hPKE}}, \mathsf{pk} = E_A, m) \rightarrow \mathsf{ct}_m$: Taking input the public parameter $\mathsf{pp}_{\mathsf{hPKE}} = (p, \ell_1, \ldots, \ell_n, \mu, E_0, \mathcal{H} = \{H_k\}_{k\in\mathcal{K}})$, public key $\mathsf{pk} = E_A$ and a message $m \in \mathcal{MS} = \{0,1\}^{\mathsf{mlen}}$, an encrypter executes this PPT algorithm and proceeds to compute ciphertext as follows:

i. Randomly samples $\mathbf{b} = (b_1, \ldots, b_n) \xleftarrow{\$} [-\mu, \mu]^n$ and define $[\mathbf{b}] = [\mathfrak{l}_1^{b_1} \cdots \mathfrak{l}_n^{b_n}] \in \mathsf{Cl}(\mathcal{O})$.
ii. Computes $\mathsf{ct}_m^{(1)} = E_B = [\mathbf{b}]E_0$ and $\mathsf{ct}_m^{(2)} = m \oplus H_k(M_C([\mathbf{b}]E_A))$ where $M_C([\mathbf{b}]E_A)$ is the Montgomery coefficient of the elliptic curve $[\mathbf{b}]E_A$.
iii. Returns a ciphertext $\mathsf{ct}_m = (\mathsf{ct}_m^{(1)}, \mathsf{ct}_m^{(2)})$.

$\mathsf{hPKE.Dec}(\mathsf{pp}_{\mathsf{hPKE}}, \mathsf{sk} = \boldsymbol{a}, \mathsf{ct}_m = (\mathsf{ct}_m^{(1)}, \mathsf{ct}_m^{(2)})) \rightarrow m/\perp$: A decrypter runs this deterministic algorithm on input the public parameter $\mathsf{pp}_{\mathsf{hPKE}} = (p, \ell_1, \ldots, \ell_n, \mu, E_0, M_C, \mathcal{H} = \{H_k\}_{k\in\mathcal{K}})$, ciphertext ct_m and its secret key $\mathsf{sk} = \boldsymbol{a}$ to recover the plaintext m in the following manner:

i. Computes $h = H_k(M_C([\boldsymbol{a}]\mathsf{ct}_m^{(1)}))$ where $M_C([\boldsymbol{a}]\mathsf{ct}_m^{(1)})$ is the Montgomery coefficient of the elliptic curve $[\boldsymbol{a}]\mathsf{ct}_m^{(1)}$.
ii. Returns $\mathsf{ct}_m^{(2)} \oplus h$.

Correctness: Note that the ciphertext in the protocol hPKE is given by

$$\mathsf{ct}_m = (\mathsf{ct}_m^{(1)} = E_B, \mathsf{ct}_m^{(2)} = m \oplus H_k(M_C([\mathbf{b}]E_A)))$$

and consequently, we have

$$\mathsf{ct}_m^{(2)} \oplus H_k(M_C([\boldsymbol{a}]\mathsf{ct}_m^{(1)}))$$

$$
\begin{aligned}
&= m \oplus H_k(M_C([\mathbf{b}]E_A)) \oplus H_k(M_C([\boldsymbol{a}]\mathsf{ct}_m^{(1)})) \\
&= m \oplus H_k(M_C([\mathbf{b}][\boldsymbol{a}]E_0)) \oplus H_k(M_C([\boldsymbol{a}][\mathbf{b}]E_0)) \\
&= m
\end{aligned}
$$

Theorem 37 ([21]). *The scheme* hPKE *is indistinguishable under the chosen plaintext attack* (IND-CPA) *as per Definition 19 under the* CSSIDDH *assumption given in Definition 13 and assuming* $\mathcal{H} = \{H_k\}_{k\in\mathcal{K}}$ *is entropy smoothing function.*

For the security proof of our updatable public key encryption scheme based on hPKE, we need to establish that hPKE is f-CS + LR secure. This is formally stated in Theorem 38 with the corresponding proof presented in Appendix 3.

Theorem 38. *The construction* hPKE *provides* f-CS + LR *security for* $f : [-\mu, \mu]^n \times \{0,1\}^{\mathsf{mlen}(\lambda)} \to [-\mu, \mu]^n$ *as per Definition 20 under the* CSSIDDH *assumption given in Definition 13, assuming* $\mathcal{H} = \{H_k\}_{k\in\mathcal{K}}$ *is entropy smoothing function and* KDF *is a secure key derivation function as per Definition 16.*

A.2 SimS: An IND-CCA Secure Encryption

Pohlig and Hellman [27] proposed an algorithm in 1978 to solve the discrete logarithm problem for a group of order 2^q for some prime q. Let μ be an element of $\mathbb{Z}_{2^q}$ and P be a generator of $\mathbb{Z}_{2^q}$. Let $\mu_0, \ldots, \mu_{q-1} \in \{0,1\}$ that satisfy $\mu = \sum_{j=0}^{q-1} \mu_j 2^j$. For a given P and μP, we want to compute μ efficiently.

- **For** $i = 0$: We compute $2^{q-1} \cdot \mu P$. If $2^{q-1} \cdot \mu P \equiv O$ then $\mu_0 = 0$ otherwise $\mu_0 = 1$. Therefore, we can obtain the value of μ_0 by computing $2^{q-1} \cdot \mu P$.
- **For** $i = 1, \ldots, q-1$: Define $\mu^{(i)} = \sum_{j=i}^{q-1} \mu_j 2^j$. So,

$$
\mu^{(i)} = \sum_{j=i}^{q-1} \mu_j 2^j = \sum_{j=0}^{q-1} \mu_j 2^j - \sum_{j=0}^{i-1} \mu_j 2^j = \mu - \sum_{j=0}^{i-1} \mu_j 2^j
$$

  If $2^{q-i-1} \cdot \mu^{(i)} P = O$ then $\mu_i = 0$ else $\mu_i = 1$. Note that, $\mu^{(i)} P = \mu P - \sum_{j=0}^{i-1} \mu_j 2^j P$. The condition $2^{q-i-1} \cdot \mu^{(i)} P = O$ equivalent to $2^{q-i-1} \cdot \mu P = 2^{q-i-1} \cdot \sum_{j=0}^{i-1} \mu_j 2^j P$. Let $Q = \mu P$, $P_i = 2^i P$ and $Q_i = 2^i Q$. Therefore, we can obtain the value of μ_i by checking $Q_{q-(i+1)} = \sum_{j=0}^{i-1} \mu_j 2^j P_{q-(i+1)}$.

Algorithm 2 recalls the Pohlig-Hellman algorithm for points on Montgomery curves, which is essential for the SimS scheme. Based on the preceding discussion, in line 7, the case for $i = 0$ is verified according to the given condition where μ is initialized as $\mu = \mu_0$. Similarly, in line 14, for $i = 1, \ldots, q-1$, the case is checked under the condition that μ is updated as $\mu = \mu + \mu_i 2^i$.

Algorithm 2 The Pohlig-Hellman algorithm for Montgomery curves

Require: The Montgomery coefficient $A \in \mathbb{F}_p$ of a supersingular elliptic curve $E_A : y^2 = x^3 + Ax^2 + x$ and x-coordinates of points $P, Q \in E$ of order 2^q each satisfying $Q \in \langle P \rangle$.

Ensure: μ or $2^q - \mu$ such that $Q = \mu P$

$x(P_0) \leftarrow x(P)$;
$x(Q_0) \leftarrow x(Q)$;
for $(i = 1, i < q - 2, i++)$ **do**
    $x(P_i) \leftarrow x(2P_{i-1})$;
    $x(Q_i) \leftarrow x(2Q_{i-1})$;
end for
if $(2^{q-1}x(Q) = 0)$ **then**
    $\mu \leftarrow 0$;
else
    $\mu \leftarrow 1$;
end if
for $(i = 2, i \leq q - 1, i++)$ **do**
    $x(R) \leftarrow x(\mu Q_{q-i})$;
    if $(x(P_{q-i}) \neq x(R))$ **then**
        $\mu \leftarrow \mu + 2^{i-1}$;
    end if
end for
return μ

Algorithm 3 computes the distinguished point P_E of order 2^q on a curve E which is a crucial component of the SimS scheme. The correctness of the Algorithm 3 follows from the following discussion.

Theorem 39 ([22]). *Let $p = 2^q \Pi_{i=1}^n \ell_i - 1$ be a prime such that $p \equiv 3 \pmod 4$ and let E be a supersingular Montgomery curve defined over $\mathbb{F}_p$ satisfying $\mathsf{End}_{\mathbb{F}_p}(E) \equiv \mathbb{Z}[\pi]$. Let $P \in E(\mathbb{F}_p)$ such that $x(P) \neq 0$. If $x(P) \notin (\mathbb{F}_p^*)^2$ then $[\ell_1 \times \cdots \times \ell_n]P$ is a point of order 2^q.*

To determine the x-coordinates of points of order 2^q in E, we must consider elements of $\mathbb{F}_p$ that are not squares. Given that $p = 2^q \Pi_{i=1}^n \ell_i - 1$ with $q > 1$, the following properties hold:

$$\left(\frac{-1}{p}\right) = -1, \left(\frac{2}{p}\right) = 1, \text{ and } \left(\frac{\ell_i}{p}\right) = 1 \text{ for } i \in \{1, \ldots, n\}$$

where $\left(\frac{a}{b}\right)$ denotes Legendre symbols for some integers a and b . Furthermore, let $\ell_1, \ldots, \ell_{n-1}$ be the first $n-1$ smallest odd primes. For every subset $I \subset \{0, 1, \ldots, n-1\}$, we obtain

$$\left(\frac{-\prod_{i \in I} \ell_i}{p}\right) = -1,$$

Using this property, we can sample x from the sequence $-2, -3, -4, \ldots$ and verify whether x corresponds to the x-coordinate of a point in $E(\mathbb{F}_p)$. If so, it suffices to check whether this point has an order divisible by 2^q. However, Theorem 39 guarantees that if such an x is the x-coordinate of a point in $E(\mathbb{F}_p)$ then the corresponding point necessarily has order divisible by 2^q. Hence, an explicit verification step for divisibility by 2^q is not required.

Algorithm 3 [22] Computing the distinguished point P_E of order 2^q on a curve E

Require: The prime $p = 2^q \Pi_{i=1}^n \ell_i - 1$ and Montgomery coefficient $A \in \mathbb{F}_p$ of a supersingular curve $E_A : y^2 = x^3 + Ax^2 + x$ where $\ell_1, \ldots, \ell_n$ are small distinct odd primes.

Ensure: $P_E \in E(\mathbb{F}_p)$ of order 2^q.

1: Set $x \leftarrow -2$;
2: **while** ($x^3 + Ax^2 + x$ is not a square in $\mathbb{F}_p$ and $-x \leq \ell_{n-1} + 1$) **do**
3: Set $x \leftarrow x - 1$;
4: **end while**
5: **if** ($-x \leq \ell_{n-1} + 1$) **then**
6: Set $P \leftarrow (x, y) \in E(\mathbb{F}_p) : y^2 = x^3 + Ax^2 + x$;
7: Set $P_E \leftarrow [\ell_1 \cdots \ell_n]P$;
8: **return** P_E
9: **else**
10: **return** $\perp$
11: **end if**

Remark A.2.1. *Algorithm 3 is deterministic, hence always outputs the same point P_E when the input is unchanged.*

We recall below the IND-CCA secure encryption scheme $\mathsf{SimS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ of Fouotsa et al. [22]. SimS which is associated with a message space $\mathcal{MS} = \mathbb{Z}_{2^{q-2}}$ satisfying the following requirements:

$\mathsf{SimS.Setup}(\lambda) \rightarrow \mathsf{pp}_{\mathsf{SimS}} = (p, q, \ell_1, \ldots, \ell_n, \mu, E_0, R_E)$: A trusted party runs this PPT algorithm on the input the security parameter λ and proceeds as follows:

i. Chooses a prime $p = 2^q \Pi_{i=1}^n \ell_i - 1$ such that $\lambda + 2 \leq q \leq \frac{1}{2} \log p$ where ℓ_i's are small distinct odd primes.
ii. Sets a base curve $E_0 : y^2 = x^3 + x \in \mathsf{Ell}_p(\mathcal{O})$ over $\mathbb{F}_p$ with $\mathcal{O} = \mathbb{Z}[\sqrt{-p}]$ and picks an integer μ such that $(2\mu + 1)^n \geq \#\mathsf{Cl}(\mathcal{O})$.
iii. Consider the function $R_E : \mathbb{F}_p \rightarrow \mathbb{F}_p$ defined as $R_E(x) = \mathsf{int}(\mathsf{bin}(x) \oplus \mathsf{bin}(M_C(E)))$ where $\mathsf{bin}(\cdot)$ and $\mathsf{int}(\cdot)$ represent the operations that convert an element of $\mathbb{F}_p$ into its binary representation and back, respectively. Here, $M_C(E)$ denotes the Montgomery coefficient associated with the elliptic curve E.
iv. Outputs the public parameter $\mathsf{pp}_{\mathsf{SimS}} = (p, q, \ell_1, \ldots, \ell_n, \mu, E_0, R_E)$.

$\mathsf{SimS.KeyGen}(\mathsf{pp}_{\mathsf{SimS}} = (p, q, \ell_1, \ldots, \ell_n, \mu, E_0, R_E)) \rightarrow (\mathsf{sk} = \boldsymbol{a}, \mathsf{pk} = E_A)$: On input the public parameter $\mathsf{pp}_{\mathsf{SimS}} = (p, q, \ell_1, \ldots, \ell_n, \mu, E_0, R_E)$, a user generates a secret-public key pair $(\mathsf{sk}, \mathsf{pk})$ in the following manner:

i. Samples $\boldsymbol{a} = (a_1, \ldots, a_n) \xleftarrow{\$} [-\mu, \mu]^n$ and define $[\boldsymbol{a}] = [\mathfrak{l}_1^{a_1} \cdots \mathfrak{l}_n^{a_n}] \in \mathsf{Cl}(\mathcal{O})$ where $\mathfrak{l}_i = \langle \ell_i, \pi - 1 \rangle$.
ii. Computes $E_A = [\boldsymbol{a}]E_0$.
iii. Returns public key $\mathsf{pk} = E_A$ and secret key $\mathsf{sk} = \boldsymbol{a}$.

$\mathsf{SimS.Enc}(\mathsf{pp}_{\mathsf{SimS}} = (p, q, \ell_1, \ldots, \ell_n, \mu, E_0, R_E), \mathsf{pk} = E_A, m) \rightarrow \mathsf{ct}_m$: Taking as input the public parameter $\mathsf{pp}_{\mathsf{SimS}} = (p, q, \ell_1, \ldots, \ell_n, \mu, E_0, R_E)$, public key $\mathsf{pk} = E_A$ and a message $m \in \mathcal{MS}$, an encrypter runs this PPT algorithm and proceeds to compute ciphertext as follows:

i. Embeds m in $\mathbb{Z}_{2^q}^{\times}$ via $m \rightarrow 2m + 1$.
ii. Randomly samples $\mathbf{b} = (b_1, \ldots, b_n) \xleftarrow{\$} [-\mu, \mu]^n$ and define $[\mathbf{b}] = [\mathfrak{l}_1^{b_1} \cdots \mathfrak{l}_n^{b_n}] \in \mathsf{Cl}(\mathcal{O})$.
iii. Computes $\mathsf{ct}_m^{(1)} = E_B = [\mathbf{b}]E_0$, $E_{AB} = [\mathbf{b}]E_A$ and $P_{AB} = [2m + 1]P_{E_{AB}}$. Here, $P_{E_{AB}}$ is a point on E_{AB} of order 2^q, which is determined using Algorithm 3.
iv. Returns a ciphertext $\mathsf{ct}_m = (\mathsf{ct}_m^{(1)} = E_B, \mathsf{ct}_m^{(2)} = R_{E_{AB}}(x(P_{AB})))$ where $R_{E_{AB}}(x(P_{AB}))) = \mathsf{int}(\mathsf{bin}(x(P_{AB})) \oplus \mathsf{bin}(M_C(E_{AB})))$.

$\mathsf{SimS.Dec}(\mathsf{pp}_{\mathsf{SimS}}, \mathsf{sk} = \boldsymbol{a}, \mathsf{ct}_m = (\mathsf{ct}_m^{(1)}, \mathsf{ct}_m^{(2)})) \rightarrow m/\perp$: A decrypter runs this deterministic algorithm on input the public parameter $\mathsf{pp}_{\mathsf{SimS}} = (p, q, \ell_1, \ldots, \ell_n, \mu, E_0, R_E)$, ciphertext $\mathsf{ct}_m = (\mathsf{ct}_m^{(1)} = E_B, \mathsf{ct}_m^{(2)} = x')$ and its secret key $\mathsf{sk} = \boldsymbol{a}$ to recover the plaintext m by executing the following steps:

i. Verifies that E_B is a supersingular curve.
ii. Computes $E_{BA} = [\boldsymbol{a}]E_B$ and a point $P_{E_{BA}}$ on E_{BA} of order 2^q by using Algorithm 3.
iii. Calculates $R_{E_{BA}}(x')$ and if $R_{E_{BA}}(x')$ is not the x-coordinate of a 2^q-torsion point on the curve E_{BA} then aborts.
iv. Solves the discrete logarithm instance between $P_{BA} = (R_{E_{BA}}(x'), \cdot)$ and $P_{E_{BA}}$ using the Pohlig-Hellman algorithm given in Algorithm 2. Let $m' \in \mathbb{Z}_{2^q}^{\times}$ be the solution of this computation. If $2^{q-1} < m'$ then it changes m' to $2^q - m'$. Finally, it returns the plaintext $\frac{(m'-1)}{2}$.

Correctness: In the protocol SimS, the ciphertext is given by

$$\mathsf{ct}_m = (\mathsf{ct}_m^{(1)}, \mathsf{ct}_m^{(2)}) = (E_B, R_{E_{AB}}(x(P_{AB})))$$

where $\mathsf{ct}_m^{(1)} = E_B = [\mathbf{b}]E_0$ and $\mathsf{ct}_m^{(2)} = R_{E_{AB}}(x(P_{AB}))$. The value P_{AB} is computed as

$$P_{AB} = [2m + 1]P_{E_{AB}},$$

where $P_{E_{AB}}$ is a point on E_{AB} of order 2^q, determined using Algorithm 3.
We have

$$E_{AB} = [\mathbf{b}]E_A = [\mathbf{b}][\boldsymbol{a}]E_0 = [\boldsymbol{a}][\mathbf{b}]E_0 = [\boldsymbol{a}]E_B = E_{BA}$$

Since $P_{E_{AB}}$ and $P_{E_{BA}}$ are points on E_{AB} and E_{BA}, respectively, both of order 2^q and given that $E_{AB} = E_{BA}$, it follows that $P_{E_{AB}} = P_{E_{BA}}$. As Algorithm 3 is deterministic, it yields the same distinguished point for both cases.

We now demonstrate that $x(P_{BA}) = x(P_{AB})$. We proceed as follows:

$$
\begin{aligned}
& x(P_{BA}) \\
& = R_{E_{BA}}(\mathsf{ct}_m^{(2)}) \\
& = R_{E_{BA}}(R_{E_{BA}}(x(P_{AB}))) \\
& = R_{E_{BA}}(\mathsf{int}(\mathsf{bin}(x(P_{AB})) \oplus \mathsf{bin}(M_C(E_{BA})))) \\
& = \mathsf{int}(\mathsf{bin}(\mathsf{int}(\mathsf{bin}(x(P_{AB})) \oplus \mathsf{bin}(M_C(E_{BA})))) \oplus \mathsf{bin}(M_C(E_{BA}))) \\
& = \mathsf{int}(\mathsf{bin}(x(P_{AB})) \oplus \mathsf{bin}(M_C(E_{BA})) \oplus \mathsf{bin}(M_C(E_{AB}))) \\
& = \mathsf{int}(\mathsf{bin}(x(P_{AB}))), \text{ since } E_{AB} = E_{BA} \\
& = x(P_{AB})
\end{aligned}
$$

As $P_{AB} = P_{BA}$ and $P_{E_{AB}} = P_{E_{BA}}$, it follows that $P_{BA} = [2m+1]P_{E_{BA}}$. Using the Pohlig-Hellman algorithm (Algorithm 2), we recover the discrete logarithm instance between P_{AB} and $P_{E_{BA}}$ obtaining either $m' = (2m+1)$ or $m' = 2^q - (2m+1)$. Since $m \in \mathbb{Z}_{2^{p-2}}$, we have $0 \leq m \leq 2^{q-2} - 1$, which implies

$$0 \leq 2m + 1 \leq 2^{p-1} - 1 < 2^{p-1}.$$

Furthermore, since $2^{p-1} < 2^q - (2m+1)$, if $m' > 2^{p-1}$, i.e., if $m' = 2^q - (2m+1)$, we adjust m' to

$$m' = 2^p - m' = (2m+1).$$

Thus, we conclude that $m = \frac{m'-1}{2}$. This guarantees the correctness of the protocol.

Theorem 40 ([22])**.** *The scheme* SimS *is* IND-CPA *secure as per Definition 19 under the assumption that $R_{E_{AB}}$ satisfies the second property of a randomizing function (see Definition 3) and the* CSSIDDH *assumption (see Definition 13) holds.*

For the security proof of our updatable public key encryption scheme based on SimS, we need to establish that SimS is f-CS + LR secure. This is formally stated in Theorem 41 with the corresponding proof presented in Appendix 4.

Theorem 41. *The scheme* SimS *is f-CS + LR secure for $f : [-\mu, \mu]^n \times \mathbb{Z}_{2^{q-2}} \to [-\mu, \mu]^n$ as per Definition 20 under the assumption that $R_{E_{AB}}$ satisfies the second property of a randomizing function (see Definition 3),* KDF *is a secure key derivation function (see Definition 16) and the* CSSIDDH *assumption given in Definition 13 holds.*