Indifferentiability of Public-Key Encryption: Theory Meets Practice*

Taiyu Wang† Cong Zhang† Hong-Sheng Zhou‡ Jiayi Ai†
Zhihong Jia† Wenli Wang† Jian Liu† Xin Wang§
Li Lin§ Kui Ren† Chun Chen†

August 19, 2026

Abstract

Public-key encryption (PKE) is a fundamental primitive in modern cryptography, and many PKE schemes have been standardized and widely deployed. To reason about security in complex and highly compositional environments, Zhandry and Zhang (CRYPTO 2020) initiated the study of indifferentiability for public-key cryptosystems. However, their PKE construction departs substantially from practical design paradigms, and to date no PKE construction following standardized group-based templates is known to achieve indifferentiability.

In this work, we further investigate indifferentiability for PKE, asking whether it can be achieved for practical group-based schemes following standardized templates. We provide evidence that the answer is yes: slightly augmented ECIES/PSEC-style constructions can achieve provable indifferentiability in idealized models. Our contributions are threefold:

- **Identifying a barrier in the existing ideal PKE model:** We revisit the ideal PKE definition of Zhandry and Zhang and identify a requirement—namely, pseudorandom public keys and ciphertexts—that constitutes an *inherent barrier* to achieving indifferentiability from generic group-based constructions.
- **Redefining the idealized model for PKE:** We propose a revised ideal PKE model that removes this pseudorandomness requirement, thereby enabling indifferentiable PKE from practical group-based schemes.
- **Making standardized-template PKE indifferentiable:** We consider ECIES and PSEC, standardized in ISO/IEC 18033-2, and show that, under slight augmentation, ECIES/PSEC-style constructions are indifferentiable from our revised ideal PKE in idealized models.

Keywords: Public-key encryption, generic group model, indifferentiability, KEM/DEM paradigm, ECIES, PSEC.

*The work was mainly supported by National Key Research and Development Program of China, Grant No. 2023YFB3106000.
†The State Key Laboratory of Blockchain and Data Security, Zhejiang University, congresearch@zju.edu.cn
‡Virginia Commonwealth University
§Digital Technologies, Ant Group

Contents

1 Introduction **4**
- 1.1 Our Results . . . 5
  - 1.1.1 Impossibility results within generic group model (Section 3) . . . 5
  - 1.1.2 Redefining ideal PKE (Section 4) . . . 7
  - 1.1.3 Making standardized-template PKE indifferentiable (Section 5 and 6) . . . 8
- 1.2 Interpretation . . . 13
- 1.3 Organization . . . 13

2 Preliminaries **13**
- 2.1 Public-key encryption . . . 14
- 2.2 Idealized models . . . 14
- 2.3 Indifferentiability framework . . . 15

3 Impossibility of indifferentiable PKE in GGM **16**

4 Redefining ideal PKE **18**

5 Making PSEC indifferentiable **19**
- 5.1 Ideal KEM with validity testing . . . 20
- 5.2 Indifferentiable PSEC-KEM . . . 20
- 5.3 Composition of ideal KEM and ideal AE . . . 21

6 Making ECIES indifferentiable **23**
- 6.1 ECIES-compatible ideal KEM . . . 23
- 6.2 Making ECIES-KEM indifferentiable . . . 23
- 6.3 Composition of ECIES-compatible ideal KEM and ideal AE . . . 24

A Proof of Theorem 3.1 **29**

B Proof of Theorem 5.1 **36**
- B.1 Transition from **Ideal** to $\mathbf{G}_1$ . . . 42
- B.2 Transition from $\mathbf{G}_1$ to $\mathbf{G}_2$ . . . 42
- B.3 Properties under the absence of failure conditions . . . 45
- B.4 Transition from $\mathbf{G}_2$ to $\mathbf{G}_3$ . . . 56
- B.5 Transition from $\mathbf{G}_3$ to $\mathbf{G}_4$ . . . 59
- B.6 Transition from $\mathbf{G}_4$ to **Real** . . . 62
- B.7 Bounding the failure conditions . . . 63

C Proof of Theorem 5.2 **67**
- C.1 Transition from **Ideal** to $\mathbf{G}_1$ . . . 71
- C.2 Transition from $\mathbf{G}_1$ to $\mathbf{G}_2$ . . . 72
- C.3 Properties under the absence of failure conditions . . . 74
- C.4 Transition from $\mathbf{G}_2$ to $\mathbf{G}_3$ . . . 79
- C.5 Transition from $\mathbf{G}_3$ to $\mathbf{G}_4$ . . . 81
- C.6 Transition from $\mathbf{G}_4$ to **Real** . . . 84
- C.7 Bounding the failure conditions . . . 85

D Proof of Theorem 6.1 **87**
D.1 Transition from **Ideal** to $\mathbf{G}_1$ . . . 91
D.2 Transition from $\mathbf{G}_1$ to $\mathbf{G}_2$ . . . 92
D.3 Properties under the absence of failure conditions . . . 94
D.4 Transition from $\mathbf{G}_2$ to $\mathbf{G}_3$ . . . 103
D.5 Transition from $\mathbf{G}_3$ to $\mathbf{G}_4$ . . . 106
D.6 Transition from $\mathbf{G}_4$ to **Real** . . . 108
D.7 Bounding the failure conditions . . . 109

E Proof of Theorem 6.2 **112**
E.1 Transition from **Ideal** to $\mathbf{G}_1$ . . . 117
E.2 Transition from $\mathbf{G}_1$ to $\mathbf{G}_2$ . . . 117
E.3 Properties under the absence of failure conditions . . . 119
E.4 Transition from $\mathbf{G}_2$ to $\mathbf{G}_3$ . . . 126
E.5 Transition from $\mathbf{G}_3$ to $\mathbf{G}_4$ . . . 128
E.6 Transition from $\mathbf{G}_4$ to **Real** . . . 131
E.7 Bounding the failure conditions . . . 132

1 Introduction

Public-key encryption (PKE) schemes constitute a fundamental building block of modern secure communication. They are widely used in standardized protocols and applications as a core mechanism for achieving confidentiality, including in secure messaging and other encrypted communication systems.

Challenges in complex systems and composable security. While PKE schemes are well studied in isolation, their deployment in complex, multi-layered systems can give rise to unforeseen vulnerabilities. For example, related-key (RKA) attacks [Wee12] exploit structured relationships between keys in different protocol components, while key-dependent message (KDM) attacks [BRS03] arise when ciphertexts encrypt messages that depend on secret keys. These attacks underscore the importance of *composable security*: a PKE scheme should remain secure even when combined with other protocols or used in settings for which it was not originally designed. Achieving security in such robust contexts requires theoretical frameworks that go beyond classical *game-based* security definitions [BR06].

Indifferentiability. A promising framework for achieving composable security is indifferentiability. Introduced by Maurer et al. [MRH04], indifferentiability provides a *simulation-based* paradigm for arguing that a real-world cryptographic construction can safely replace an idealized object in any environment, as long as no efficient distinguisher can tell the two apart. Consequently, proving that a scheme is indifferentiable from an ideal object implies that it can be securely used within complex protocols, inheriting the security guarantees of the ideal object even when composed with other cryptographic components. This property is particularly attractive for the design and analysis of complex systems, as it enables rigorous security proofs in which the construction can be treated as a black-box ideal object. Notably, indifferentiability has been widely used to strengthen the security analysis of hash functions and block ciphers [CDMP05, BDPV08, HKT11, DSSL16], and more recently to characterize authenticated encryption with associated data (AEAD) [BF18].

Limitations of existing indifferentiable PKE. Until recently, indifferentiability had been studied primarily for symmetric primitives and domain extenders. Zhandry and Zhang [ZZ20] made the first attempt to extend indifferentiability to public-key cryptosystems, including non-interactive key exchange (NIKE), public-key encryption, and digital signatures. However, we observe that their indifferentiable PKE construction is based on a rather nonstandard paradigm, and the resulting construction appears somewhat artificial. Specifically, their construction follows the paradigm

$$\mathsf{NIKE} \rightarrow \mathsf{Deterministic\ PKE} \rightarrow \mathsf{PKE}.$$

That is, starting from an ideal NIKE functionality, the construction first realizes an indifferentiable deterministic public-key encryption (DPKE) scheme, and then uses this ideal DPKE functionality to obtain an indifferentiable (randomized) PKE scheme. This two-step route, however, is somewhat indirect and does not reflect the standard way public-key encryption is typically constructed.

Furthermore, the encryption algorithm of the resulting indifferentiable PKE scheme in [ZZ20] is itself rather artificial. More concretely, let $(\mathsf{KG}_{\mathsf{NIKE}}, \mathsf{SHK}_{\mathsf{NIKE}})$ denote an ideal NIKE functionality, the encryption algorithm Enc takes the public key pk, the message m, and the nonce r as inputs and outputs a ciphertext of the following form:

$$\mathsf{c}_1 = \mathsf{KG}_{\mathsf{NIKE}}\big(H(\mathsf{pk}||\mathsf{msg})\big),$$
$$\mathsf{c}_2 = P(\mathsf{pk}||\mathsf{msg}) \oplus \mathsf{SHK}_{\mathsf{NIKE}}(\mathsf{pk}, H(\mathsf{pk}||\mathsf{msg})),$$

where H and H' are two random oracles, and $\underline{\mathsf{msg} := \mathsf{m}||H'(\mathsf{pk}||\mathsf{m}||\mathsf{r})||0^\lambda}$.

In particular, the plaintext is first transformed into the highly derived form msg, and the ciphertext is then computed based on pk‖msg. This makes the construction substantially different from practical/standardized public-key encryption schemes and rather artificial from a deployment perspective.

Practical paradigm: KEM/DEM. In practice, public-key encryption schemes are most often realized through the KEM/DEM paradigm, which combines a key-encapsulation mechanism with a symmetric data-encryption

mechanism. This paradigm has been widely standardized and deployed. For instance, ISO/IEC 18033-2 [Sho04] specifies ECIES as a KEM/DEM-based hybrid encryption scheme for efficient public-key encryption, and also includes PSEC-KEM, a provably secure elliptic-curve-based key-encapsulation mechanism. More recently, NIST FIPS 203 [Nat24] standardized ML-KEM, formerly known as Kyber, as a post-quantum key-encapsulation mechanism.

Taken together, these observations reveal a clear gap between existing indifferentiable PKE constructions and the public-key encryption schemes that are actually standardized and deployed in practice. This gap motivates the following natural question:

Can indifferentiability be achieved for practical, standardized public-key encryption schemes?

Answering this question is essential for bridging the gap between theoretical security guarantees and the demands of real-world cryptographic applications. An ideal resolution would enable existing standardized public-key encryption components to be seamlessly integrated into complex protocols while retaining strong, holistic security guarantees, thereby reconciling the indifferentiability framework of [ZZ20] with practical cryptosystem design.

1.1 Our Results

In this work, we give an affirmative answer to the above question in the setting of standardized group-based PKE. Specifically, we focus on two group-based templates specified in ISO/IEC 18033-2, namely ECIES and PSEC. A natural approach to establishing indifferentiability would be to replace the underlying groups in ECIES and PSEC with idealized generic groups, and then prove that the resulting constructions are indifferentiable from the ideal PKE functionality defined in [ZZ20]. However, our results show that this approach is fundamentally impossible.

Our contributions are threefold. First, we provide evidence that, under the existing idealization of PKE in [ZZ20], no group-based public-key encryption scheme in the generic group model can achieve indifferentiability.[1] Second, we trace this impossibility to the pseudorandom requirement built into the existing idealization, and propose a revised ideal PKE functionality equipped with an additional capability that removes this property. Third, we prove that slightly augmented versions of the standardized public-key encryption schemes PSEC and ECIES are indifferentiable from our revised ideal PKE functionality.

To better understand our results, we begin with a concise overview of the high-level notion of ideal PKE in [ZZ20].

- **Ideal PKE**. The ideal PKE can be viewed, roughly speaking, as an oracle offering three interfaces:
  - KG is a random injection, that takes a private key as input and outputs the corresponding public key;
  - Enc is a random injection, that takes a public key, a message, and a nonce as inputs and outputs the corresponding ciphertext;
  - Dec is a deterministic function, that takes a private key and a ciphertext as inputs and outputs the underlying plaintext (or $\perp$), subject to the correctness condition[2].

1.1.1 Impossibility results within generic group model (Section 3)

According to the definition of ideal PKE above, the ideal functionality inherently provides both pseudorandom public keys and pseudorandom ciphertexts. More concretely, we say that a public key pk is *valid* if there exists a secret key sk such that $\mathsf{pk} = \mathsf{KG}(\mathsf{sk})$. Pseudorandomness of public keys then requires that,

[1] This already rules out the most natural route toward obtaining positive examples, since existing standardized group-based encryption schemes are generic in nature.

[2] $\mathsf{Dec}(\mathsf{sk}, \mathsf{c}) = \mathsf{m}$ if $\exists\ \mathsf{r}$ such that $\mathsf{Enc}(\mathsf{KG}(\mathsf{sk}), \mathsf{m}, \mathsf{r}) = \mathsf{c}$, and $\perp$ otherwise.

given an element from the public-key space, it is hard to distinguish whether this element is a valid public key or merely a uniformly random string.

To establish indifferentiability, Zhandry and Zhang [ZZ20] construct an indifferentiable NIKE scheme in the random oracle model and the ideal permutation model, starting from a standard-model NIKE whose public keys are indistinguishable from uniformly random strings. Their construction relies on Elligator [BHKL13, Tib14], a technique that maps uniformly random points on prime-order elliptic curves to uniformly random strings. Combined with their NIKE-to-PKE transformation, this yields a group-based instantiation of indifferentiable PKE.

We emphasize, however, that Elligator is an encoding-specific technique rather than a generic group operation, even though it has been used in practical tools such as Tor's obfs4 protocol [The23]. As such, it does not faithfully capture how standardized public-key encryption schemes are typically designed and deployed. This motivates the question of whether indifferentiable PKE can be achieved using only generic group operations, thereby better aligning with standardized group-based PKE schemes.

We prove that, with respect to the existing idealization of PKE, this goal is impossible. Namely, there exists no efficient generic construction in the GGM that is indifferentiable from the ideal PKE functionality. Specifically, let $\mathcal{G}$ denote a generic group. We show that for any generic construction $C^{\mathcal{G}}$, there exists a statistical distinguisher $\mathcal{D}$ that makes only polynomially many queries and distinguishes $C^{\mathcal{G}}$ from the ideal PKE with non-negligible probability.

We establish this impossibility via a black-box separation argument. Observe that if a construction $C^{\mathcal{G}}$ were indifferentiable from an ideal PKE, then by the definition of indifferentiability, any cryptographic primitive—provided its security can be expressed as a single-stage game [RSS11]—that exists relative to the ideal PKE would also exist relative to the generic group model. Our strategy therefore proceeds in two steps: (1) we identify a cryptographic primitive that exists relative to the ideal PKE; and (2) we demonstrate a black-box separation between this primitive and the generic group model.[3]

Specifically, we identify the relevant primitive as PKE schemes that are secure against message-recovery attacks (MRA) and whose public keys and ciphertexts are pseudorandom. Informally, a PKE scheme is MRA-secure if no adversary can recover the entire plaintext with non-negligible probability. Its public keys and ciphertexts are pseudorandom if no adversary can distinguish valid ones[4] from uniformly random strings.

By the definition of ideal PKE in [ZZ20], Enc always outputs uniformly random ciphertexts, regardless of whether it is invoked on a valid public key or on a random string. Moreover, Dec does not provide a validity test for ciphertexts: for a fixed sk, the fact that $\mathsf{Dec}(\mathsf{sk}, \mathsf{c}) = \perp$ only shows that c is invalid under sk, but does not exclude the possibility that c is valid under some other private key. Thus, public keys and ciphertexts in the ideal PKE are statistically indistinguishable from random strings. Consequently, the above PKE primitive can be realized trivially in the ideal PKE model, even against query-efficient but computationally unbounded adversaries.

Thus, it suffices to establish the black-box separation within the GGM.

Theorem 1.1 (Informal)**.** *In the generic group model, there exists no MRA-secure public-key encryption scheme with pseudorandom public keys and pseudorandom ciphertexts against adversaries that are query-efficient but computationally unbounded.*

We next briefly explain the high-level idea behind this black-box separation. Let $\mathcal{G} = (\mathcal{G}^{\mathsf{label}}, \mathcal{G}^{\mathsf{add}})$ denote a generic group model. It's straightforward to build a MRA-secure PKE within $\mathcal{G}$,

$$\mathsf{KG}(x) := \mathcal{G}^{\mathsf{label}}(x);\ \mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r}) := \mathcal{G}^{\mathsf{label}}(\mathsf{r}), H(\mathsf{pk}^{\mathsf{r}}) \oplus \mathsf{m}.$$

However, this construction fails to achieve pseudorandom public keys, since valid public keys are always

[3]For simplicity, we assume a sparse encoding of the generic group, ensuring that any query-efficient adversary outputs a valid group element that was not previously obtained from oracle queries only with negligible probability.

[4]Here, a *valid* public key means a public key in the image of KG, and a *valid* ciphertext means a ciphertext in the image of the encryption algorithm Enc on some valid public key, message, and randomness.

valid group elements in $\mathcal{G}$, and their validity can be verified trivially[5].

Building on prior black-box separations for public-key primitives with additional properties in the generic group model, Ji et al. [JZW+24] established a separation for NIKE against key-recovery attacks (KRA) with shorter public keys in the GGM using longer encodings.[6] Subsequently, Wang et al. [WZZ+25] proved a black-box separation for KRA-secure NIKE with admissible encodings in the GGM.

The core insight of [JZW+24, WZZ+25] is that, if the additional properties of the NIKE scheme ensure that *no algorithm can derive a valid group element from public keys without knowing the corresponding discrete logarithm*, then one can construct an adversary that breaks KRA security using only labeling queries.[7] In other words, the attack can be carried out already in the labeling-only generic group model, denoted by $\mathcal{G}^{\mathsf{label}}$. More concretely, when the adversary is restricted to labeling queries, the labeling-only generic group model can be faithfully simulated in the random oracle model. This makes it possible to construct a KRA adversary in a manner analogous to the black-box separation of Impagliazzo and Rudich [IR89].

We adopt a strategy similar to that of [WZZ+25] and extend the black-box separation to the public-key encryption setting. Specifically, we show that pseudorandomness of both public keys and ciphertexts likewise prevents algorithms from deriving valid group elements without knowing the corresponding discrete logarithms.

At a high level, suppose that there exists an algorithm A which, given $(\mathsf{pk}, \mathsf{c})$ as input, outputs a valid group element in the generic group model with non-negligible probability, while not knowing its discrete logarithm. Since public keys and ciphertexts are pseudorandom, replacing $(\mathsf{pk}, \mathsf{c})$ with uniformly random strings should preserve A's success probability up to a negligible loss. Yet, as shown in [Zha22], if an algorithm's inputs are independent of the generic group representation, then it cannot output a valid group element without knowing its discrete logarithm, except with negligible probability.

Remark 1.1. *A careful reader may observe that, besides ideal PKE, Zhandry and Zhang also formalize an ideal NIKE functionality, which likewise enforces pseudorandom public keys. We emphasize that our impossibility result applies equally to the NIKE setting: under the existing definition of ideal NIKE, no group-based construction using only generic group operations can be indifferentiable from the ideal NIKE functionality.*

1.1.2 Redefining ideal PKE (Section 4)

From the above analysis, we observe that the primary obstacle to constructing indifferentiable PKE within the GGM lies in the requirement of pseudorandom public keys and ciphertexts. To overcome this, we redefine the ideal PKE by removing this requirement. In a nutshell,

- **Revised ideal PKE**. The revised ideal PKE can be viewed, roughly speaking, as an oracle offering three interfaces:
  - KG is defined as in [ZZ20];
  - Enc is defined as a random injection equipped with *public-key validity testing*, with invalid inputs indicated by a distinguished symbol $\perp_{\mathsf{pk}}$. Specifically, $\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$ outputs $\perp_{\mathsf{pk}}$ if and only if there exists no sk such that $\mathsf{KG}(\mathsf{sk}) = \mathsf{pk}$ (i.e., pk is invalid).
  - Dec is a deterministic function subject to correctness and *ciphertext validity testing*, with invalid ciphertexts indicated by another distinguished symbol $\perp_{\mathsf{c}}$. Specifically, $\mathsf{Dec}(\mathsf{sk}, \mathsf{c})$ outputs $\perp_{\mathsf{c}}$ if and only if there exist no sk', m', and r' such that $\mathsf{Enc}(\mathsf{KG}(\mathsf{sk}'), \mathsf{m}', \mathsf{r}') = \mathsf{c}$ (i.e., c is invalid).

By incorporating validity testing, our revised ideal PKE removes the requirement of pseudorandom public keys and ciphertexts, thereby opening up the opportunity to construct indifferentiable PKE within the GGM.

[5] By the definition of the GGM [Sho97], $\mathcal{G}^{\mathsf{add}}(\mathsf{str}, \mathsf{str})$ returns $\perp$ if and only if str is not a valid group element; that is, there exists no x such that $\mathcal{G}^{\mathsf{label}}(x) = \mathsf{str}$.

[6] The separation established in [JZW+24] is not a standard black-box separation, since it relies on the NIKE scheme and the GGM sharing the same security parameter.

[7] Once the discrete logarithm of each group element is known, addition queries can be trivially simulated using labeling queries.

Remark 1.2. *A careful reader might argue that the revised ideal PKE, compared to that in [ZZ20], may leak additional information due to the inclusion of validity testing. We emphasize, however, that these modifications do not weaken the intended security guarantees. Specifically, in the real-world group-based construction, invalid public keys or ciphertexts are trivially detectable, and protocols necessarily behave differently on such malformed inputs. The inclusion of a validity-testing mechanism merely makes this behavior explicit in the ideal model, rather than endowing the adversary with any fundamentally new capability. At the same time, for all well-formed public keys and ciphertexts, the revised ideal model preserves both correctness and full randomness. Thus, our revised model only alters how to respond to malformed inputs. Moreover, the well-received generic group model exposes validity testing for group elements without compromising any meaningful security guarantees.*

1.1.3 Making standardized-template PKE indifferentiable (Section 5 and 6)

Building on our revised PKE model, we show that two standardized public-key encryption schemes—namely, ECIES and PSEC, as specified in ISO/IEC 18033-2—achieve indifferentiability under minor modifications. Consistent with ISO/IEC 18033-2, these schemes follow the KEM/DEM paradigm, which our constructions likewise adopt.

In the KEM/DEM paradigm, the encryptor first executes the KEM encapsulation algorithm Encaps to obtain a ciphertext c together with a shared key k. The message m is then encrypted using a DEM—i.e., a symmetric-key encryption scheme—under the key k. To achieve indifferentiability, our construction models the DEM component using the ideal authenticated encryption model (ideal AE) proposed by Barbosa and Farshim [BF18]. For the KEM component, we define it as follows:

- **Ideal KEM**. The ideal KEM can be viewed as an oracle that provides three interfaces:
  - KG as a random injection, that takes a private key as input and outputs the corresponding public key;
  - Encaps as a random function with validity testing on public keys, that takes a public key and a nonce as input and outputs the corresponding ciphertext and shared key (or $\perp_{\mathsf{pk}}$). Moreover, it satisfies an injectivity requirement: each distinct pair of public key and nonce maps to a unique ciphertext;
  - Decaps as a deterministic function subject to correctness and validity testing on ciphertexts, that takes a private key and a ciphertext as inputs and outputs the underlying shared key (or $\perp$ for decryption errors or $\perp_{\mathsf{c}}$ for invalid ciphertexts).

Note that the injectivity of Encaps ensures that each ciphertext can only be decapsulated by a unique private key. This property suffices to guarantee correctness and, moreover, enables an *explicit rejection* capability: decapsulation using any incorrect private key deterministically returns $\perp$. Next, we present the high-level idea for making PSEC-KEM indifferentiable from the ideal KEM above.

Making PSEC-KEM indifferentiable. Let $\mathcal{G} = (\mathcal{G}^{\mathsf{label}}, \mathcal{G}^{\mathsf{add}})$ be a generic group model. Following the design of PSEC-KEM, for any secret key $\mathsf{sk} \in \mathbb{Z}_N$, we define the key-generation algorithm by setting $\mathsf{KG}(\mathsf{sk}) := \mathcal{G}^{\mathsf{label}}(\mathsf{sk})$. We show, however, that in this setting there exists a differentiator that proceeds as follows:

1. Sample $\mathsf{sk}_1, \mathsf{sk}_2 \in \mathbb{Z}_N$ uniformly at random and compute $\mathsf{sk} = \mathsf{sk}_1 + \mathsf{sk}_2 \bmod N$.
2. Obtain corresponding pk_1, pk_2, and pk via queries to KG.
3. Query the simulator for operation $\mathcal{G}^{\mathsf{add}}$ on inputs $(\mathsf{pk}_1, \mathsf{pk}_2)$ to obtain pk'.
4. Check whether $\mathsf{pk} = \mathsf{pk}'$.

Observe that, in Step 3, the simulator is required to answer the addition query correctly. However, since the simulator does not know sk_1 or sk_2, it cannot compute the corresponding secret value sk and hence

cannot obtain pk through the KG interface. Consequently, the simulation fails with high probability. The reason is that KG coincides with $\mathcal{G}^{\mathsf{label}}$, and therefore preserves the algebraic structure of $\mathcal{G}$.

To resolve this issue, we modify KG by introducing a random oracle $\mathcal{H}_1$ and defining $\mathsf{KG}(\mathsf{sk}) := \mathcal{G}^{\mathsf{label}}\big(\mathcal{H}_1(\mathsf{sk})\big)$. This modification breaks the algebraic structure described above and therefore prevents the preceding attack. More concretely, we have $\mathsf{pk}_1 = \mathcal{G}^{\mathsf{label}}(\mathcal{H}_1(\mathsf{sk}_1)), \mathsf{pk}_2 = \mathcal{G}^{\mathsf{label}}(\mathcal{H}_1(\mathsf{sk}_2)), \mathsf{pk} = \mathcal{G}^{\mathsf{label}}(\mathcal{H}_1(\mathsf{sk}))$. Thus, even if $\mathsf{sk} = \mathsf{sk}_1 + \mathsf{sk}_2$, we have $\mathcal{H}_1(\mathsf{sk}) \neq \mathcal{H}_1(\mathsf{sk}_1) + \mathcal{H}_1(\mathsf{sk}_2)$ except with negligible probability. Consequently, the public key corresponding to sk is algebraically independent of the sum of the public keys corresponding to sk_1 and sk_2. This allows the simulator to answer the addition query $\mathcal{G}^{\mathsf{add}}(\mathsf{pk}_1, \mathsf{pk}_2)$ using a fresh random valid public key.

We next turn to the encapsulation algorithm and explain how it is made indifferentiable. Following the design of PSEC-KEM, Encaps takes as input a public key pk and a nonce seed. If pk is not a valid group element, it outputs $\perp_{\mathsf{pk}}$. Otherwise, it computes $y \leftarrow \mathcal{H}_2(\mathsf{seed})$, and outputs a ciphertext $(\mathsf{c}_1, \mathsf{c}_2)$ together with a shared key k, where

$$\mathsf{c}_1 := \mathcal{G}^{\mathsf{label}}(y),\ \mathsf{c}_2 := \mathsf{seed} \oplus \mathcal{H}_3\big(\mathsf{c}_1, \mathsf{pk}^y\big),\ \text{and } \mathsf{k} := \mathcal{H}_4(\mathsf{seed}).$$

For decapsulation, the algorithm Decaps takes as input the secret key sk and a ciphertext $(\mathsf{c}_1, \mathsf{c}_2)$. If c_1 is not a valid group element, it outputs $\perp_{\mathsf{c}}$. Otherwise, it computes

$$\mathsf{seed} \leftarrow \mathsf{c}_2 \oplus \mathcal{H}_3(\mathsf{c}_1, \mathsf{c}_1^{\mathcal{H}_1(\mathsf{sk})}).$$

If $\mathsf{c}_1 = \mathcal{G}^{\mathsf{label}}\big(\mathcal{H}_2(\mathsf{seed})\big)$, then Decaps returns the shared key $\mathsf{k} = \mathcal{H}_4(\mathsf{seed})$; otherwise, it outputs $\perp$.

Clearly, this original PSEC-KEM construction fails to achieve indifferentiability, as the shared key k depends solely on the nonce seed. Under the ideal KEM definition introduced above, the encapsulation algorithm Encaps is modeled as a random function. Consequently, for distinct public keys $\mathsf{pk}_1 \neq \mathsf{pk}_2$, the outputs $\mathsf{Encaps}(\mathsf{pk}_1, \mathsf{seed})$ and $\mathsf{Encaps}(\mathsf{pk}_2, \mathsf{seed})$ differ with overwhelming probability, even when the same nonce seed is used.

To remedy this deficiency, we naturally modify the key derivation to bind the shared key to the public key by defining $\mathsf{k} := \mathcal{H}_4(\mathsf{pk}, \mathsf{seed})$.

However, this construction is still not provably indifferentiable. The differentiator can proceed as follows:

1. Sample $\mathsf{sk} \in \mathbb{Z}_N$ uniformly at random, and obtain $\mathsf{pk} \leftarrow \mathsf{KG}(\mathsf{sk})$ and $\mathcal{H}_1(\mathsf{sk})$.
2. Sample seed uniformly at random, and obtain $(\mathsf{c}_1, \mathsf{c}_2, \mathsf{k}) \leftarrow \mathsf{Encaps}(\mathsf{pk}, \mathsf{seed})$.
3. Compute $\mathsf{mask} := \mathsf{seed} \oplus \mathsf{c}_2$.
4. Check whether $\mathsf{mask} = \mathcal{H}_3\left(\mathsf{c}_1, \mathsf{c}_1^{\mathcal{H}_1(\mathsf{sk})}\right)$.

In the real world, mask is always equal to $\mathcal{H}_3(\mathsf{c}_1, \mathsf{c}_1^{\mathcal{H}_1(\mathsf{sk})})$. In contrast, in the ideal world the simulator has access only to sk, c_1, and c_2, and therefore can recover at most the shared key $\mathcal{H}_4(\mathsf{pk}, \mathsf{seed})$. Without knowledge of the nonce seed, the simulator cannot correctly simulate the value $\mathcal{H}_3(\mathsf{c}_1, \mathsf{c}_1^{\mathcal{H}_1(\mathsf{sk})})$ except with negligible probability.

Note that the simulator's failure stems from the fact that the differentiator obtains the ciphertext $(\mathsf{c}_1, \mathsf{c}_2)$ without revealing the underlying nonce. To overcome this issue, we introduce an additional random oracle and correspondingly modify the encapsulation algorithm Encaps. Specifically, on input a public key pk and a nonce r, the algorithm first computes $\mathsf{seed} \leftarrow \mathcal{H}^*(\mathsf{pk}, \mathsf{r})$ and then proceeds exactly as in the construction above.

Clearly, the introduction of this additional random oracle endows the simulator with the ability to control the value of seed. In particular, when the differentiator obtains $(\mathsf{c}_1, \mathsf{c}_2, \mathsf{k})$ via $\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$, it still learns no information about seed. Consequently, the simulator may sample seed uniformly at random and implicitly set $\mathsf{seed} \leftarrow \mathcal{H}^*(\mathsf{pk}, \mathsf{r})$, after which it can properly respond to the query $\mathcal{H}_3(\mathsf{c}_1, \mathsf{c}_1^{\mathcal{H}_1(\mathsf{sk})})$ by returning $\mathsf{seed} \oplus \mathsf{c}_2$.

Unfortunately, even with this modification, our construction still fails to achieve indifferentiability. We next describe another differentiator, defined as follows:

1. Sample $\mathsf{sk} \in \mathbb{Z}_N$ uniformly at random, and obtain $\mathsf{pk} \leftarrow \mathsf{KG}(\mathsf{sk})$.
2. Sample seed uniformly at random rather than obtaining it from $\mathcal{H}^*$.
3. Compute c_1 and c_2 from $(\mathsf{pk}, \mathsf{seed})$ following the construction.
4. Check whether $\mathsf{Decaps}(\mathsf{sk}, \mathsf{c}_1, \mathsf{c}_2) = \mathcal{H}_4(\mathsf{pk}, \mathsf{seed})$.

In the real world, by correctness, $\mathsf{Decaps}(\mathsf{sk}, \mathsf{c}_1, \mathsf{c}_2)$ always equals $\mathcal{H}_4(\mathsf{pk}, \mathsf{seed})$. In contrast, in the ideal world the simulator cannot answer the query $\mathcal{H}_4(\mathsf{pk}, \mathsf{seed})$ in a consistent manner, except with negligible probability. Although the KEM admits two possible methods for deriving the shared key, the simulator—lacking knowledge of the secret key sk—cannot emulate decapsulation directly. Its only viable strategy is therefore to invoke $\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$ in order to simulate the computation of the ciphertext components c_1 and c_2 (as in Step 3 of the differentiator).

However, by the definition of Encaps, the computation of c_1—given seed—is independent of the public key pk. Consequently, at the moment c_1 is generated, the simulator has no information about pk. Although pk is known to the simulator when computing c_2, it is infeasible for the simulator to find a nonce r such that the first component of $\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$ coincides with the prescribed value c_1. Therefore, the pair $(\mathsf{c}_1, \mathsf{c}_2)$ cannot be faithfully emulated via an $\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$ query. As a result, the simulator is unable to ensure consistency with $\mathcal{H}_4(\mathsf{pk}, \mathsf{seed})$, except with negligible probability.

To address this issue, we introduce an ideal permutation model $(\mathcal{P}, \mathcal{P}^{-1})$[8] to bind c_1 and c_2 together. We then modify the encapsulation algorithm as follows: after computing $(\mathsf{c}_1, \mathsf{c}_2, \mathsf{k})$, the algorithm outputs (c, k), where $\mathsf{c} := \mathcal{P}(\mathsf{c}_1 || \mathsf{c}_2)$, and the decapsulation algorithm is modified accordingly.

We now explain why this additional model prevents the above attack. Observe that, when computing $\mathcal{P}(\mathsf{c}_1 || \mathsf{c}_2)$, the simulator already knows the public key pk. Consequently, the simulator can sample a nonce r at random, issue a query $\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$, and obtain the pair (c, k). It then programs the permutation by setting $\mathcal{P}(\mathsf{c}_1 || \mathsf{c}_2) = \mathsf{c}$. Thereafter, when the differentiator queries $\mathcal{H}_4(\mathsf{pk}, \mathsf{seed})$, the simulator can respond consistently by returning the value k.

With these modifications, we can finally establish the following theorem:

Theorem 1.2 (Informal)**.** *The slightly modified PSEC-KEM is indifferentiable from the ideal KEM.*

Remark 1.3. *Compared to the original PSEC-KEM, our construction introduces two modifications: an additional random oracle $\mathcal{H}^*$ and an ideal permutation $\mathcal{P}$. We emphasize that these modifications are not ad hoc, but rather reflect a recurring phenomenon in indifferentiability analyses. As shown by Coron et al. [CDMP05], the standard Merkle–Damgård transform underlying widely deployed hash functions such as SHA-1 and MD5 fails to achieve indifferentiability in its original form. Crucially, their resolution is not to discard the construction, but to introduce minimal tweaks that preserve the original design while rendering it provably indifferentiable and efficient in practice. Similarly, Barbosa and Farshim [BF18] demonstrate that many well-established AEAD schemes cannot be proven indifferentiable as originally specified; instead, small modifications are required to better align the real-world construction with the ideal AEAD.*

Why does "validity testing" help? In our revised ideal PKE, the oracles Enc and Dec can verify the validity of public keys and ciphertexts, respectively, thereby strengthening the simulator. To illustrate the power of validity testing, we first present a differentiator that breaks indifferentiability in the absence of validity testing.

1. Sample $\mathsf{sk} \in \mathbb{Z}_N$ uniformly at random and obtain $\mathsf{pk} \leftarrow \mathsf{KG}(\mathsf{sk})$.
2. Sample a random string str, and set $\mathsf{str}_0 := \mathsf{pk}$ and $\mathsf{str}_1 := \mathsf{str}$.

[8] This model can be readily instantiated via the ideal cipher model with a singleton key space. Moreover, indifferentiable ideal ciphers can be constructed from random oracles [CHK+16].

3. Flip a coin b, and query the simulator for the group operation $\mathcal{G}^{\mathsf{add}}$ on inputs $(\mathsf{str}_b, \mathsf{str}_b)$ to obtain $\hat{\mathsf{str}}$.
4. Check whether $\hat{\mathsf{str}} \neq \perp$ when $b = 0$ and $\hat{\mathsf{str}} = \perp$ when $b = 1$.

Clearly, in the real world, the test always passes. In contrast, in the ideal world, the simulator cannot distinguish between str_0 and str_1, and thus fails to respond to the queries correctly with high probability.

In our revised ideal PKE, this type of attack is easily prevented. Specifically, for any str, the simulator can verify its validity by querying $\mathsf{Enc}(\mathsf{str}, \mathsf{m}, \mathsf{r})$ for any m and r. If Enc returns $\perp_{\mathsf{pk}}$, the simulator responds with $\perp$.

While we have prevented certain natural attacks, it remains necessary to establish indifferentiability against all possible attacks. To this end, we employ a carefully designed simulation strategy and prove indifferentiability through a sequence of hybrids in Section 5 and Appendix B.

Making ECIES-KEM indifferentiable. In real-world deployments, ECIES is far more widely deployed than PSEC. Accordingly, we next present the high-level approach for achieving indifferentiability in an ECIES-KEM.

Following the same methodology as in the indifferentiable PSEC-KEM setting, we define the key generation algorithm as $\mathsf{KG}(\mathsf{sk}) := \mathcal{G}^{\mathsf{label}}\big(\mathcal{H}_1(\mathsf{sk})\big)$. In the original ECIES-KEM, Encaps takes as input a public key pk and a nonce r. If pk is not a valid group element, it outputs $\perp_{\mathsf{pk}}$. Otherwise, it outputs a ciphertext c together with a shared key k, where

$$\mathsf{c} := \mathcal{G}^{\mathsf{label}}(\mathsf{r}),\ \mathsf{k} := \mathcal{H}(\mathsf{pk}^{\mathsf{r}}).$$

For decapsulation, the algorithm Decaps takes as input the secret key sk and a ciphertext c. If c is not a valid group element, it outputs $\perp_{\mathsf{c}}$. Otherwise, it returns the shared key $\mathcal{H}\left(\mathsf{c}^{\mathcal{H}_1(\mathsf{sk})}\right)$.

In contrast to PSEC-KEM, decapsulation in ECIES-KEM never outputs $\perp$. Specifically, for any ciphertext c that is a valid group element and for any secret key sk, $\mathsf{Decaps}(\mathsf{sk}, \mathsf{c})$ always returns a shared key. However, under the ideal KEM definition—which requires injectivity—each ciphertext must be decapsulable under a unique secret key, while decapsulation under any other secret key outputs $\perp$. To bridge this gap and establish indifferentiability for ECIES-KEM, we next introduce a variant of the ideal KEM that aligns with the structure of ECIES.

- **ECIES-compatible ideal KEM**. Similar to the ideal KEM, this oracle provides three interfaces:
  - KG is defined identically to that of the ideal KEM.
  - Encaps as a random function with validity testing on public keys, that takes a public key and a nonce as input and outputs the corresponding ciphertext and shared key (or $\perp_{\mathsf{pk}}$). Furthermore, the ciphertext is determined solely by the nonce, and the mapping from nonces to ciphertexts is injective.
  - Decaps is defined identically to that of the ideal KEM.

Note that, under this definition, any valid public key combined with the same nonce yields the same ciphertext, and every valid ciphertext admits a well-defined decapsulation outcome. In particular, the decapsulation algorithm never rejects, thereby matching the structural properties of ECIES.

Remark 1.4. *A careful reader might observe a gap between this ideal variant and the ideal PKE, since decryption in the ideal PKE outputs $\perp$ when an incorrect secret key is used, whereas this ideal variant admits no rejection mechanism. We stress that this gap can be bridged by the capabilities of the ideal authenticated encryption (AE), which we employ as the DEM. Following the KEM/DEM paradigm, the shared key derived by the KEM is used as the secret key for the ideal AE. If an incorrect KEM secret key is used, then with overwhelming probability* Decaps *outputs a different shared key, which is consequently rejected by the decryption algorithm of the ideal AE.*

We next present the high-level idea for making ECIES-KEM indifferentiable from this ideal variant. Analogously to the use of $\mathcal{H}_1$ in key generation, we first apply the same modifications and set

$$\mathsf{c} := \mathcal{G}^{\mathsf{label}}\big(\mathcal{H}_2(\mathsf{r})\big),\ \mathsf{k} := \mathcal{H}_3(\mathsf{pk}^{\mathcal{H}_2(\mathsf{r})}).$$

However, this construction is not provably indifferentiable. The differentiator works as follows:

1. Sample $\mathsf{sk}_1, \mathsf{sk}_2 \in \mathbb{Z}_N$ uniformly at random.
2. Query the simulator for $\mathcal{H}_1$ to obtain $\mathcal{H}_1(\mathsf{sk}_1)$ and $\mathcal{H}_1(\mathsf{sk}_2)$.
3. Sample $x_1 \in \mathbb{Z}_N$ uniformly at random and compute x_2 such that
$$x_1 \cdot \mathcal{H}_1(\mathsf{sk}_1) = x_2 \cdot \mathcal{H}_1(\mathsf{sk}_2).$$
4. Query the simulator for $\mathcal{G}^{\mathsf{label}}$ on inputs x_1 and x_2 to obtain c_1 and c_2;
5. Check whether $\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{c}_1) = \mathsf{Decaps}(\mathsf{sk}_2, \mathsf{c}_2)$.

In the real world, this test always passes. In the ideal world, however, such a collision is hard to realize, since ideal Encaps behaves as a random function. To prevent this attack, we bind both the public key and the ciphertext into the key derivation, and instead define

$$\mathsf{c} := \mathcal{G}^{\mathsf{label}}\big(\mathcal{H}_2(\mathsf{r})\big),\ \mathsf{k} := \mathcal{H}_3(\mathsf{pk}, \mathsf{c}, \mathsf{pk}^{\mathcal{H}_2(\mathsf{r})}).$$

This modification rules out the above attack. The key point is that, by including both the public key and the ciphertext in the input to $\mathcal{H}_3$[9], the output of the key-derivation procedure becomes tied to the entire pair $(\mathsf{pk}, \mathsf{c}, \mathsf{pk}^{\mathcal{H}_2(\mathsf{r})})$ rather than only to the underlying group element $\mathsf{pk}^{\mathcal{H}_2(\mathsf{r})}$. Consequently, even if $x_1 \cdot \mathcal{H}_1(\mathsf{sk}_1) = x_2 \cdot \mathcal{H}_1(\mathsf{sk}_2)$, the derived keys remain independent as long as $(\mathsf{pk}_1, \mathsf{c}_1) \neq (\mathsf{pk}_2, \mathsf{c}_2)$. Thus, in the real world, $\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{c}_1) \neq \mathsf{Decaps}(\mathsf{sk}_2, \mathsf{c}_2)$ except with negligible probability. This allows the simulator to answer the $\mathcal{G}^{\mathsf{label}}$ queries in Step 4 with fresh random valid ciphertexts.

Unfortunately, this construction still fails to achieve indifferentiability. Observe that in the ideal KEM there are two distinct validity tests: one for public keys and one for ciphertexts. In the above construction, any valid public key is also a valid ciphertext, and vice versa. However, under the ideal KEM definition, a valid public key is, with overwhelming probability, not a valid ciphertext.

To eliminate this unintended dependency between public keys and ciphertexts, we introduce an additional ideal permutation model $(\mathcal{P}, \mathcal{P}^{-1})$ and modify the encapsulation as follows:

$$\mathsf{c} := \mathcal{P}\left(\mathcal{G}^{\mathsf{label}}\big(\mathcal{H}_2(\mathsf{r})\big)\right), \mathsf{k} := \mathcal{H}_3(\mathsf{pk}, \mathsf{c}, \mathsf{pk}^{\mathcal{H}_2(\mathsf{r})}).$$

With these modifications, we establish the following theorem in Section 6.

Theorem 1.3 (Informal)**.** *The slightly modified ECIES-KEM is indifferentiable from the ECIES-compatible ideal KEM.*

Composition of ideal KEM and ideal AE. Given an ideal KEM (including the ECIES-compatible ideal variant) and an ideal AE, we next show how to compose them in a manner that is indifferentiable from our revised ideal PKE. In the KEM/DEM paradigm, the encryption Enc takes as input the public key pk, the message m and the nonce r. It first computes $(\mathsf{c}_\mathsf{k}, \mathsf{k}) \leftarrow \mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$, then computes $\mathsf{c}_\mathsf{m} \leftarrow \mathsf{AE.Enc}(\mathsf{k}, \mathsf{m})$, and finally outputs $(\mathsf{c}_\mathsf{k}, \mathsf{c}_\mathsf{m})$. Accordingly, the decryption computes $\mathsf{k} \leftarrow \mathsf{Decaps}(\mathsf{sk}, \mathsf{c}_\mathsf{k})$ and outputs the message $\mathsf{AE.Dec}(\mathsf{k}, \mathsf{c}_\mathsf{m})$.

Following reasoning similar to the analyses above, achieving indifferentiability requires binding c_k and c_m together. To this end, we introduce an additional ideal permutation model $(\mathcal{P}, \mathcal{P}^{-1})$ and modify the encryption so that, after computing $(\mathsf{c}_\mathsf{k}, \mathsf{c}_\mathsf{m})$, it outputs the ciphertext $\mathsf{c} := \mathcal{P}(\mathsf{c}_\mathsf{k} || \mathsf{c}_\mathsf{m})$, with the decryption modified accordingly. We next establish the following theorem in Section 5.3, with full proofs given in Appendix C.

Theorem 1.4 (Informal)**.** *The slightly modified KEM/DEM composition of the ideal KEM and the ideal AE is indifferentiable from the revised ideal PKE.*

[9]Our design matches the approach used in TLS 1.3, where including both public keys in the key-derivation function effectively authenticates the protocol transcript.

1.2 Interpretation

We interpret our results from three perspectives.

Theoretical perspective. Our results highlight, for the first time, that indifferentiability can be reconciled with standard group-based constructions. Prior work [ZZ20] mandated pseudorandom encodings for public keys and ciphertexts, which can only be achieved by non-generic tools such as Elligator. This approach, however, conflicts with the generic-group setting and does not reflect how group elements are naturally represented. By contrast, we show that it is not necessary to erase algebraic structure. Instead, one can refine the definition of ideal PKE so that it accounts for validity testing—a property that real-world implementations inherently possess. This adjustment removes the impossibility barrier, preserves the clean algebraic nature of the ideal, and provides a more faithful abstraction of group-based public-key schemes.

Practical perspective. Recognizing validity testing in the definition of the ideal aligns theoretical guarantees with what implementations already do (e.g., curve or encoding checks). In this refined setting, we prove that modestly augmented versions of well-established and standardized schemes—ECIES, and PSEC in ISO/IEC 18033-2—achieve indifferentiability from the revised ideal PKE. This not only closes a long-standing gap between theory and practice but also enables composable analyses of real-world building blocks without introducing ad hoc encodings. As a result, widely deployed public-key infrastructure can now be studied under the powerful indifferentiability framework, yielding security guarantees that extend to complex protocol compositions.

Broader lesson. Beyond specific constructions, our work illustrates a more general principle: when defining ideal objects in the indifferentiability framework, one should capture the unavoidable structural properties of real schemes rather than abstract them away. Once this reality is embraced—here, through explicit validity testing—indifferentiability becomes achievable for standard group-based primitives. We view this as evidence that our revised ideals strike the right balance between abstraction and realizability, offering a foundation for future progress in composable security for public-key cryptography.

1.3 Organization

The remainder of this paper is organized as follows. Section 2 introduces the notation used throughout the paper and recalls the basic definitions of public-key encryption, idealized models, and the indifferentiability framework. In Section 3, we show that no construction in the generic group model can be indifferentiable from the ideal PKE of Zhandry and Zhang [ZZ20], with the full separation proof deferred to Appendix A. Section 4 presents our revised definition of ideal PKE. In Section 5, we prove that a slightly augmented PSEC-based construction is indifferentiable from our revised ideal PKE, with the full proofs deferred to Appendices B and C. Finally, Section 6 establishes the indifferentiability of a slightly augmented ECIES-based construction from our revised ideal PKE, with the full proof and additional details provided in Appendices D and E.

2 Preliminaries

Notations. Throughout this paper, let $\lambda \in \mathbb{N}$ denote the security parameter. For any non-empty finite set $\mathcal{S}$, we use $s \xleftarrow{\$} \mathcal{S}$ to denote a random sample s from $\mathcal{S}$ according to the uniform distribution, and $|\mathcal{S}|$ to denote the size of $\mathcal{S}$. We denote by $\mathcal{S}_1 \times \mathcal{S}_2$ the Cartesian product of the sets $\mathcal{S}_1$ and $\mathcal{S}_2$. For a probabilistic algorithm Alg, we overload the notation and write $X \xleftarrow{\$} \mathsf{Alg}(I)$ to denote that the variable X is obtained by running Alg on input I (where I may be a tuple, i.e., $I = (I_1, ..., I_n)$). When the algorithm Alg is deterministic, we write $X \leftarrow \mathsf{Alg}(I)$ instead.

For bit strings X and Y, we write $X||Y$ to denote the concatenation. We let $\{0,1\}^*$ denote the set of all finite-length bit strings, and $\{0,1\}^n$ denote the set of all n-bit strings, where $n \in \mathbb{N}$.

A positive function $\mathsf{negl}(\cdot)$ is said to be negligible if, for every positive polynomial $p(\cdot)$, there exists a constant $\lambda_0 > 0$ such that $\mathsf{negl}(\lambda) < 1/p(\lambda)$ for all $\lambda > \lambda_0$. A function $\rho(\cdot)$ is called noticeable in λ if $1/\rho(\lambda)$ is polynomial in λ.

Cryptosystems. A cryptosystem Σ consists of a set of algorithms. Here, Σ is accessible through two distinct interfaces: $\Sigma.\mathrm{hon}$ and $\Sigma.\mathrm{adv}$, where $\Sigma.\mathrm{hon}$ provides an honest interface through which the system can be accessed by all parties in a black-box manner, and $\Sigma.\mathrm{adv}$ models the adversarial access to the inner working part of Σ.

2.1 Public-key encryption

Definition 2.1 (Public-Key Encryption). *Denote $\mathcal{SK} := \{\mathcal{SK}_\lambda\}_{\lambda\in\mathbb{N}}$, $\mathcal{PK} := \{\mathcal{PK}_\lambda\}_{\lambda\in\mathbb{N}}$, $\mathcal{M} := \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$, $\mathcal{R} := \{\mathcal{R}_\lambda\}_{\lambda\in\mathbb{N}}$, and $\mathcal{C} := \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ as the private-key space, public-key space, message space, randomness space, and ciphertext space, respectively. A public-key encryption scheme Π_{PKE} over $\mathcal{SK}$, $\mathcal{PK}$, $\mathcal{M}$, $\mathcal{R}$, and $\mathcal{C}$ is a tuple of algorithms defined as follows:*

- *The* public-key generation algorithm $\Pi_{\mathsf{PKE}}.\mathsf{KG}$ *takes as input a randomly chosen private key $\mathsf{sk} \in \mathcal{SK}_\lambda$, and outputs a public key $\mathsf{pk} \in \mathcal{PK}_\lambda$;*
- *The* encryption algorithm $\Pi_{\mathsf{PKE}}.\mathsf{Enc}$ *takes as input a public key, a message $\mathsf{m} \in \mathcal{M}_\lambda$, and a randomness $\mathsf{r} \in \mathcal{R}_\lambda$, and outputs a ciphertext $\mathsf{c} \in \mathcal{C}_\lambda$;*
- *The* decryption algorithm $\Pi_{\mathsf{PKE}}.\mathsf{Dec}$ *takes as input a private key and a ciphertext, and outputs a message m or a symbol $\perp$, where $\perp$ indicates a failed decryption.*

Correctness. *We say Π_{PKE} achieves perfect correctness if for any $\mathsf{sk} \in \mathcal{SK}_\lambda$, $\mathsf{m} \in \mathcal{M}_\lambda$, and $\mathsf{r} \in \mathcal{R}_\lambda$, with $\mathsf{pk} \leftarrow \Pi_{\mathsf{PKE}}.\mathsf{KG}(\mathsf{sk})$ and $\mathsf{c} \leftarrow \Pi_{\mathsf{PKE}}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$, it holds that*

$$\Pr\left[\Pi_{\mathsf{PKE}}.\mathsf{Dec}(\mathsf{sk}, \mathsf{c}) = \mathsf{m}\right] = 1,$$

Security. *We say Π_{PKE} is secure against message-recovery attack (MRA) if for any PPT (adversarial) machine $\mathcal{A}$, there is a negligible function negl such that*

$$|\Pr[\mathcal{A}(\mathsf{pk}, \mathsf{c}) = \mathsf{m}] - 1/|\mathcal{M}_\lambda|| \leq \mathsf{negl}(\lambda).$$

Here, for randomly chosen $\mathsf{sk} \xleftarrow{\$} \mathcal{SK}_\lambda$, $\mathsf{m} \xleftarrow{\$} \mathcal{M}_\lambda$, and $\mathsf{r} \xleftarrow{\$} \mathcal{R}_\lambda$, compute $\mathsf{pk} \leftarrow \Pi_{\mathsf{PKE}}.\mathsf{KG}(\mathsf{sk})$ and $\mathsf{c} \leftarrow \Pi_{\mathsf{PKE}}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$, respectively.

2.2 Idealized models

In this subsection, we introduce idealized models covered in this paper, including the Random Oracle Model (ROM) [BR93], the Random Permutation Model (RPM) [RS08], the Generic Group Model (GGM) [Sho97], and the Ideal Authenticated Encryption Model [BF18]. In each idealized model, all entities including the adversary and the challenger, are provided with access to the corresponding oracle. Below we will specify the behavior of the oracle in each idealized model.

Definition 2.2 (Random Oracle Model [BR93]). *Let $\mathcal{I}_{*,S}$ denote the set of all functions mapping $\{0,1\}^*$ to a finite set S. The random oracle model $\mathcal{H}$ is an idealized model that samples h uniformly at random from $\mathcal{I}_{*,S}$. Every algorithm can query $\mathcal{H}$ with an input x, obtaining the corresponding output $h(x) \in S$.*

Definition 2.3 (Random Permutation Model [RS08]). *Let $\mathcal{I}_{S,S}$ denote the set of all permutations on a finite set S. The random permutation model $\mathcal{P}$ is an idealized model that samples a permutation π uniformly at random from $\mathcal{I}_{S,S}$. Every algorithm can query $\mathcal{P}$ with an input $x \in S$ for both π and its inverse π^{-1}, obtaining the corresponding output $\pi(x) \in S$ or $\pi^{-1}(x) \in S$, respectively.*

Definition 2.4 (Generic Group Model [Sho97]). *Let $\mathcal{I}_{\mathbb{Z}_N,S}$ denote the set of all injections from $\mathbb{Z}_N$ to S, where $S = \{0,1\}^\ell$. The generic group model $\mathcal{G}_{N,\ell}$ is an idealized model that samples an injection σ uniformly at random from $\mathcal{I}_{\mathbb{Z}_N,S}$ and provides two interfaces, $\mathcal{G}^{\mathsf{label}}_{N,\ell}$ and $\mathcal{G}^{\mathsf{add}}_{N,\ell}$. Concretely,*

- *the labeling interface $\mathcal{G}^{\mathsf{label}}_{N,\ell}$ takes as input $x \in \mathbb{Z}_N$ and returns $\sigma(x)$;*
- *the addition interface $\mathcal{G}^{\mathsf{add}}_{N,\ell}$ takes as input $\mathsf{str}_1, \mathsf{str}_2 \in S$ and does the following: if there are $x_1, x_2 \in \mathbb{Z}_N$ such that $\sigma(x_1) = \mathsf{str}_1$ and $\sigma(x_2) = \mathsf{str}_2$, returns $\sigma(x_1 + x_2)$; otherwise it returns $\perp$.*

Definition 2.5 (Ideal Authenticated Encryption Model [BF18]). *Let $\mathcal{K}$, $\mathcal{N}$, $\mathcal{M}$, and $\mathcal{C}$ be the shared-key space, nonce space, message space, and ciphertext space. Define Inj as the set of all functions $f : \mathcal{K} \times \mathcal{N} \times \mathcal{M} \to \mathcal{C}$ that are injective on $\mathcal{M}$:*

$$\forall(\mathsf{k},\mathsf{n},\mathsf{m}),(\mathsf{k},\mathsf{n},\mathsf{m}') \in \mathcal{K}\times\mathcal{N}\times\mathcal{M} :\ \mathsf{m} \neq \mathsf{m}' \implies f(\mathsf{k},\mathsf{n},\mathsf{m}) \neq f(\mathsf{k},\mathsf{n},\mathsf{m}'),$$

and satisfy the length restriction: $\forall(\mathsf{k},\mathsf{n},\mathsf{m}) \in \mathcal{K}\times\mathcal{N}\times\mathcal{M} : |f(\mathsf{k},\mathsf{n},\mathsf{m})| = |\mathsf{m}| + \lambda$, where $\lambda \in \mathbb{N}$ is the security parameter. The ideal authenticated encryption *is an idealized model that uniformly samples a function f from Inj, and provides two interfaces, $\mathcal{AE}$ and $\mathcal{AD}$. Specifically,*

- *$\mathcal{AE}$ takes as input $(\mathsf{k},\mathsf{n},\mathsf{m}) \in \mathcal{K}\times\mathcal{N}\times\mathcal{M}$, and returns $f(\mathsf{k},\mathsf{n},\mathsf{m}) \in \mathcal{C}$;*
- *$\mathcal{AD}$ takes as input $(\mathsf{k},\mathsf{n},\mathsf{c}) \in \mathcal{K}\times\mathcal{N}\times\mathcal{C}$, and if there exists $\mathsf{m} \in \mathcal{M}$ such that $f(\mathsf{k},\mathsf{n},\mathsf{m}) = \mathsf{c}$, return m; otherwise, return $\perp$.*

2.3 Indifferentiability framework

The framework of indifferentiability is proposed by Maurer, Renner, and Holenstein [MRH04], which formalizes a set of necessary and sufficient conditions for securely replacing one cryptosystem with another in an arbitrary environment. This framework is used to justify the structural soundness of various cryptographic primitives, including hash functions [CDMP05, DRS09], block ciphers [ABD+13, CHK+16, DSSL16, GWL23], domain extenders [CDMS10], authenticated encryption with associated data [BF18], and public key cryptosystems [ZZ20]. It can also be used to study the relationship between idealized models [ZZ23]. Within the context of the indifferentiability framework, it is customary to consider that a cryptosystem either implements certain ideal objects denoted as $\mathcal{F}$, or it is a construction denoted as $C^{\mathcal{F}'}$ that relies on underlying ideal objects $\mathcal{F}'$.

Definition 2.6 (Indifferentiability [MRH04]). *Let Σ_1 and Σ_2 be two cryptosystems and $\mathcal{S}$ be a simulator. The indifferentiability advantage of a distinguisher $\mathcal{D}$ against (Σ_1,Σ_2) with respect to $\mathcal{S}$ is*

$$\mathrm{Adv}^{\mathsf{indif}}_{\Sigma_1,\Sigma_2,\mathcal{S}}(\mathcal{D}) := \Pr[\mathbf{Real}^{\Sigma_1}(\mathcal{D}) = 1] - \Pr[\mathbf{Ideal}^{\Sigma_2}_{\mathcal{S}}(\mathcal{D}) = 1],$$

where two games, $\mathbf{Real}^{\Sigma_1}$ and $\mathbf{Ideal}^{\Sigma_2}_{\mathcal{S}}$, are defined in Fig. 1. We say Σ_1 is indifferentiable from Σ_2, if there exists an efficient simulator $\mathcal{S}$ such that for any efficient distinguisher $\mathcal{D}$, the advantage above is negligible. Moreover, we say Σ_1 is statistically indifferentiable from Σ_2, if there exists an efficient simulator such that, for any unbounded distinguisher $\mathcal{D}$, the advantage above is negligible.

Below, we also use the notations in [BF18] and consider the definition above to two systems with interfaces as:

$$(\Sigma_1.\mathrm{hon}(X), \Sigma_1.\mathrm{adv}(x)) := (\Pi^{\mathcal{F}_1}(X), \mathcal{F}_1(x))$$
$$(\Sigma_2.\mathrm{hon}(X), \Sigma_2.\mathrm{adv}(x)) := (\mathcal{F}_2(X), \mathcal{F}_2(x)),$$

where $\mathcal{F}_1$ and $\mathcal{F}_2$ are two ideal objects sampled from their distributions and $\Pi^{\mathcal{F}_1}$ is a construction of $\mathcal{F}_2$ by calling $\mathcal{F}_1$. MRH prove the composition theorem for the framework of indifferentiability; for simplicity, we give a game-based formalization from [RSS11].

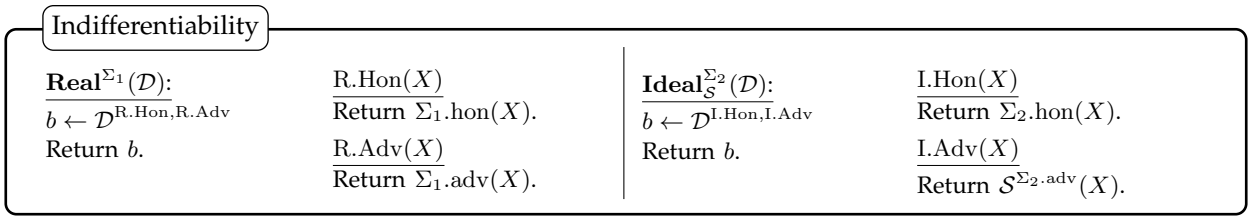

Figure 1: Indifferentiability of Σ_1 and Σ_2, where $\mathcal{S}$ is the simulator and $\mathcal{D}$ is the adversary.

Theorem 2.1 (Composition Theorem [MRH04]). *Let $\Sigma_1 := (\Pi^{\mathcal{F}_1}, \mathcal{F}_1)$ and $\Sigma_2 := (\mathcal{F}_2, \mathcal{F}_2)$ be two systems that Σ_1 is indifferentiable from Σ_2 with respect to a simulator $\mathcal{S}$, then Σ_1 is as secure as Σ_2 for any single-stage game. More concretely, let* Game *be a single-stage game, then for any adversary $\mathcal{A}$, there is an adversary $\mathcal{B}$ and a distinguisher $\mathcal{D}$ such that*

$$\Pr[\mathsf{Game}_{\Pi^{\mathcal{F}_1},\mathcal{A}^{\mathcal{F}_1}}] \leq \Pr[\mathsf{Game}_{\mathcal{F}_2,\mathcal{B}^{\mathcal{F}_2}}] + \mathrm{Adv}^{\mathsf{indif}}_{\Sigma_1,\Sigma_2,\mathcal{S},\mathcal{D}}.$$

The proof of Theorem 2.1 is straightforward.

3 Impossibility of indifferentiable PKE in GGM

In this section, we show that under the definition of ideal PKE from [ZZ20], it is impossible to construct an indifferentiable PKE in the GGM.

We establish this impossibility via a black-box separation argument. Suppose, toward contradiction, that a construction $C^{\mathcal{G}}$ were indifferentiable from an ideal PKE. Then, by the definition of indifferentiability, any cryptographic primitive that exists relative to the ideal PKE would also exist relative to the GGM. Our proof strategy therefore proceeds in two steps: (1) we identify a cryptographic primitive that exists relative to the ideal PKE; and (2) we establish a black-box separation between this primitive and the GGM.

To identify an appropriate primitive, we first recall the formal definition of ideal PKE from [ZZ20].

Definition 3.1 (Ideal PKE [ZZ20]). *Let $\mathcal{SK}$, $\mathcal{PK}$, $\mathcal{M}$, $\mathcal{R}$, and $\mathcal{C}$ be the private-key space, public-key space, message space, randomness space, and ciphertext space, respectively.*

Define $\mathrm{Inj}_{\mathsf{KG}}$ as the set of all injections from $\mathcal{SK}$ to $\mathcal{PK}$, and $\mathrm{Inj}_{\mathsf{Enc}}$ as the set of all injections from $\mathcal{PK}\times\mathcal{M}\times\mathcal{R}$ to $\mathcal{C}$. The ideal public-key encryption is defined by sampling a function pair (h, f) uniformly at random from $\mathrm{Inj}_{\mathsf{KG}}\times\mathrm{Inj}_{\mathsf{Enc}}$, and exposing the following interfaces:

- KG *takes as input* $\mathsf{sk} \in \mathcal{SK}$ *and returns* $h(\mathsf{sk})$;
- Enc *takes as input* $\mathsf{pk} \in \mathcal{PK}$, $\mathsf{m} \in \mathcal{M}$, $\mathsf{r} \in \mathcal{R}$, *and returns* $f(\mathsf{pk}, \mathsf{m}, \mathsf{r})$.
- Dec *takes as input* $\mathsf{sk} \in \mathcal{SK}$ *and* $\mathsf{c} \in \mathcal{C}$, *and does as follows:*
  - *if there is* $(\mathsf{m}, \mathsf{r}) \in \mathcal{M} \times \mathcal{R}$ *such that* $f(h(\mathsf{sk}), \mathsf{m}, \mathsf{r}) = \mathsf{c}$, *return* m;
  - *otherwise, return* $\perp$.

The appropriate primitive should exhibit a property that is implied by the ideal PKE but is absent in the GGM. Observe that ideal PKE yields pseudorandom public keys and pseudorandom ciphertexts. In contrast, algorithms in the GGM can always distinguish valid group elements from uniformly random strings. We therefore identify the target primitive as MRA-secure PKE (cf. Definition 2.1) with pseudorandom public keys and ciphertexts, which we formally define as follows:

Definition 3.2 (Pseudorandom Public Key of PKE [ZZ20])**.** *The public-key encryption scheme*

$$\Pi_{\mathsf{PKE}} = (\Pi_{\mathsf{PKE}}.\mathsf{KG}, \Pi_{\mathsf{PKE}}.\mathsf{Enc}, \Pi_{\mathsf{PKE}}.\mathsf{Dec})$$

has pseudorandom public key *if there exists a negligible function* negl *such that for any* PPT *adversary* $\mathcal{A}$*, it holds that*

$$\left|\Pr[\mathcal{A}(\mathsf{pk}) = 1] - \Pr[\mathcal{A}(\mathsf{rpk}) = 1]\right| \leq \mathsf{negl}(\lambda),$$

where $\mathsf{pk} \leftarrow \Pi_{\mathsf{PKE}}.\mathsf{KG}(\mathsf{sk})$ *with* $\mathsf{sk} \xleftarrow{\$} \mathcal{SK}_\lambda$*, and* $\mathsf{rpk} \xleftarrow{\$} \mathcal{PK}_\lambda$*.*

Definition 3.3 (Pseudorandom Ciphertext of PKE [ZZ20])**.** *The public-key encryption scheme*

$$\Pi_{\mathsf{PKE}} = (\Pi_{\mathsf{PKE}}.\mathsf{KG}, \Pi_{\mathsf{PKE}}.\mathsf{Enc}, \Pi_{\mathsf{PKE}}.\mathsf{Dec})$$

has pseudorandom ciphertext *if there exists a negligible function* negl *such that for any* PPT *adversary* $\mathcal{A}$*, it holds that*

$$\left|\Pr[\mathcal{A}(\mathsf{pk}, \mathsf{c}) = 1] - \Pr[\mathcal{A}(\mathsf{pk}, \mathsf{c}_\mathsf{r}) = 1]\right| \leq \mathsf{negl}(\lambda).$$

Here, for randomly chosen $\mathsf{sk} \xleftarrow{\$} \mathcal{SK}_\lambda$*,* $\mathsf{m} \xleftarrow{\$} \mathcal{M}_\lambda$*,* $\mathsf{r} \xleftarrow{\$} \mathcal{R}_\lambda$*, and* $\mathsf{c}_\mathsf{r} \xleftarrow{\$} \mathcal{C}_\lambda$*, compute* $\mathsf{pk} \leftarrow \Pi_{\mathsf{PKE}}.\mathsf{KG}(\mathsf{sk})$ *and* $\mathsf{c} \leftarrow \Pi_{\mathsf{PKE}}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$*, respectively.*

Evidently, the ideal PKE implies a MRA-secure PKE with pseudorandom public keys and ciphertexts. Therefore, it suffices to establish the black-box separation in the GGM.

Theorem 3.1. *Let* $\mathcal{G}_{N,\ell}$ *be a generic group model satisfying* $\ell - \log N \geq \omega(\log \lambda)$*. Let*

$$\Pi_{\mathsf{PKE}}^{\mathcal{G}_{N,\ell}} = (\Pi_{\mathsf{PKE}}.\mathsf{KG}^{\mathcal{G}_{N,\ell}}, \Pi_{\mathsf{PKE}}.\mathsf{Enc}^{\mathcal{G}_{N,\ell}}, \Pi_{\mathsf{PKE}}.\mathsf{Dec}^{\mathcal{G}_{N,\ell}})$$

be a perfectly correct PKE with pseudorandom public keys and ciphertexts. Then there exists an adversary $\mathcal{A}$ *that makes polynomially many queries to* $\mathcal{G}_{N,\ell}$ *and breaks the MRA-secure game of* $\Pi_{\mathsf{PKE}}^{\mathcal{G}_{N,\ell}}$*.*

Proof Sketch. For simplicity, we assume that $\Pi_{\mathsf{PKE}}^{\mathcal{G}_{N,\ell}}$ achieves perfect correctness, and each of $\Pi_{\mathsf{PKE}}.\mathsf{KG}$, $\Pi_{\mathsf{PKE}}.\mathsf{Enc}$, and $\Pi_{\mathsf{PKE}}.\mathsf{Dec}$ makes at most q queries.

Consider the communication in which Alice sends her public key pk to Bob, and Bob encrypts a message under pk and sends the resulting ciphertext c back to Alice. We construct an adversary $\mathcal{A}$ that, given pk and c, recovers the corresponding message with non-negligible probability. The adversary initializes two empty sets $S_{\mathsf{que\text{-}res}}$ and S_m, and then iterates through $6q + 1$ rounds of the following phases:

- **Phase I**: Searching for a proper key-generation view for Alice. Specifically, the adversary identifies a set $\tilde{S}_\mathcal{A}$ of labeling query-response pairs (x, str) together with a private key $\widetilde{\mathsf{sk}}$ such that:

  1. $\tilde{S}_\mathcal{A}$ is consistent with $S_{\mathsf{que\text{-}res}}$; and
  2. this view causes $\widetilde{\mathsf{sk}}$ to generate the correct pk, i.e., $\mathsf{pk} = \Pi_{\mathsf{PKE}}.\mathsf{KG}^{S_{\mathsf{que\text{-}res}} \cup \tilde{S}_\mathcal{A}}(\widetilde{\mathsf{sk}})$.

  The adversary then decrypts the ciphertext in this simulated view by computing

  $$\widetilde{\mathsf{m}} \leftarrow \Pi_{\mathsf{PKE}}.\mathsf{Dec}^{S_{\mathsf{que\text{-}res}} \cup \tilde{S}_\mathcal{A}}(\widetilde{\mathsf{sk}}, \mathsf{c}),$$

  and inserts $\widetilde{\mathsf{m}}$ into the set S_m.

- **Phase II**: Updating $S_{\mathsf{que\text{-}res}}$ using the real oracle. For each pair $(x, \mathsf{str}) \in \tilde{S}_\mathcal{A} \setminus S_{\mathsf{que\text{-}res}}$, the adversary queries the real oracle to obtain $(x, \mathcal{G}_{N,\ell}^{\mathsf{label}}(x))$, replaces the entry accordingly, and adds the resulting pair to $S_{\mathsf{que\text{-}res}}$.

Finally, $\mathcal{A}$ outputs the majority value among the messages recorded in S_{m}.

To analyze the winning probability of $\mathcal{A}$, let S_{Bob} denote the set of all query/response pairs (x, str) such that str appears during the real execution of the encryption algorithm by Bob. For any iteration, there are three cases:

- Case 1: There exists $(x, \mathsf{str}_1) \in \tilde{S}_{\mathcal{A}}$ and $(x, \mathsf{str}_2) \in S_{\mathsf{Bob}}$ such that $\mathsf{str}_1 \neq \mathsf{str}_2$.
- Case 2: There exists $(x_1, \mathsf{str}) \in \tilde{S}_{\mathcal{A}}$ and $(x_2, \mathsf{str}) \in S_{\mathsf{Bob}}$ such that $x_1 \neq x_2$.
- Case 3: For any $(x_1, \mathsf{str}_1) \in \tilde{S}_{\mathcal{A}}$ and $(x_2, \mathsf{str}_2) \in S_{\mathsf{Bob}}$ it holds that $x_1 = x_2$ if and only if $\mathsf{str}_1 = \mathsf{str}_2$.

Immediately note that Case 1 can occur in at most $3q$ iterations, since $|S_{\mathsf{Bob}}| \leq 3q$[10] and each update in Phase II eliminates at least one pair from S_{Bob}. For Case 3, there exists an alternative generic group $\tilde{\mathcal{G}}_{N,\ell}$ that remains consistent with both $\tilde{S}_{\mathcal{A}}$ and S_{Bob}. By perfect correctness, the message decrypted in such an iteration must therefore be the correct one.

It thus remains to bound the probability of Case 2. We next show that, by exploiting the pseudorandomness of public keys and ciphertexts, the probability of Case 2 is negligible.

Observe that Case 2 implies that the adversary has produced a valid group element str without making the corresponding query. Recall that $\mathcal{G}_{N,\ell}$ is defined as a random injection from $\mathbb{Z}_N$ to $\{0,1\}^\ell$, where $\ell - \log N > \omega(\log \lambda)$. This condition implies that, for any query-efficient algorithm $\mathcal{E}^{\mathcal{G}_{N,\ell}}$ receiving only the security parameter as input, the probability of outputting a valid group element that has not appeared in any previous query (that is, a "new" group element) is negligible. Therefore, it suffices to show the following reduction: if the adversary $\mathcal{A}$, given pk and c, can produce a new group element with non-negligible probability, then one can construct an extractor $\mathcal{E}$ that, given only the security parameter, uses $\mathcal{A}$ as a subroutine and outputs a new group element with non-negligible probability.

To ensure that any group element that is new to $\mathcal{A}$ is also new to $\mathcal{E}$, we employ the following extractor $\mathcal{E}$:

- Step 1: $\mathcal{E}$ samples a public key pk and a ciphertext c uniformly from the public-key space and ciphertext space, respectively, and runs $\mathcal{A}(\mathsf{pk}, \mathsf{c})$.
- Step 2: Whenever $\mathcal{A}$ makes an additional query whose input contains a new group element str, $\mathcal{E}$ outputs str.

A potential gap is that pk and/or c sampled uniformly at random may be invalid with overwhelming probability, whereas $\mathcal{A}$ might generate a new group element only when both pk and c are valid. Fortunately, pseudorandom public keys and pseudorandom ciphertexts bridge this gap. Specifically, if $\mathcal{A}$ can generate a new group element with non-negligible probability when given a valid public key and ciphertext, then essentially the same behavior must persist when pk and c are sampled uniformly at random. In that case, the new group element can still be extracted by $\mathcal{E}$.

Many technical details are omitted in this proof sketch; a full and rigorous proof is provided in Appendix A.

4 Redefining ideal PKE

As shown in Section 3, the indifferentiable PKE of [ZZ20] are inherently separated from the cryptographic groups because they *implicitly* require that public keys and ciphertexts be pseudorandom strings. In group-based cryptography, however, encodings of public keys and ciphertext components are *recognizable* by efficient predicates (e.g., point/encoding validation). An idealized model that refuses to acknowledge this recognizability creates artificial gaps and trivial differentiator.

Our remedy is simple and faithful to practice: we keep the algebraic black-box flavor of [ZZ20] but make the ideal model *recognizability-aware*. Concretely, the ideal PKE explicitly tests the validity of inputs that,

[10] $\Pi_{\mathsf{PKE}}.\mathsf{Enc}$ makes at most q queries, and each query involves at most three group elements.

in real schemes, are subject to syntactic/algebraic checks. This restores alignment with deployed systems while retaining clean simulators for our indifferentiability proofs.

Why validity testing belongs in the ideal PKE? We highlight two reasons:

1. *Faithfulness.* Validity testing is an explicit requirement in standardized group-based cryptosystems. For example, NIST SP 800-56A requires public-key validation for elliptic-curve schemes, while RFC 9180 requires validation of the group element contained in a P-256 HPKE ciphertext before use. Omitting this would let the environment exploit behavior that honest implementations never exhibit.
2. *Eliminating a trivial differentiator.* Without validation, an environment can feed arbitrary bit strings to ideal interfaces and force the simulator to produce "as-if-random" outputs where the real system must reject.

Definition 4.1 (Ideal PKE with Validity Testing)**.** *Let $\mathcal{SK}$, $\mathcal{PK}$, $\mathcal{M}$, $\mathcal{R}$, and $\mathcal{C}$ be the private-key space, public-key space, message space, randomness space, and ciphertext space, respectively. Define $\mathrm{Inj}_{\mathsf{KG}}$ as the set of all injections from $\mathcal{SK}$ to $\mathcal{PK}$, and $\mathrm{Inj}_{\mathsf{Enc}}$ as the set of all injections from $\mathcal{PK} \times \mathcal{M} \times \mathcal{R}$ to $\mathcal{C}$. The ideal public-key encryption* $\mathbf{PKE}$ *is defined by sampling (h, f) uniformly at random from $\mathrm{Inj}_{\mathsf{KG}} \times \mathrm{Inj}_{\mathsf{Enc}}$, and exposing the following interfaces:*

- $\mathbf{PKE}.\mathsf{KG}$ *takes as input* $\mathsf{sk} \in \mathcal{SK}$ *and returns* $h(\mathsf{sk})$*;*
- $\mathbf{PKE}.\mathsf{Enc}$ *takes as input* $\mathsf{pk} \in \mathcal{PK}$, $\mathsf{m} \in \mathcal{M}$, $\mathsf{r} \in \mathcal{R}$*, and does as follows:*
  - *if there is no* $\mathsf{sk} \in \mathcal{SK}$ *such that* $h(\mathsf{sk}) = \mathsf{pk}$*, return* $\perp_{\mathsf{pk}}$*;*
  - *otherwise, return* $f(\mathsf{pk}, \mathsf{m}, \mathsf{r})$.
- $\mathbf{PKE}.\mathsf{Dec}$ *takes as input* $\mathsf{sk} \in \mathcal{SK}$ *and* $\mathsf{c} \in \mathcal{C}$*, and does as follows:*
  - *if there is no* $(\mathsf{sk}', \mathsf{m}', \mathsf{r}') \in \mathcal{SK} \times \mathcal{M} \times \mathcal{R}$ *such that* $f\big(h(\mathsf{sk}'), \mathsf{m}', \mathsf{r}'\big) = \mathsf{c}$*, return* $\perp_{\mathsf{c}}$*;*
  - *if there is* $(\mathsf{m}, \mathsf{r}) \in \mathcal{M} \times \mathcal{R}$ *such that* $f(h(\mathsf{sk}), \mathsf{m}, \mathsf{r}) = \mathsf{c}$*, return* m*;*
  - *otherwise, return* $\perp_{\mathsf{m}}$.

Rationale and example. In real ECIES encryption scheme, a party that receives pk first executes a public-key validation procedure (e.g., curve/encoding checks) before performing any group operation. If validation fails, the session aborts. The $\mathbf{PKE}.\mathsf{Enc}$ interface expresses this by returning $\perp_{\mathsf{pk}}$ instead of a ciphertext. In contrast, if this check were omitted and a ciphertext c was always returned, a trivial one-query differentiator would emerge: submit a random string str (not a valid encoding) to both worlds—while the real system aborts, the ideal world (without validation) would nevertheless return a ciphertext.

Why three distinct error symbols? We use three distinguished symbols: $\perp_{\mathsf{pk}}$ for invalid public keys, $\perp_{\mathsf{c}}$ for invalid ciphertexts, and $\perp_{\mathsf{m}}$ for decryption failures on *valid* ciphertexts. We deliberately avoid overloading a single "$\perp$" because different failure causes are *observable* in practice (e.g., through timing, logging, or API branching). As emphasized in [KL20, page 157], "... it is extraordinarily hard to ensure that the different errors cannot be distinguished, since, e.g., even a difference in the *time* to return each of these errors may allow an adversary to distinguish between them...." Therefore, distinguishing these causes in the ideal does not empower the environment beyond what the real world already reveals; it merely removes an information mismatch.

5 Making PSEC indifferentiable

In this section, we show that the PSEC scheme standardized in ISO/IEC 18033-2, under slight augmentation, achieves indifferentiability with respect to our revised ideal PKE. Our approach follows the KEM/DEM paradigm and proceeds in three steps:

- Introduce an idealized model for key encapsulation, termed the *ideal key-encapsulation mechanism* (ideal KEM).
- Show that a slightly augmented PSEC-KEM is indifferentiable from the ideal KEM.
- Prove that the composition of the ideal KEM and the ideal AE is indifferentiable from our revised ideal PKE.

5.1 Ideal KEM with validity testing

In this section, we introduce an idealized KEM model, defined as follows:

Definition 5.1 (Ideal KEM with Validity Testing)**.** *Let $\mathcal{SK}$, $\mathcal{PK}$, $\mathcal{R}$, $\mathcal{C}$, and $\mathcal{K}$ be the private-key space, public-key space, randomness space, ciphertext space, and key space, respectively. Define* Inj *as the set of all injections from $\mathcal{SK}$ to $\mathcal{PK}$, and* Func *as the set of all functions from $\mathcal{PK} \times \mathcal{R}$ to $\mathcal{C} \times \mathcal{K}$. Let $\mathcal{S} \subseteq \mathrm{Inj} \times \mathrm{Func}$ be the set of all pairs (h, f), satisfying:*

- ***Partial Injectivity:*** *For all $(\mathsf{pk}, \mathsf{r}), (\mathsf{pk}', \mathsf{r}') \in \mathcal{PK} \times \mathcal{R}$, let $(\mathsf{c}, \mathsf{k}) \leftarrow f(\mathsf{pk}, \mathsf{r})$ and $(\mathsf{c}', \mathsf{k}') \leftarrow f(\mathsf{pk}', \mathsf{r}')$. Then $(\mathsf{pk}, \mathsf{r}) = (\mathsf{pk}', \mathsf{r}') \iff \mathsf{c} = \mathsf{c}'$.*

The ideal key-encapsulation mechanism $\mathbf{KEM}$ *is defined by sampling (h, f) uniformly at random from $\mathcal{S}$, and exposing the following interfaces:*

- $\mathbf{KEM}.\mathsf{KG}$ *takes as input $\mathsf{sk} \in \mathcal{SK}$ and returns $h(\mathsf{sk})$;*
- $\mathbf{KEM}.\mathsf{Encaps}$ *takes as input $\mathsf{pk} \in \mathcal{PK}$, $\mathsf{r} \in \mathcal{R}$, and does as follows:*
  - *if there is no $\mathsf{sk} \in \mathcal{SK}$ such that $h(\mathsf{sk}) = \mathsf{pk}$, return $\perp_{\mathsf{pk}}$;*
  - *otherwise, return $f(\mathsf{pk}, \mathsf{r})$.*
- $\mathbf{KEM}.\mathsf{Decaps}$ *takes as input sk and $\mathsf{c} \in \mathcal{C}$, and does as follows:*
  - *if there is no $(\widetilde{\mathsf{sk}}, \tilde{\mathsf{r}}) \in \mathcal{SK} \times \mathcal{R}$ such that $f\left(h(\widetilde{\mathsf{sk}}), \tilde{\mathsf{r}}\right) = (\mathsf{c}, *)$, return $\perp_{\mathsf{c}}$;*
  - *if there is $\mathsf{r} \in \mathcal{R}$ such that $f\left(h(\mathsf{sk}), \mathsf{r}\right) = (\mathsf{c}, \mathsf{k})$, return k;*
  - *otherwise, return $\perp$.*

5.2 Indifferentiable PSEC-KEM

In this section, we show that a slightly augmented PSEC-KEM achieves indifferentiability with respect to the ideal KEM. Recall the PSEC-KEM scheme, defined over a cyclic group $\mathbb{G}$ of prime order N with generator g, and equipped with three hash functions $\mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3$. The construction is specified as follows:

$$\mathsf{PSEC.KG}(\mathsf{sk}) := g^{\mathsf{sk}}, \mathsf{PSEC.Encaps}(\mathsf{pk}, \mathsf{seed}) := (\mathsf{c}_1, \mathsf{c}_2, \mathsf{k}),$$

where $\mathsf{c}_1 := g^{\mathsf{H}_1(\mathsf{seed})}$, $\mathsf{c}_2 := \mathsf{seed} \oplus \mathsf{H}_2(g^{\mathsf{H}_1(\mathsf{seed})} || \mathsf{pk}^{\mathsf{H}_1(\mathsf{seed})})$, $\mathsf{k} := \mathsf{H}_3(\mathsf{seed})$. The decapsulation algorithm $\mathsf{PSEC.Decaps}$ first recovers seed from sk and $(\mathsf{c}_1, \mathsf{c}_2)$, and then outputs the shared key $\mathsf{k} := \mathsf{H}_3(\mathsf{seed})$.

Similarly, we lift the underlying group and hash functions into the GGM and the random oracle models. Specifically, our construction is defined over the private-key space $\mathcal{SK} := \mathbb{Z}_N$, the public-key space $\mathcal{PK} := \{0,1\}^{\ell}$, the randomness space $\mathcal{R} := \{0,1\}^{\lambda}$, the ciphertext space $\mathcal{C} := \{0,1\}^{\ell+\lambda}$, and the key space $\mathcal{K}$. Here N is a λ-bit prime, and ℓ is a function of the security parameter $\lambda \in \mathbb{N}$ with $\ell(\lambda) \geq \lambda + 1$. The construction relies on the following building blocks:

- $\mathcal{H}_x$ and $\mathcal{H}_y$: two random oracles mapping from $\{0,1\}^*$ to $\mathbb{Z}_N$;

- $\mathcal{H}_{\mathsf{seed}}$ and $\mathcal{H}_{\mathsf{mask}}$: two random oracles mapping from $\{0,1\}^*$ to $\{0,1\}^\lambda$;
- $\mathcal{H}_{\mathsf{key}}$: a random oracle mapping from $\{0,1\}^*$ to $\mathcal{K}$;
- $\mathcal{P}$: a random permutation defined over $\mathcal{C}$;
- $\mathcal{G} = (\mathcal{G}^{\mathsf{label}}, \mathcal{G}^{\mathsf{add}})$: a generic group model that maps $\mathbb{Z}_N$ to $\mathcal{PK}$;

For convenience, we denote $\mathcal{O} := (\mathcal{H}_x, \mathcal{H}_{\mathsf{seed}}, \mathcal{H}_y, \mathcal{H}_{\mathsf{mask}}, \mathcal{H}_{\mathsf{key}}, \mathcal{P}, \mathcal{G})$ and design our construction $\Pi^{\mathcal{O}}_{\mathsf{KEM}}$ as follows.

Construction $\Pi^{\mathcal{O}}_{\mathsf{KEM}}$

$\underline{\Pi_{\mathsf{KEM}}.\mathsf{KG}^{\mathcal{O}}(\mathsf{sk})}$:
  $x \leftarrow \mathcal{H}_x(\mathsf{sk})$; $\mathsf{pk} \leftarrow \mathcal{G}^{\mathsf{label}}(x)$;
  return pk.
$\underline{\Pi_{\mathsf{KEM}}.\mathsf{Encaps}^{\mathcal{O}}(\mathsf{pk}, \mathsf{r})}$:
  if $\mathcal{G}^{\mathsf{add}}(\mathsf{pk}, \mathsf{pk}) = \perp$: **return** $\perp_{\mathsf{pk}}$;
  $\mathsf{seed} \leftarrow \mathcal{H}_{\mathsf{seed}}(\mathsf{pk}, \mathsf{r})$; $\mathsf{k} \leftarrow \mathcal{H}_{\mathsf{key}}(\mathsf{pk}||\mathsf{seed})$;
  $y \leftarrow \mathcal{H}_y(\mathsf{seed})$; $\tilde{\mathsf{c}}_1 \leftarrow \mathcal{G}^{\mathsf{label}}(y)$; $\mathsf{str} \leftarrow \mathsf{pk}^y$;
  $\mathsf{mask} \leftarrow \mathcal{H}_{\mathsf{mask}}(\tilde{\mathsf{c}}_1||\mathsf{str})$; $\tilde{\mathsf{c}}_2 \leftarrow \mathsf{seed} \oplus \mathsf{mask}$;
  $\mathsf{c} \leftarrow \mathcal{P}(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2)$;
  return (c, k).
$\underline{\Pi_{\mathsf{KEM}}.\mathsf{Decaps}^{\mathcal{O}}(\mathsf{sk}, \mathsf{c})}$:
  $\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2 \leftarrow \mathcal{P}^{-1}(\mathsf{c})$;
  if $\mathcal{G}^{\mathsf{add}}(\tilde{\mathsf{c}}_1, \tilde{\mathsf{c}}_1) = \perp$: **return** $\perp_{\mathsf{c}}$;
  $x \leftarrow \mathcal{H}_x(\mathsf{sk})$; $\mathsf{pk} \leftarrow \mathcal{G}^{\mathsf{label}}(x)$; $\mathsf{str} \leftarrow \tilde{\mathsf{c}}_1^x$;
  $\mathsf{mask} \leftarrow \mathcal{H}_{\mathsf{mask}}(\tilde{\mathsf{c}}_1||\mathsf{str})$; $\mathsf{seed} \leftarrow \tilde{\mathsf{c}}_2 \oplus \mathsf{mask}$;
  $y \leftarrow \mathcal{H}_y(\mathsf{seed})$; if $\mathcal{G}^{\mathsf{label}}(y) \neq \tilde{\mathsf{c}}_1$: **return** $\perp$;
  return $\mathcal{H}_{\mathsf{key}}(\mathsf{pk}||\mathsf{seed})$.

Correctness of our construction is immediate. We now establish indifferentiability in the following theorem.

Theorem 5.1. *$\Pi^{\mathcal{O}}_{\mathsf{KEM}}$ is indifferentiable from the ideal KEM. More precisely, there exists a simulator $\mathcal{S}$ such that for all q-query differentiator $\mathcal{D}$, we have*

$$\mathrm{Adv}^{\mathsf{indiff}}_{\Pi^{\mathcal{O}}_{\mathsf{KEM}}, \mathbf{KEM}, \mathcal{S}}(\mathcal{D}) \leq O\left(\frac{(\lambda \cdot q)^4}{N} + \frac{(\lambda \cdot q)^3}{2^\lambda}\right).$$

Here, the simulator makes at most $O(\lambda \cdot q)$ queries to its oracles.

The full proof is deferred to Appendix B.

5.3 Composition of ideal KEM and ideal AE

In this section, we complete the construction of an indifferentiable PKE from the ideal KEM and the ideal AE, following the KEM/DEM paradigm. Specifically, our construction relies on the following components. For consistency with Section 5.2, we present them together with concrete instantiations of the corresponding spaces.

- Ideal AE $(\mathcal{AE}, \mathcal{AD})$ with shared-key space $\mathcal{K} := \{0,1\}^\lambda$, nonce space $\mathcal{N} := \{0,1\}^\lambda$, message space $\mathcal{M} := \{0,1\}^\lambda$, and ciphertext space $\mathcal{C}_\mathsf{m} := \{0,1\}^{2\lambda}$;
- Ideal KEM $\mathbf{KEM}$ with private-key space $\mathcal{SK} := \mathbb{Z}_N$, public-key space $\mathcal{PK} := \{0,1\}^\ell$, randomness space $\mathcal{R} := \{0,1\}^\lambda$, ciphertext space $\mathcal{C}_\mathsf{k} := \{0,1\}^{\ell+\lambda}$, and key space $\mathcal{K} \times \mathcal{N}$;
- Random oracle $\mathcal{H}_\mathsf{r} : \{0,1\}^* \to \mathcal{R}$;
- Random permutation $(\mathcal{P}, \mathcal{P}^{-1})$ defined over $\mathcal{C}_\mathsf{k} \times \mathcal{C}_\mathsf{m}$.

Here N is a λ-bit prime, and ℓ is a function of $\lambda \in \mathbb{N}$ with $\ell(\lambda) \geq \lambda + 1$.

We are now ready to construct our indifferentiable PKE. For convenience, we denote $\mathcal{O} = (\mathcal{AE}, \mathcal{AD}, \mathbf{KEM}, \mathcal{H}_\mathsf{r}, \mathcal{P}, \mathcal{P}^{-1})$. The resulting construction, denoted $\Pi^{\mathcal{O}}_{\mathsf{PKE}}$, is defined over the private-key space $\mathcal{SK}$, public-key space $\mathcal{PK}$, randomness space $\mathcal{R}$, message space $\mathcal{M}$, and ciphertext space $\mathcal{C} := \mathcal{C}_\mathsf{k} \times \mathcal{C}_\mathsf{m}$.

Construction $\Pi^{\mathcal{O}}_{\mathsf{PKE}}$

$\underline{\Pi_{\mathsf{PKE}}.\mathsf{KG}^{\mathcal{O}}(\mathsf{sk})}$:

return $\mathbf{KEM}.\mathsf{KG}(\mathsf{sk})$.

$\underline{\Pi_{\mathsf{PKE}}.\mathsf{Enc}^{\mathcal{O}}(\mathsf{pk}, \mathsf{m}, \mathsf{r})}$:

if $\mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r}) = \perp_{\mathsf{pk}}$: **return** $\perp_{\mathsf{pk}}$;

$\tilde{\mathsf{r}} \leftarrow \mathcal{H}_\mathsf{r}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$;

$(\mathsf{c}_\mathsf{k}, \mathsf{k}||\mathsf{n}) \leftarrow \mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk}, \tilde{\mathsf{r}})$, $\mathsf{c}_\mathsf{m} \leftarrow \mathcal{AE}(\mathsf{k}, \mathsf{n}, \mathsf{m})$;

return $\mathcal{P}(\mathsf{c}_\mathsf{k}||\mathsf{c}_\mathsf{m})$.

$\underline{\Pi_{\mathsf{PKE}}.\mathsf{Dec}^{\mathcal{O}}(\mathsf{sk}, \mathsf{c})}$:

$\mathsf{c}_\mathsf{k}||\mathsf{c}_\mathsf{m} \leftarrow \mathcal{P}^{-1}(\mathsf{c})$;

if $\mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c}_\mathsf{k}) = \perp_\mathsf{c}$: **return** $\perp_\mathsf{c}$;

if $\mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c}_\mathsf{k}) = \perp$: **return** $\perp_\mathsf{m}$;

$\mathsf{k}||\mathsf{n} \leftarrow \mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c}_\mathsf{k})$;

if $\mathcal{AD}(\mathsf{k}, \mathsf{n}, \mathsf{c}_\mathsf{m}) = \perp$, **return** $\perp_\mathsf{m}$;

return $\mathcal{AD}(\mathsf{k}, \mathsf{n}, \mathsf{c}_\mathsf{m})$.

Correctness of our construction is immediate. We now establish indifferentiability in the following theorem.

Theorem 5.2. *$\Pi^{\mathcal{O}}_{\mathsf{PKE}}$ is indifferentiable from the ideal PKE with validity testing. More precisely, there exists a simulator $\mathcal{S}$ such that for all q-query differentiator $\mathcal{D}$, we have*

$$\mathrm{Adv}^{\mathsf{indiff}}_{\Pi^{\mathcal{O}}_{\mathsf{PKE}}, \mathbf{PKE}, \mathcal{S}}(\mathcal{D}) \leq O\left(\frac{(\lambda \cdot q)^2}{N} + \frac{(\lambda \cdot q)^2}{2^\lambda}\right).$$

Here, the simulator makes at most $O(\lambda \cdot q)$ queries to its oracles.

The full proof is deferred to Appendix C.

6 Making ECIES indifferentiable

In this section, we show that the ECIES scheme standardized in ISO/IEC 18033-2, with slight augmentation, also achieves indifferentiability. Following the same strategy as in Section 5, we proceed in three steps:

- We introduce an ECIES-compatible variant of the ideal KEM model.
- We prove that a slightly augmented ECIES-KEM is indifferentiable from this ideal KEM.
- We show that its composition with ideal AE is indifferentiable from our revised ideal PKE.

6.1 ECIES-compatible ideal KEM

Definition 6.1 (ECIES-compatible Ideal KEM)**.** *Let $\mathcal{SK}$, $\mathcal{PK}$, $\mathcal{R}$, $\mathcal{C}$, and $\mathcal{K}$ be the private-key space, public-key space, randomness space, ciphertext space, and key space, respectively.*

Define Inj *as the set of all injections from $\mathcal{SK}$ to $\mathcal{PK}$, and* Func *as the set of all functions from $\mathcal{PK} \times \mathcal{R}$ to $\mathcal{C} \times \mathcal{K}$. Let $\mathcal{S} \subseteq \mathrm{Inj} \times \mathrm{Func}$ be the set of all pairs (h, f), satisfying:*

- ***Partial Injectivity:*** *For all $(\mathsf{pk}, \mathsf{r}), (\mathsf{pk}', \mathsf{r}') \in \mathcal{PK} \times \mathcal{R}$, let $(\mathsf{c}, \mathsf{k}) \leftarrow f(\mathsf{pk}, \mathsf{r})$ and $(\mathsf{c}', \mathsf{k}') \leftarrow f(\mathsf{pk}', \mathsf{r}')$. Then, $\mathsf{r} = \mathsf{r}' \iff \mathsf{c} = \mathsf{c}'$.*

The ECIES-compatible ideal key-encapsulation mechanism **IES-KEM** *is defined by sampling a function pair (h, f) uniformly at random from $\mathcal{S}$, and exposing the following interfaces:*

- **IES-KEM**.KG *takes as input $\mathsf{sk} \in \mathcal{SK}$ and returns $h(\mathsf{sk})$;*
- **IES-KEM**.Encaps *takes as input $\mathsf{pk} \in \mathcal{PK}$, $\mathsf{r} \in \mathcal{R}$, and does as follows:*
  - *if there is no $\mathsf{sk} \in \mathcal{SK}$ such that $h(\mathsf{sk}) = \mathsf{pk}$, return $\perp_{\mathsf{pk}}$;*
  - *otherwise, return $f(\mathsf{pk}, \mathsf{r})$.*
- **IES-KEM**.Decaps *takes as input sk and $\mathsf{c} \in \mathcal{C}$, and does as follows:*
  - *if there exists no $\mathsf{r} \in \mathcal{R}$ such that $f\big(h(\mathsf{sk}), \mathsf{r}\big) = (\mathsf{c}, *)$, return $\perp_{\mathsf{c}}$;*
  - *if there is $\mathsf{r} \in \mathcal{R}$ such that $f(h(\mathsf{sk}), \mathsf{r}) = (\mathsf{c}, \mathsf{k})$, return k.*

The only difference between these two idealized KEM models is that the latter enforces that the ciphertext is determined solely by the nonce. This modification is needed to faithfully capture the structural feature of ECIES. We next show that a slightly augmented ECIES-KEM is indifferentiable from this ECIES-compatible ideal KEM.

6.2 Making ECIES-KEM indifferentiable

Recall the ECIES-KEM scheme $\mathsf{IES} = (\mathsf{IES.KG}, \mathsf{IES.Encaps}, \mathsf{IES.Decaps})$ which is defined over a cyclic group $\mathbb{G}$ of prime order N with generator $g \in \mathbb{G}$, together with a hash function H. The algorithms are specified as follows:

$$\mathsf{IES.KG}(\mathsf{sk}) := g^{\mathsf{sk}};\ \mathsf{IES.Encaps}(\mathsf{pk}, \mathsf{r}) := (g^{\mathsf{r}}, \mathsf{H}(\mathsf{pk}^{\mathsf{r}}));\ \mathsf{IES.Decaps}(\mathsf{sk}, \mathsf{c}) := H(\mathsf{c}^{\mathsf{sk}}).$$

Similarly, we lift the underlying group and hash function into the GGM and the random oracle model. Specifically, our construction is defined over the private-key space $\mathcal{SK} := \mathbb{Z}_N$, the public-key space $\mathcal{PK} := \{0,1\}^{\ell}$, the randomness space $\mathcal{R} := \mathbb{Z}_N$, the ciphertext space $\mathcal{C} := \{0,1\}^{\ell}$, and the key space $\mathcal{K}$. Here N is a λ-bit prime, and ℓ is a function of the security parameter λ with $\ell(\lambda) \geq \lambda + 1$. The construction relies on the following building blocks:

- $\mathcal{H}_x$ and $\mathcal{H}_y$: two independent random oracles mapping from $\{0,1\}^*$ to $\mathbb{Z}_N$;

- $\mathcal{H}_{\mathsf{key}}$: a random oracle mapping from $\{0,1\}^*$ to $\mathcal{K}$;
- $\mathcal{P}$: a random permutation defined over $\{0,1\}^\ell$;
- $\mathcal{G} = (\mathcal{G}^{\mathsf{label}}, \mathcal{G}^{\mathsf{add}})$: a generic group model that maps $\mathbb{Z}_N$ to $\mathcal{PK}$;

For convenience, we denote $\mathcal{O} := (\mathcal{H}_x, \mathcal{H}_y, \mathcal{H}_{\mathsf{key}}, \mathcal{P}, \mathcal{G})$ and design our construction $\Pi^{\mathcal{O}}_{\mathsf{IES}}$ as follows:

Construction $\Pi^{\mathcal{O}}_{\mathsf{IES}}$

$\Pi_{\mathsf{IES}}.\mathsf{KG}^{\mathcal{O}}(\mathsf{sk})$:

$x \leftarrow \mathcal{H}_x(\mathsf{sk})$; $\mathsf{pk} \leftarrow \mathcal{G}^{\mathsf{label}}(x)$, **return** pk.

$\Pi_{\mathsf{IES}}.\mathsf{Encaps}^{\mathcal{O}}(\mathsf{pk}, \mathsf{r})$:

if $\mathcal{G}^{\mathsf{add}}(\mathsf{pk}, \mathsf{pk}) = \perp$: **return** $\perp_{\mathsf{pk}}$;

$y \leftarrow \mathcal{H}_y(\mathsf{r})$; $\tilde{\mathsf{c}} \leftarrow \mathcal{G}^{\mathsf{label}}(y)$; $\mathsf{str} \leftarrow \mathsf{pk}^y$;

$\mathsf{c} \leftarrow \mathcal{P}(\tilde{\mathsf{c}})$; $\mathsf{k} \leftarrow \mathcal{H}_{\mathsf{key}}(\mathsf{pk}, \tilde{\mathsf{c}}, \mathsf{str})$;

return (c, k).

$\Pi_{\mathsf{IES}}.\mathsf{Decaps}^{\mathcal{O}}(\mathsf{sk}, \mathsf{c})$:

$\tilde{\mathsf{c}} \leftarrow \mathcal{P}^{-1}(\mathsf{c})$;

if $\mathcal{G}^{\mathsf{add}}(\tilde{\mathsf{c}}, \tilde{\mathsf{c}}) = \perp$: **return** $\perp_{\mathsf{c}}$;

$x \leftarrow \mathcal{H}_x(\mathsf{sk})$; $\mathsf{pk} \leftarrow \mathcal{G}^{\mathsf{label}}(x)$; $\mathsf{str} \leftarrow \tilde{\mathsf{c}}^x$;

return $\mathcal{H}_{\mathsf{key}}(\mathsf{pk}, \tilde{\mathsf{c}}, \mathsf{str})$.

Correctness of our construction is immediate. We now establish indifferentiability in the following theorem.

Theorem 6.1. *$\Pi^{\mathcal{O}}_{\mathsf{IES}}$ is indifferentiable from the ECIES-compatible ideal KEM* **IES-KEM**. *More precisely, there exists a simulator $\mathcal{S}$ such that for all q-query differentiator $\mathcal{D}$, we have*

$$\mathrm{Adv}^{\mathsf{indiff}}_{\Pi^{\mathcal{O}}_{\mathsf{IES}}, \mathbf{IES\text{-}KEM}, \mathcal{S}}(\mathcal{D}) \leq O\left(\frac{(\lambda \cdot q)^4}{N} + \frac{(\lambda \cdot q)^2}{2^\lambda}\right).$$

Here, the simulator makes at most $O(\lambda \cdot q)$ queries to its oracles.

The full proof is deferred to Appendix D.

6.3 Composition of ECIES-compatible ideal KEM and ideal AE

In this section, we complete the construction of an indifferentiable PKE from the ECIES-compatible ideal KEM and the ideal AE, following the KEM/DEM paradigm.

Specifically, we take the KEM to be the ECIES-compatible ideal KEM and the DEM to be the ideal AE. Our construction is built upon the following components. For consistency with Section 6.2, we present them along with concrete instantiations of the associated spaces:

- $(\mathcal{AE}, \mathcal{AD})$: an ideal AE with shared-key space $\mathcal{K} := \{0,1\}^\lambda$, nonce space $\mathcal{N} := \{0,1\}^\lambda$, message space $\mathcal{M} := \{0,1\}^\lambda$, and ciphertext space $\mathcal{C}_{\mathsf{m}} := \{0,1\}^{2\lambda}$.

- **IES-KEM**: an ECIES-compatible ideal KEM with private-key space $\mathcal{SK} := \mathbb{Z}_N$, public-key space $\mathcal{PK} := \{0,1\}^\ell$, randomness space $\mathcal{R} := \mathbb{Z}_N$, ciphertext space $\mathcal{C}_\mathsf{k} := \{0,1\}^\ell$, and key space $\mathcal{K} \times \mathcal{N}$.
- $\mathcal{H}$: a random oracle mapping from $\{0,1\}^*$ to $\mathcal{R}$;
- $(\mathcal{P}, \mathcal{P}^{-1})$: a random permutation defined over $\mathcal{C}_\mathsf{k} \times \mathcal{C}_\mathsf{m}$.

Here N is a λ-bit prime, and ℓ is a function of the security parameter λ with $\ell(\lambda) \geq \lambda + 1$.

We are now ready to construct our indifferentiable PKE. For convenience, we denote

$$\mathcal{O} = (\mathcal{AE}, \mathcal{AD}, \mathbf{IES\text{-}KEM}, \mathcal{H}, \mathcal{P}, \mathcal{P}^{-1}).$$

The resulting construction, denoted $\Pi^{\mathcal{O}}_{\mathsf{IES\text{-}PKE}}$, is defined over the private-key space $\mathcal{SK}$, public-key space $\mathcal{PK}$, randomness space $\mathcal{R}$, message space $\mathcal{M}$, and ciphertext space $\mathcal{C} := \mathcal{C}_\mathsf{k} \times \mathcal{C}_\mathsf{m}$.

Construction $\Pi^{\mathcal{O}}_{\mathsf{PKE}}$

$\underline{\Pi_{\mathsf{PKE}}.\mathsf{KG}^{\mathcal{O}}(\mathsf{sk})}$:

return $\mathbf{IES\text{-}KEM}.\mathsf{KG}(\mathsf{sk})$.

$\underline{\Pi_{\mathsf{PKE}}.\mathsf{Enc}^{\mathcal{O}}(\mathsf{pk}, \mathsf{m}, \mathsf{r})}$:

if $\mathbf{IES\text{-}KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r}) = \perp_\mathsf{pk}$: **return** $\perp_\mathsf{pk}$;

$\tilde{\mathsf{r}} \leftarrow \mathcal{H}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$;

$(\mathsf{c}_\mathsf{k}, \mathsf{k}||\mathsf{n}) \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{Encaps}(\mathsf{pk}, \tilde{\mathsf{r}})$;

$\mathsf{c}_\mathsf{m} \leftarrow \mathcal{AE}(\mathsf{k}, \mathsf{n}, \mathsf{m})$;

return $\mathcal{P}(\mathsf{c}_\mathsf{k}||\mathsf{c}_\mathsf{m})$.

$\underline{\Pi_{\mathsf{PKE}}.\mathsf{Dec}^{\mathcal{O}}(\mathsf{sk}, \mathsf{c})}$:

$\mathsf{c}_\mathsf{k}||\mathsf{c}_\mathsf{m} \leftarrow \mathcal{P}^{-1}(\mathsf{c})$;

if $\mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c}_\mathsf{k}) = \perp_\mathsf{c}$: **return** $\perp_\mathsf{c}$;

$\mathsf{k}||\mathsf{n} \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c}_\mathsf{k})$;

if $\mathcal{AD}(\mathsf{k}, \mathsf{n}, \mathsf{c}_\mathsf{m}) = \perp$, **return** $\perp$;

return $\mathcal{AD}(\mathsf{k}, \mathsf{n}, \mathsf{c}_\mathsf{m})$.

Correctness of our construction is immediate. We now establish indifferentiability in the following theorem.

Theorem 6.2. *$\Pi^{\mathcal{O}}_{\mathsf{IES\text{-}PKE}}$ is indifferentiable from the ideal PKE with validity testing $\mathbf{PKE}$. More precisely, there exists a simulator $\mathcal{S}$ such that for all q-query differentiator $\mathcal{D}$, we have*

$$\mathrm{Adv}^{\mathsf{indiff}}_{\Pi^{\mathcal{O}}_{\mathsf{IES\text{-}PKE}}, \mathbf{PKE}, \mathcal{S}}(\mathcal{D}) \leq O\left(\frac{(\lambda \cdot q)^2}{N} + \frac{(\lambda \cdot q)^2}{2^\lambda}\right).$$

Here, the simulator makes at most $O(\lambda \cdot q)$ queries to its oracles.

The full proof is deferred to Appendix E.

Acknowledgement. Cong Zhang was supported by the National Key Research and Development Program of China (Grant No. 2023YFB3106000). This work was supported by Ant Group through CCF-Ant Research Fund (Grant No. CCF-AFSG RF20230308). Hong-Sheng Zhou's work was partially supported by the VCU Quest Fund.

References

[ABD+13] Elena Andreeva, Andrey Bogdanov, Yevgeniy Dodis, Bart Mennink, and John P. Steinberger. On the indifferentiability of key-alternating ciphers. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 531–550. Springer, Berlin, Heidelberg, August 2013.

[BDPV08] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the indifferentiability of the Sponge construction. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 181–197. Springer, Berlin, Heidelberg, April 2008.

[BF18] Manuel Barbosa and Pooya Farshim. Indifferentiable authenticated encryption. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 187–220. Springer, Cham, August 2018.

[BHKL13] Daniel J. Bernstein, Mike Hamburg, Anna Krasnova, and Tanja Lange. Elligator: elliptic-curve points indistinguishable from uniform random strings. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 967–980. ACM Press, November 2013.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Berlin, Heidelberg, May / June 2006.

[BRS03] John Black, Phillip Rogaway, and Thomas Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In Kaisa Nyberg and Howard M. Heys, editors, *SAC 2002*, volume 2595 of *LNCS*, pages 62–75. Springer, Berlin, Heidelberg, August 2003.

[CDMP05] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård revisited: How to construct a hash function. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 430–448. Springer, Berlin, Heidelberg, August 2005.

[CDMS10] Jean-Sébastien Coron, Yevgeniy Dodis, Avradip Mandal, and Yannick Seurin. A domain extender for the ideal cipher. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 273–289. Springer, Berlin, Heidelberg, February 2010.

[CHK+16] Jean-Sébastien Coron, Thomas Holenstein, Robin Künzler, Jacques Patarin, Yannick Seurin, and Stefano Tessaro. How to build an ideal cipher: The indifferentiability of the Feistel construction. *Journal of Cryptology*, 29(1):61–114, January 2016.

[DRS09] Yevgeniy Dodis, Thomas Ristenpart, and Thomas Shrimpton. Salvaging Merkle-Damgård for practical applications. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 371–388. Springer, Berlin, Heidelberg, April 2009.

[DSSL16] Yevgeniy Dodis, Martijn Stam, John P. Steinberger, and Tianren Liu. Indifferentiability of confusion-diffusion networks. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 679–704. Springer, Berlin, Heidelberg, May 2016.

[GWL23] Chun Guo, Lei Wang, and Dongdai Lin. Impossibility of indifferentiable iterated blockciphers from 3 or less primitive calls. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part IV*, volume 14007 of *LNCS*, pages 408–439. Springer, Cham, April 2023.

[HKT11] Thomas Holenstein, Robin Künzler, and Stefano Tessaro. The equivalence of the random oracle model and the ideal cipher model, revisited. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 89–98. ACM Press, June 2011.

[IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *21st ACM STOC*, pages 44–61. ACM Press, May 1989.

[JZW+24] Keyu Ji, Cong Zhang, Taiyu Wang, Bingsheng Zhang, Hong-Sheng Zhou, Xin Wang, and Kui Ren. On the complexity of cryptographic groups and generic group models. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part VII*, volume 15490 of *LNCS*, pages 3–35. Springer, Singapore, December 2024.

[KL20] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. CRC Press, 3rd edition, 2020.

[MRH04] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 21–39. Springer, Berlin, Heidelberg, February 2004.

[Nat24] National Institute of Standards and Technology. Module-Lattice-Based Key-Encapsulation Mechanism Standard. NIST FIPS 203, Department of Commerce, August 2024.

[RS08] Phillip Rogaway and John P. Steinberger. Constructing cryptographic hash functions from fixed-key blockciphers. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 433–450. Springer, Berlin, Heidelberg, August 2008.

[RSS11] Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with composition: Limitations of the indifferentiability framework. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 487–506. Springer, Berlin, Heidelberg, May 2011.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Berlin, Heidelberg, May 1997.

[Sho04] Victor Shoup. Information technology-security techniques-encryption algorithms-part 2: Asymmetric ciphers. *ISO/IEC 18033-2*, 2004.

[The23] The Tor Project. Collector: Bridge Pool Assignments. https://collector.torproject.org/archive/bridge-pool-assignments/, 2023.

[Tib14] Mehdi Tibouchi. Elligator squared: Uniform points on elliptic curves of prime order as uniform random strings. In Nicolas Christin and Reihaneh Safavi-Naini, editors, *FC 2014*, volume 8437 of *LNCS*, pages 139–156. Springer, Berlin, Heidelberg, March 2014.

[Wee12] Hoeteck Wee. Public key encryption against related key attacks. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 262–279. Springer, Berlin, Heidelberg, May 2012.

[WZZ+25] Taiyu Wang, Cong Zhang, Hong-Sheng Zhou, Xin Wang, Pengfei Chen, Wenli Wang, Kui Ren, and Chun Chen. Attention is still what you need: Another round of exploring shoup's GGM. In Goichiro Hanaoka and Bo-Yin Yang, editors, *ASIACRYPT 2025, Part IV*, volume 16248 of *LNCS*, pages 239–271. Springer, Singapore, December 2025.

[Zha22] Mark Zhandry. To label, or not to label (in generic groups). In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022*, pages 66–96, Cham, 2022. Springer Nature Switzerland.

[ZZ20] Mark Zhandry and Cong Zhang. Indifferentiability for public key cryptosystems. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 63–93. Springer, Cham, August 2020.

[ZZ23] Cong Zhang and Mark Zhandry. The relationship between idealized models under computationally bounded adversaries. In *ASIACRYPT 2023*, 2023.

A Proof of Theorem 3.1

Theorem 3.1. *Let $\mathcal{G}_{N,\ell}$ be a generic group model satisfying $\ell - \log N \geq \omega(\log \lambda)$. Let $\Pi_{\mathsf{PKE}}^{\mathcal{G}_{N,\ell}}$ be a perfectly correct PKE with pseudorandom public keys and ciphertexts. Then there exists an adversary $\mathcal{A}$ that makes polynomially many queries to $\mathcal{G}_{N,\ell}$ and breaks the MRA-secure game of $\Pi_{\mathsf{PKE}}^{\mathcal{G}_{N,\ell}}$.*

Proof. For simplicity, we redenote the GGM as $\mathcal{G}$ in the following analysis. To establish the proof, we present a formal description of the adversary $\mathcal{A}$, as shown in Figure 2. Here, we specify that any algorithm in $\Pi_{\mathsf{PKE}}^{\mathcal{G}}$ makes at most q queries to $\mathcal{G}$, where q is a polynomial.

Adversary $\mathcal{A}$

$\mathcal{A}^{\mathcal{G}}(\mathsf{pk}, \mathsf{c})$:

$S_{\mathsf{m}} \leftarrow \emptyset$; $S_{\mathsf{que\text{-}res}} \leftarrow \emptyset$;

For $i = 1$ to $6q + 1$:

[Phase I: Searching a proper view of key generation]

Search a secret key $\widetilde{\mathsf{sk}}$ and a set of labeling query-response tuples $\tilde{S}_{\mathcal{A}}$, subject to the following requirements:

1. $\tilde{S}_{\mathcal{A}}$ is consistent with $S_{\mathsf{que\text{-}res}}$ and $|\tilde{S}_{\mathcal{A}}| \leq 6q$;
2. $\mathsf{pk} \leftarrow \Pi_{\mathsf{PKE}}.\mathsf{KG}^{S_{\mathsf{que\text{-}res}} \cup \tilde{S}_{\mathcal{A}}}(\widetilde{\mathsf{sk}})$;
3. When running the algorithm $\widetilde{\mathsf{m}}_i \leftarrow \Pi_{\mathsf{PKE}}.\mathsf{Dec}^{S_{\mathsf{que\text{-}res}} \cup \tilde{S}_{\mathcal{A}}}(\widetilde{\mathsf{sk}}, \mathsf{c})$, $S_{\mathsf{que\text{-}res}}$ is sufficient such that:
   - for a labeling query $\mathsf{que} = x$, there must be (x, res) in $S_{\mathsf{que\text{-}res}} \cup \tilde{S}_{\mathcal{A}}$ and responds with res;
   - for an addition query $\mathsf{que} = (\mathsf{str}_1, \mathsf{str}_2)$:
     - if either str_1 or str_2 is not covered in $S_{\mathsf{que\text{-}res}} \cup \tilde{S}_{\mathcal{A}}$, then responds with $\perp$;
     - otherwise, there must be (x_1, str_1), (x_2, str_2) and $(x_1 + x_2, \mathsf{str}_3)$ in $S_{\mathsf{que\text{-}res}} \cup \tilde{S}_{\mathcal{A}}$, and responds with str_3.

If there is no such $\widetilde{\mathsf{sk}}$ and $\tilde{S}_{\mathcal{A}}$: return $\perp$;

Else: $S_{\mathsf{m}} \leftarrow S_{\mathsf{m}} \cup \{\widetilde{\mathsf{m}}_i\}$;

[Phase II: Updating the guessing queries with real oracle]

For each $(x, \mathsf{str}) \in \tilde{S}_{\mathcal{A}} \setminus S_{\mathsf{que\text{-}res}}$: $S_{\mathsf{que\text{-}res}} \leftarrow S_{\mathsf{que\text{-}res}} \cup (x, \mathcal{G}^{\mathsf{label}}(x))$;

Return the most frequent value in S_{m}.

Figure 2: The description of $\mathcal{A}$ that breaks the MRA-secure game of Π_{PKE} in $\mathcal{G}$.

It is immediate that the adversary $\mathcal{A}$ is query-efficient. We now show that $\mathcal{A}$ can guess the correct message with noticeable probability. From the description of $\mathcal{A}$ in Figure 2, we observe that $\mathcal{A}$ aborts if it fails to obtain a proper view during Phase I of some iteration. We denote by $\mathsf{FindView}$ the event that $\mathcal{A}$ successfully identifies a proper view in every iteration. Since in the MRA-security game the adversary is given $\Pi_{\mathsf{PKE}}.\mathsf{KG}^{\mathcal{G}}(\mathsf{sk})$ and $\Pi_{\mathsf{PKE}}.\mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$, a proper view $(\widetilde{\mathsf{sk}}, \tilde{S}_{\mathcal{A}})$ must always exist, i.e., $\widetilde{\mathsf{sk}} = \mathsf{sk}$, and $\tilde{S}_{\mathcal{A}}$ is consistent with the real $\mathcal{G}$. Thus, we have

$$\begin{aligned} \Pr[\mathcal{A} \text{ wins}] &\geq \Pr[\mathsf{FindView}] \cdot \Pr[\mathcal{A} \text{ wins} | \mathsf{FindView}] \\ &= \Pr[\mathcal{A} \text{ wins} | \mathsf{FindView}] \end{aligned} \tag{6}$$

To analyze the winning probability of $\mathcal{A}$ when FindView occurs, let $S_{\mathsf{Enc\text{-}L}}$ denote the set of all valid group encodings that appear when running the algorithm $\Pi_{\mathsf{PKE}}.\mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$; these encodings are either the responses of labeling/addition queries or the valid inputs of the addition queries. Clearly, $|S_{\mathsf{Enc\text{-}L}}| \leq 3q$, since each algorithm makes at most q queries to $\mathcal{G}$. We then define:

$$S_{\mathsf{Enc}} := \big\{(x, \mathsf{str}) | \mathsf{str} \in S_{\mathsf{Enc\text{-}L}}, \mathcal{G}^{\mathsf{label}}(x) = \mathsf{str}\big\}.$$

Note that in each iteration, if the encoding pairs in $\tilde{S}_{\mathcal{A}}$ are consistent with S_{Enc}, the message decrypted in that iteration would be correct. Specifically, in such a context, another GGM instance exists that is consistent with both the simulation view of $\mathcal{A}$ and the real view of encryption, and the perfect correctness of Π_{PKE} ensures that the message decrypted in this iteration is correct. However, without knowledge of sk, $\tilde{S}_{\mathcal{A}}$ might be inconsistent with S_{Enc} with high probability. Specifically, there are three events:

- Event 1: There exists $(x, \mathsf{str}) \in \tilde{S}_{\mathcal{A}}$ and $(x, \mathsf{str}') \in S_{\mathsf{Enc}}$ such that $\mathsf{str} \neq \mathsf{str}'$.
- Event 2: There exists $(x, \mathsf{str}) \in \tilde{S}_{\mathcal{A}}$ and $(x', \mathsf{str}) \in S_{\mathsf{Enc}}$ such that $x \neq x'$.
- Event 3: For any $(x, \mathsf{str}) \in \tilde{S}_{\mathcal{A}}$ and $(x', \mathsf{str}') \in S_{\mathsf{Enc}}$ it holds that $x = x'$ if and only if $\mathsf{str} = \mathsf{str}'$.

We immediately observe that Event 1 occurs at most $3q$ times, since the updating phase would eliminate at least one pair in S_{Enc} with each occurrence. Next we show that, with a noticeable probability, Event 3 would deduce the valid shared key. For Event 3, according to the adversary described in Figure 2, we note that the set $\tilde{S}_{\mathcal{A}} \cup S_{\mathsf{que\text{-}res}}$ properly handles the labeling queries. For the addition queries, say $\mathsf{que} = (\mathsf{str}_1, \mathsf{str}_2)$, there are two cases:

- Case 1: $\tilde{S}_{\mathcal{A}} \cup S_{\mathsf{que\text{-}res}}$ covers $(x_1, \mathsf{str}_1), (x_2, \mathsf{str}_2)$, and $(x_1 + x_2, \mathsf{str}_3)$;
- Case 2: either str_1 or str_2 is not collected in $\tilde{S}_{\mathcal{A}} \cup S_{\mathsf{que\text{-}res}}$.

For the former case, $\tilde{S}_{\mathcal{A}} \cup S_{\mathsf{que\text{-}res}}$ responds to the addition query properly; for the latter case, $\tilde{S}_{\mathcal{A}} \cup S_{\mathsf{que\text{-}res}}$ always responds with $\perp$, indicating that if both str_1 and str_2 are valid group elements, the response is invalid. Therefore, the only bad case preventing Event 3 from correctly resuming the message is that the adversary $\mathcal{A}$ generates valid group elements str without making labeling queries (referred to below as a "new" group element). Besides, we observe that when Event 2 occurs, the adversary also generates such new group elements. For clarity, we denote the event in which $\mathcal{A}$ generate new group elements as New. Then we have

$$\begin{aligned}\Pr[\mathcal{A}\text{ wins}|\mathsf{FindView}] &\geq \Pr[\mathcal{A}\text{ wins}|\neg\mathsf{New} \wedge \mathsf{FindView}] \cdot \Pr[\neg\mathsf{New}|\mathsf{FindView}] \\ &= 1 - \Pr[\mathsf{New}|\mathsf{FindView}]. \end{aligned} \tag{7}$$

Since $\Pr[\mathsf{FindView}] = 1$, it is straightforward that

$$\Pr[\mathsf{New}|\mathsf{FindView}] = \Pr[\mathsf{New} \wedge \mathsf{FindView}]. \tag{8}$$

Hence, it suffices to prove that $\Pr[\mathsf{New} \wedge \mathsf{FindView}]$ is small.

In the following lemma, we introduce a simple modification to the MRA-secure game. Specifically, we replace the inputs of $\mathcal{A}$ from $\Pi_{\mathsf{PKE}}.\mathsf{KG}^{\mathcal{G}}(\mathsf{sk})$ and $\Pi_{\mathsf{PKE}}.\mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$ to $\mathsf{rpk}, \mathsf{c_r}$, where rpk and $\mathsf{c_r}$ are uniformly chosen from public-key space and ciphertext space, respectively. Let $\mathsf{FindView}_{\mathsf{rpk,rc}}$ and $\mathsf{New}_{\mathsf{rpk,rc}}$ denote the events where $\mathcal{A}$, given rpk and $\mathsf{c_r}$ as inputs, finds a proper view in each iteration and generates new group elements, respectively.

Lemma A.1. *There exists a negligible function ϵ such that*

$$\Delta = \big|\Pr[\mathsf{New} \wedge \mathsf{FindView}] - \Pr[\mathsf{New}_{\mathsf{rpk,rc}} \wedge \mathsf{FindView}_{\mathsf{rpk,rc}}]\big| \leq \epsilon(\lambda).$$

Adversary $\mathcal{B}_1$

$\underline{\mathcal{B}_1^{\mathcal{G}}(\mathsf{pk}, \mathsf{c})}$:

$S_{\text{que-res}} \leftarrow \emptyset$; $\mathsf{Tag}_1 = 1$; $\mathsf{Tag}_2 = 0$;

For $i = 1$ to $6q + 1$:

[Phase I: Searching a proper view of Alice]

Search a secret key $\widetilde{\mathsf{sk}}$ and a set of labeling query-response tuples $\tilde{S}_{\mathcal{A}}$, subject to the following requirements:

1. $\tilde{S}_{\mathcal{A}}$ is consistent with $S_{\text{que-res}}$ and $|\tilde{S}_{\mathcal{A}}| \leq 6q$;
2. $\mathsf{pk} \leftarrow \Pi_{\mathsf{PKE}}.\mathsf{KG}^{S_{\text{que-res}} \cup \tilde{S}_{\mathcal{A}}}(\widetilde{\mathsf{sk}})$;
3. When running the algorithm $\widetilde{\mathsf{m}}_i \leftarrow \Pi_{\mathsf{PKE}}.\mathsf{Dec}^{S_{\text{que-res}} \cup \tilde{S}_{\mathcal{A}}}(\widetilde{\mathsf{sk}}, \mathsf{c})$, $S_{\text{que-res}}$ is sufficient such that:
   - for a labeling query $\mathsf{que} = x$, there must be (x, res) in $S_{\text{que-res}} \cup \tilde{S}_{\mathcal{A}}$ and responds with res;
   - for an addition query $\mathsf{que} = (\mathsf{str}_1, \mathsf{str}_2)$:
     · if either str_1 or str_2 is not covered in $S_{\text{que-res}} \cup \tilde{S}_{\mathcal{A}}$, then responds with $\perp$;
     · otherwise, there must be (x_1, str_1), (x_2, str_2) and $(x_1 + x_2, \mathsf{str}_3)$ in $S_{\text{que-res}} \cup \tilde{S}_{\mathcal{A}}$, and responds with str_3.

If there is no such $\widetilde{\mathsf{sk}}$ and $\tilde{S}_{\mathcal{A}}$: $\mathsf{Tag}_1 \leftarrow 0$;

Else: $S_{\mathsf{m}} \leftarrow S_{\mathsf{m}} \cup \{\widetilde{\mathsf{m}}_i\}$;

For each $(x, \mathsf{str}) \in \tilde{S}_{\mathcal{A}} \setminus S_{\text{que-res}}$: if $\mathcal{G}^{\mathsf{add}}(\mathsf{str}, \mathsf{str}) \neq \perp$, $\mathsf{Tag}_2 \leftarrow 1$;

Re-run $\widetilde{\mathsf{m}}_i \leftarrow \Pi_{\mathsf{PKE}}.\mathsf{Dec}^{S_{\text{que-res}} \cup \tilde{S}_{\mathcal{A}}}(\widetilde{\mathsf{sk}}, \mathsf{c})$ and $\mathsf{que} = (\mathsf{str}_1, \mathsf{str}_2)$ is an addition query during the execution:

if $\mathsf{str}_1 \notin S_{\text{que-res}}$ and $\mathcal{G}^{\mathsf{add}}(\mathsf{str}_1, \mathsf{str}_1) \neq \perp$, $\mathsf{Tag}_2 \leftarrow 1$;
if $\mathsf{str}_2 \notin S_{\text{que-res}}$ and $\mathcal{G}^{\mathsf{add}}(\mathsf{str}_2, \mathsf{str}_2) \neq \perp$, $\mathsf{Tag}_2 \leftarrow 1$;

[Phase II: Updating the guessing queries with real oracle]

For each $(x, \mathsf{str}) \in \tilde{S}_{\mathcal{A}} \setminus S_{\text{que-res}}$: $S_{\text{que-res}} \leftarrow S_{\text{que-res}} \cup (x, \mathcal{G}^{\mathsf{label}}(x))$;

Return $\mathsf{Tag}_1 \cdot \mathsf{Tag}_2$.

Figure 3: The description of the adversary $\mathcal{B}_1$.

Proof. To complete the proof, we leverage the pseudorandom public key and pseudorandom ciphertext properties of Π_{PKE}. Specifically, we introduce an alternative scenario where $\mathcal{A}$ is given $\Pi_{\mathsf{PKE}}.\mathsf{KG}^{\mathcal{G}}(\mathsf{sk})$ and a ciphertext c_r that is uniformly sampled from the ciphertext space. Let $\mathsf{FindView}_{\mathsf{rc}}$ and $\mathsf{New}_{\mathsf{rc}}$ represent the events in this scenario where $\mathcal{A}$ identifies a valid view in each iteration and generates new group elements, respectively. The proof proceeds in two steps:

1. Showing that
$$\Delta_1 = \left| \Pr[\mathsf{New} \wedge \mathsf{FindView}] - \Pr[\mathsf{New}_{\mathsf{rc}} \wedge \mathsf{FindView}_{\mathsf{rc}}] \right| \leq \epsilon_1(\lambda).$$
2. Showing that
$$\Delta_2 = \left| \Pr[\mathsf{New}_{\mathsf{rc}} \wedge \mathsf{FindView}_{\mathsf{rc}}] - \Pr[\mathsf{New}_{\mathsf{rpk},\mathsf{rc}} \wedge \mathsf{FindView}_{\mathsf{rpk},\mathsf{rc}}] \right| \leq \epsilon_2(\lambda).$$

Here ϵ_1 and ϵ_2 are two negligible functions.

For step 1, we construct an adversary $\mathcal{B}_1$, as illustrated in Figure 3, to distinguish between the distributions $(\Pi_{\mathsf{PKE}}.\mathsf{KG}^{\mathcal{G}}(\mathsf{sk}), \Pi_{\mathsf{PKE}}.\mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathsf{m}, \mathsf{r}))$ and $(\Pi_{\mathsf{PKE}}.\mathsf{KG}^{\mathcal{G}}(\mathsf{sk}), \mathsf{c_r})$. The adversary $\mathcal{B}_1$ takes a public key and a ciphertext as input, and initialises two bits, Tag_1 and Tag_2, to track the occurrence of two specific events. The adversary's behavior closely mirrors that of $\mathcal{A}$ in the MRA-secure game, depicted in Figure 2. Specifically, $\mathcal{B}_1$ proceeds two phases in each iteration. In Phase I, if a proper view cannot be identified, then set Tag_1 from initial 1 to 0; and if a new group element appears, either in $\tilde{S}_{\mathcal{A}}$ or as an input of some addition queries, then sets Tag_2 from initial 0 to 1 (highlighted in red). Observe that once the GGM instance, the valid public key, and the corresponding valid ciphertext are fixed, $\mathcal{B}_1$ proceeds identically to the adversary $\mathcal{A}$ in the MRA-secure game. Finally, $\mathcal{B}_1$ outputs 1 if and only if both Tag_1 and Tag_2 are equal to 1, indicating that $\mathcal{B}_1$ finds a proper view in each iteration and generates new group elements. Thus, we have

$$\Pr[\mathsf{New} \wedge \mathsf{FindView}] = \Pr[\mathcal{B}_1^{\mathcal{G}}(\Pi_{\mathsf{PKE}}.\mathsf{KG}^{\mathcal{G}}(\mathsf{sk}), \Pi_{\mathsf{PKE}}.\mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathsf{m}, \mathsf{r})) = 1].$$

Analogously,

$$\Pr[\mathsf{New}_{\mathsf{rc}} \wedge \mathsf{FindView}_{\mathsf{rc}}] = \Pr[\mathcal{B}_1^{\mathcal{G}}(\Pi_{\mathsf{PKE}}.\mathsf{KG}^{\mathcal{G}}(\mathsf{sk}), \mathsf{c_r}) = 1].$$

Therefore, if there exists a polynomial poly such that

$$\Delta_1 \geq \frac{1}{\mathsf{poly}(\lambda)}$$

for infinitely many values of λ, it directly follows that

$$\left| \Pr[\mathcal{B}_1^{\mathcal{G}}(\Pi_{\mathsf{PKE}}.\mathsf{KG}^{\mathcal{G}}(\mathsf{sk}), \Pi_{\mathsf{PKE}}.\mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathsf{m}, \mathsf{r})) = 1] - \Pr[\mathcal{B}_1^{\mathcal{G}}(\Pi_{\mathsf{PKE}}.\mathsf{KG}^{\mathcal{G}}(\mathsf{sk}), \mathsf{c_r}) = 1] \right| \geq \frac{1}{\mathsf{poly}(\lambda)},$$

indicating that $\mathcal{B}_1$ can efficiently distinguish between the ciphertext generated by Enc through the valid public key and uniformly sampled ciphertext, leading into a contraposition of the pseudorandom ciphertext of Π_{PKE}.

For step 2, we construct an adversary $\mathcal{B}_2$, as illustrated in Figure 4, to distinguish between the distributions $(\Pi_{\mathsf{PKE}}.\mathsf{KG}^{\mathcal{G}}(\mathsf{sk}), \mathsf{c_r})$ and $(\mathsf{rpk}, \mathsf{c_r})$. Following the similar strategy, we have

$$\Pr[\mathsf{New}_{\mathsf{rc}} \wedge \mathsf{FindView}_{\mathsf{rc}}] = \Pr[\mathcal{B}_2^{\mathcal{G}}(\Pi_{\mathsf{PKE}}.\mathsf{KG}^{\mathcal{G}}(\mathsf{sk})) = 1],$$

and

$$\Pr[\mathsf{New}_{\mathsf{rpk,rc}} \wedge \mathsf{FindView}_{\mathsf{rpk,rc}}] = \Pr[\mathcal{B}_2^{\mathcal{G}}(\mathsf{rpk}) = 1].$$

Therefore, if $\mathcal{B}_2$ can efficiently distinguish between a public key generated by KG and a uniformly sampled public key, this would contradict the pseudorandom public key property of Π_{PKE}.

Thus, we finish the proof of Lemma A.1.

□ □

Combining Eq. (6), (7), (8) and Lemma A.1, we obtain

$$\Pr[\mathcal{A} \text{ wins}] \geq 1 - \epsilon(\lambda) - \Pr[\mathsf{New}_{\mathsf{rpk,rc}} \wedge \mathsf{FindView}_{\mathsf{rpk,rc}}]. \tag{9}$$

Therefore, it suffices to prove that $\Pr[\mathsf{New}_{\mathsf{rpk,rc}} \wedge \mathsf{FindView}_{\mathsf{rpk,rc}}]$ is small.

Consider choosing a GGM $\mathcal{G}$ along with a public key rpk and a ciphertext $\mathsf{c_r}$, and fixing them. We observe that once the randomness for the $\mathcal{A}$ is set, the algorithm $\mathcal{A}^{\mathcal{G}}(\mathsf{rpk}, \mathsf{c_r})$ becomes deterministic. Here, rpk and $\mathsf{c_r}$ are uniformly chosen from public-key space and ciphertext space, respectively. Thus, we say a tuple $(\mathcal{G}, \mathsf{rpk}, \mathsf{c_r})$ is "good" if

$$\Pr[\mathsf{New}_{\mathsf{rpk,rc}} \wedge \mathsf{FindView}_{\mathsf{rpk,rc}} \text{ occurs with respect to } (\mathcal{G}, \mathsf{rpk}, \mathsf{c_r})] < \frac{1}{8},$$

Adversary $\mathcal{B}_2$

$\underline{\mathcal{B}_2^{\mathcal{G}}(\mathsf{pk})}$:

$\mathsf{c_r} \xleftarrow{\$} \mathcal{C}_\lambda$; $S_{\text{que-res}} \leftarrow \emptyset$; $\mathsf{Tag}_1 = 1$; $\mathsf{Tag}_2 = 0$

For $i = 1$ to $6q + 1$:

[Phase I: Searching a proper view of Alice]

Search a secret key $\widetilde{\mathsf{sk}}$ and a set of labeling query-response tuples $\tilde{S}_{\mathcal{A}}$, subject to the following requirements:

1. $\tilde{S}_{\mathcal{A}}$ is consistent with $S_{\text{que-res}}$ and $|\tilde{S}_{\mathcal{A}}| \leq 6q$;
2. $\mathsf{pk} \leftarrow \Pi_{\mathsf{PKE}}.\mathsf{KG}^{S_{\text{que-res}} \cup \tilde{S}_{\mathcal{A}}}(\widetilde{\mathsf{sk}})$;
3. When running the algorithm $\widetilde{\mathsf{m}}_i \leftarrow \Pi_{\mathsf{PKE}}.\mathsf{Dec}^{S_{\text{que-res}} \cup \tilde{S}_{\mathcal{A}}}(\widetilde{\mathsf{sk}}, \mathsf{c_r})$, $S_{\text{que-res}}$ is sufficient such that:
   - for a labeling query $\mathsf{que} = x$, there must be (x, res) in $S_{\text{que-res}} \cup \tilde{S}_{\mathcal{A}}$ and responds with res;
   - for an addition query $\mathsf{que} = (\mathsf{str}_1, \mathsf{str}_2)$:
     · if either str_1 or str_2 is not covered in $S_{\text{que-res}} \cup \tilde{S}_{\mathcal{A}}$, then responds with $\perp$;
     · otherwise, there must be (x_1, str_1), (x_2, str_2) and $(x_1 + x_2, \mathsf{str}_3)$ in $S_{\text{que-res}} \cup \tilde{S}_{\mathcal{A}}$, and responds with str_3.

If there is no such $\widetilde{\mathsf{sk}}$ and $\tilde{S}_{\mathcal{A}}$: $\mathsf{Tag}_1 \leftarrow 0$;

Else: $S_{\mathsf{m}} \leftarrow S_{\mathsf{m}} \cup \{\widetilde{\mathsf{m}}_i\}$;

For each $(x, \mathsf{str}) \in \tilde{S}_{\mathcal{A}} \setminus S_{\text{que-res}}$: if $\mathcal{G}^{\mathsf{add}}(\mathsf{str}, \mathsf{str}) \neq \perp$, $\mathsf{Tag}_2 \leftarrow 1$;

Re-run $\widetilde{\mathsf{m}}_i \leftarrow \Pi_{\mathsf{PKE}}.\mathsf{Dec}^{S_{\text{que-res}} \cup \tilde{S}_{\mathcal{A}}}(\widetilde{\mathsf{sk}}, \mathsf{c_r})$ and $\mathsf{que} = (\mathsf{str}_1, \mathsf{str}_2)$ is an addition query during the execution:

if $\mathsf{str}_1 \notin S_{\text{que-res}}$ and $\mathcal{G}^{\mathsf{add}}(\mathsf{str}_1, \mathsf{str}_1) \neq \perp$, $\mathsf{Tag}_2 \leftarrow 1$;
if $\mathsf{str}_2 \notin S_{\text{que-res}}$ and $\mathcal{G}^{\mathsf{add}}(\mathsf{str}_2, \mathsf{str}_2) \neq \perp$, $\mathsf{Tag}_2 \leftarrow 1$;

[Phase II: Updating the guessing queries with real oracle]

For each $(x, \mathsf{str}) \in \tilde{S}_{\mathcal{A}} \setminus S_{\text{que-res}}$: $S_{\text{que-res}} \leftarrow S_{\text{que-res}} \cup (x, \mathcal{G}^{\mathsf{label}}(x))$;

Return $\mathsf{Tag}_1 \cdot \mathsf{Tag}_2$.

Figure 4: The description of the adversary $\mathcal{B}_2$.

where the probability is over the internal randomness of $\mathcal{A}$. Analogously, we say a tuple $(\mathcal{G}, \mathsf{rpk}, \mathsf{c_r})$ is "bad" if

$$\Pr[\mathsf{New}_{\mathsf{rpk},\mathsf{rc}} \wedge \mathsf{FindView}_{\mathsf{rpk},\mathsf{rc}} \text{ occurs with respect to } (\mathcal{G}, \mathsf{rpk}, \mathsf{c_r})] \geq \frac{1}{8}.$$

For clarity, we denote these two events in which the tuple $(\mathcal{G}, \mathsf{rpk}, \mathsf{c_r})$ is good or bad as GoodTuple and BadTuple, respectively. It follows that

$$\Pr[\mathsf{New}_{\mathsf{rpk},\mathsf{rc}} \wedge \mathsf{FindView}_{\mathsf{rpk},\mathsf{rc}}] \leq \Pr[\mathsf{BadTuple}] + \frac{1}{8}. \tag{10}$$

To establish that $\Pr[\mathsf{BadTuple}]$ is small, we leverage the same strategy as shown in Section 3—the property of the GGM. We show that if $\Pr[\mathsf{BadTuple}]$ is high, then the adversary $\mathcal{A}$ is likely to generate a new

Adversary $\mathcal{E}^{\mathcal{G}}(1^\lambda)$

$\mathsf{rpk} \xleftarrow{\$} \mathcal{PK}_\lambda$; $\mathsf{c_r} \xleftarrow{\$} \mathcal{C}_\lambda$; $S_{\mathsf{que\text{-}res}} \leftarrow \emptyset$;

For $i = 1$ to $6q + 1$:

[Phase I: Searching a proper view of Alice]

Search a secret key $\widetilde{\mathsf{sk}}$ and a set of labeling query-response tuples $\tilde{S}_{\mathcal{A}}$, subject to the following requirements:

1. $\tilde{S}_{\mathcal{A}}$ is consistent with $S_{\mathsf{que\text{-}res}}$ and $|\tilde{S}_{\mathcal{A}}| \leq 6q$;
2. $\mathsf{rpk} \leftarrow \Pi_{\mathsf{PKE}}.\mathsf{KG}^{S_{\mathsf{que\text{-}res}} \cup \tilde{S}_{\mathcal{A}}}(\widetilde{\mathsf{sk}})$;
3. When running the algorithm $\widetilde{\mathsf{m}}_i \leftarrow \Pi_{\mathsf{PKE}}.\mathsf{Dec}^{S_{\mathsf{que\text{-}res}} \cup \tilde{S}_{\mathcal{A}}}(\widetilde{\mathsf{sk}}, \mathsf{c_r})$, $S_{\mathsf{que\text{-}res}}$ is sufficient such that:
   - for a labeling query $\mathsf{que} = x$, there must be (x, res) in $S_{\mathsf{que\text{-}res}} \cup \tilde{S}_{\mathcal{A}}$ and responds with res;
   - for an addition query $\mathsf{que} = (\mathsf{str}_1, \mathsf{str}_2)$:
     - if either str_1 or str_2 is not covered in $S_{\mathsf{que\text{-}res}} \cup \tilde{S}_{\mathcal{A}}$, then responds with $\perp$;
     - otherwise, there must be (x_1, str_1), (x_2, str_2) and $(x_1 + x_2, \mathsf{str}_3)$ in $S_{\mathsf{que\text{-}res}} \cup \tilde{S}_{\mathcal{A}}$, and responds with str_3.

If there is no such $\widetilde{\mathsf{sk}}$ and $\tilde{S}_{\mathcal{A}}$: return $\perp$

For each $(x, \mathsf{str}) \in \tilde{S}_{\mathcal{A}} \setminus S_{\mathsf{que\text{-}res}}$: if $\mathcal{G}^{\mathsf{add}}(\mathsf{str}, \mathsf{str}) \neq \perp$, return str;

Re-run $\widetilde{\mathsf{m}}_i \leftarrow \Pi_{\mathsf{PKE}}.\mathsf{Dec}^{S_{\mathsf{que\text{-}res}} \cup \tilde{S}_{\mathcal{A}}}(\widetilde{\mathsf{sk}}, \mathsf{c_r})$ and $\mathsf{que} = (\mathsf{str}_1, \mathsf{str}_2)$ is an addition query during the execution:

if $\mathsf{str}_1 \notin S_{\mathsf{que\text{-}res}}$ and $\mathcal{G}^{\mathsf{add}}(\mathsf{str}_1, \mathsf{str}_1) \neq \perp$, return str_1;
if $\mathsf{str}_2 \notin S_{\mathsf{que\text{-}res}}$ and $\mathcal{G}^{\mathsf{add}}(\mathsf{str}_2, \mathsf{str}_2) \neq \perp$, return str_2;

[Phase II: Updating the guessing queries with real oracle]

For each $(x, \mathsf{str}) \in \tilde{S}_{\mathcal{A}} \setminus S_{\mathsf{que\text{-}res}}$: $S_{\mathsf{que\text{-}res}} \leftarrow S_{\mathsf{que\text{-}res}} \cup (x, \mathcal{G}^{\mathsf{label}}(x))$;

Return $\perp$.

Figure 5: The description of $\mathcal{E}$ w.r.t. $\Pi^{\mathcal{G}}_{\mathsf{PKE}}$ that outputs a new group element of $\mathcal{G}$.

group element with a good probability, which contradicts the assumed hardness of producing a new group element in the GGM. Specifically, our proof proceeds in two steps:

1. Construct an extractor $\mathcal{E}$ that takes only the security parameter as input;
2. Demonstrate that, if $\Pr[\mathsf{BadTuple}]$ is high, then $\mathcal{E}$ outputs a new group element with a good probability.

We present the formal description of the extractor $\mathcal{E}$, as depicted in Figure 5. The extractor $\mathcal{E}$ takes only the security parameter as input, randomly selects a public key rpk and a ciphertext $\mathsf{c_r}$, and proceeds in a manner closely resembling the adversary $\mathcal{A}$ as shown in Figure 2. Specifically, the extractor proceeds two phases in each iteration, and during the phase I in each iteration, if a new group element appears, either in

$\tilde{S}_{\mathcal{A}}$ or as an input of some addition queries, then $\mathcal{E}$ outputs this element (highlighted in red). Observe that the randomness of $\mathcal{E}$ is composed of three components: rpk, $\mathsf{c_r}$, and the randomness used in each iteration. Furthermore, once the GGM instance, the public key, and the ciphertext are fixed, the extractor $\mathcal{E}$ proceeds identically to the adversary $\mathcal{A}$, continuing until $\mathcal{E}$ successfully outputs a new group element.

Next, we analyze the probability of the extractor wins, which occurs when $\mathcal{E}$ outputs a new group element. Let $\mathcal{G}$ denote the GGM to which the extractor has access. Let rpk and $\mathsf{c_r}$ denote the public key and ciphertext sampled by $\mathcal{E}$. We define the pair $(\mathsf{rpk}, \mathsf{c_r})$ to be bad with respect to $\mathcal{G}$, if the tuple $(\mathcal{G}, \mathsf{rpk}, \mathsf{c_r})$ is a $\mathsf{BadTuple}$. For clarity, we refer to this event as $\mathsf{BadPair}$ in the following analysis. According to the definition of $\mathsf{BadTuple}$, we note that if $\mathcal{E}$ fortunately samples a bad pair, i.e., $\mathsf{BadPair}$ occurs, then $\mathcal{E}$ wins with a good probability. Concretely,

$$\Pr[\mathcal{E}^{\mathcal{G}} \text{ wins} | \mathsf{BadPair}] \geq \frac{1}{8}.$$

Next, we analyze the probability of $\mathsf{BadPair}$ with respect to any specific GGM instance. We introduce the concept of "bad" generic group model. Specifically, for any instance of GGM, denoted as $\mathcal{G}$, we say $\mathcal{G}$ is a bad instance if there are more than $\frac{1}{8} \cdot |\mathcal{PK}_\lambda| \cdot |\mathcal{C}_\lambda|$ pairs of $(\mathsf{rpk}, \mathsf{c_r})$ such that $(\mathcal{G}, \mathsf{rpk}, \mathsf{c_r})$ is a $\mathsf{BadTuple}$. Then, we have

$$\begin{aligned}\Pr[\mathcal{E}^{\mathcal{G}} \text{ wins} | \mathcal{G} \text{ is bad}] &\geq \Pr[\mathcal{E}^{\mathcal{G}} \text{ wins} | \mathsf{BadPair}] \cdot \frac{\text{Num of bad pairs}}{\text{Num of all pairs}} \\ &\geq \frac{1}{8} \cdot \frac{1}{8} = \frac{1}{64}.\end{aligned}$$

Next, we analyze the probability of $\mathcal{E}$ wins. Specifically,

$$\begin{aligned}\Pr[\mathcal{E} \text{ wins}] &\geq \Pr[\mathcal{E}^{\mathcal{G}} \text{ wins} | \mathcal{G} \text{ is bad}] \cdot \frac{\text{Num of bad GGM instances}}{\text{Num of all GGM instances}} \\ &\geq \frac{1}{64} \cdot \frac{\text{Num of bad GGM instances}}{\text{Num of all GGM instances}}.\end{aligned}$$

For clarity, let $\mathsf{TotalGGM}$ denote the set of all GGM instances and BadGGM the subset of bad ones. Define

$$\mathsf{BadRatio} := \frac{|\mathsf{BadGGM}|}{|\mathsf{TotalGGM}|}.$$

Thus, the probability that the extractor $\mathcal{E}$ wins is bounded below by

$$\Pr[\mathcal{E} \text{ wins}] \geq \frac{1}{64} \cdot \mathsf{BadRatio}.$$

Lemma A.2. *If* $\Pr[\mathsf{BadTuple}] \geq \frac{1}{4}$, *then* $\mathsf{BadRatio} \geq \frac{1}{8}$.

Proof. By definition, we have

$$\begin{aligned}\Pr[\mathsf{BadTuple}] &= \frac{\sum_{\mathcal{G} \in \mathsf{TotalGGM}} \left(\text{Num of BadPair w.r.t. } \mathcal{G}\right)}{|\mathsf{TotalGGM}| \cdot |\mathcal{PK}_\lambda| \cdot |\mathcal{C}_\lambda|} \\ &\leq \frac{\sum_{\mathcal{G} \in \mathsf{BadGGM}} |\mathcal{PK}_\lambda| \cdot |\mathcal{C}_\lambda|}{|\mathsf{TotalGGM}| \cdot |\mathcal{PK}_\lambda| \cdot |\mathcal{C}_\lambda|} + \frac{\sum_{\mathcal{G} \notin \mathsf{BadGGM}} \frac{1}{8} \cdot |\mathcal{PK}_\lambda| \cdot |\mathcal{C}_\lambda|}{|\mathsf{TotalGGM}| \cdot |\mathcal{PK}_\lambda| \cdot |\mathcal{C}_\lambda|} \\ &\leq \frac{|\mathsf{BadGGM}|}{|\mathsf{TotalGGM}|} + \frac{1}{8}.\end{aligned}$$

Here, the summations $\sum_{\mathcal{G} \in \mathsf{TotalGGM}}$ and $\sum_{\mathcal{G} \in \mathsf{BadGGM}}$ are taken over all GGM instances and bad GGM instances, respectively.

Therefore, if $\Pr[\mathsf{BadTuple}] \geq \frac{1}{4}$, we conclude that

$$\mathsf{BadRatio} = \frac{|\mathsf{BadGGM}|}{|\mathsf{TotalGGM}|} \geq \Pr[\mathsf{BadTuple}] - \frac{1}{8} \geq \frac{1}{8}.$$

□ □

Lemma A.2 implies that if $\Pr[\mathsf{BadTuple}] \geq \frac{1}{4}$, then

$$\Pr[\mathcal{E} \text{ wins}] \geq \frac{1}{64} \cdot \mathsf{BadRatio} \geq \frac{1}{512},$$

which contradicts the assumed hardness of producing a new group element in the GGM. Therefore, we claim that $\Pr[\mathsf{BadTuple}] < \frac{1}{4}$. Combining Eq. (9) and (10), we have

$$\Pr[\mathcal{A} \text{ wins}] \geq \frac{5}{8} - \epsilon(\lambda).$$

Therefore, the advantage of $\mathcal{A}$ in the message-recovery attack game is

$$\left|\Pr[\mathcal{A} \text{ wins}] - \frac{1}{|\mathcal{M}|}\right| \geq \frac{5}{8} - \epsilon(\lambda) - \frac{1}{|\mathcal{M}|}.$$

Since $|\mathcal{M}| \geq 2$, we have $1/|\mathcal{M}| \leq 1/2$, and hence the advantage is at least $1/8 - \epsilon(\lambda)$.

This settles the entire proof.

□
□

B Proof of Theorem 5.1

Theorem 5.1. $\Pi^{\mathcal{O}}_{\mathsf{KEM}}$ is indifferentiable from the ideal KEM $\mathbf{KEM}$. More precisely, there exists a simulator $\mathcal{S}$ such that for all q-query differentiator $\mathcal{D}$, we have

$$\mathrm{Adv}^{\mathsf{indiff}}_{\Pi^{\mathcal{O}}_{\mathsf{KEM}},\mathbf{KEM},\mathcal{S}}(\mathcal{D}) \leq O\left(\frac{(\lambda \cdot q)^4}{N} + \frac{(\lambda \cdot q)^3}{2^\lambda}\right).$$

Here, the simulator makes at most $O(\lambda \cdot q)$ queries to its oracles.

Proof. Correctness of our scheme follows immediately. For indifferentiability, note that in the real world the differentiator interacts with three honest interfaces $(\Pi_{\mathsf{KEM}}.\mathsf{KG}, \Pi_{\mathsf{KEM}}.\mathsf{Encaps}, \Pi_{\mathsf{KEM}}.\mathsf{Decaps})$, along with the adversarial interfaces consisting of the random oracles $\mathcal{H}_x$, $\mathcal{H}_{\mathsf{seed}}$, $\mathcal{H}_y$, $\mathcal{H}_{\mathsf{mask}}$, and $\mathcal{H}_{\mathsf{key}}$, the random permutation $(\mathcal{P}, \mathcal{P}^{-1})$, and the generic group $\mathcal{G} = (\mathcal{G}^{\mathsf{label}}, \mathcal{G}^{\mathsf{add}})$. To establish the proof, we construct an efficient simulator $\mathcal{S}$ in the ideal world such that:

1. $\mathcal{S}$ has access to the ideal KEM $\mathbf{KEM}$;
2. $\mathcal{S}$ simulates those nine adversarial interfaces

$$(\mathcal{S}.\mathcal{H}_x, \mathcal{S}.\mathcal{H}_{\mathsf{seed}}, \mathcal{S}.\mathcal{H}_y, \mathcal{S}.\mathcal{H}_{\mathsf{mask}}, \mathcal{S}.\mathcal{H}_{\mathsf{key}}, \mathcal{S}.\mathcal{P}, \mathcal{S}.\mathcal{P}^{-1}, \mathcal{S}.\mathcal{G}^{\mathsf{label}}, \mathcal{S}.\mathcal{G}^{\mathsf{add}}).$$

To describe the simulator precisely, we first introduce some notation. Let $\mathsf{sk}_{\mathsf{test}} \in \mathcal{SK}$ be a fixed but otherwise arbitrary private key. Let $\mathsf{r}_{\mathsf{test}} \in \mathcal{R}$ be a fixed but otherwise arbitrary randomness. In the generic group model, a labeling query $\mathsf{str} \leftarrow \mathcal{G}^{\mathsf{label}}(x)$ corresponds to a linear equation $(\mathsf{str} - x \cdot \diamondsuit = 0)$, where $\diamondsuit$ represents the generator, i.e., $\diamondsuit = \mathcal{G}^{\mathsf{label}}(1)$. Similarly, an addition query $\mathsf{str} \leftarrow \mathcal{G}^{\mathsf{add}}(\mathsf{str}_1, \mathsf{str}_2)$ corresponds to a linear equation $(\mathsf{str} - \mathsf{str}_1 - \mathsf{str}_2 = 0)$. Let T_{eq} denote the set of linear equations resulting from these two types of queries. We write $\mathsf{Eq} \in \mathsf{Span}(T_{\mathsf{eq}})$ to indicate that the equation Eq lies in the linear span of the equations in T_{eq}.

In the following, we illustrate the full description of our simulator.

Simulator $\mathcal{S}$

$\underline{\mathcal{S}.\mathcal{H}_x(\mathsf{sk})}$:

if $\exists(\mathsf{sk}, x) \in T_{\mathcal{H}_x}$, then **return** x; ..... Case 1

else:

$x \xleftarrow{\$} \mathbb{Z}_N$; $T_{\mathcal{H}_x} \leftarrow T_{\mathcal{H}_x} \cup \{(\mathsf{sk}, x)\}$;

$\mathsf{pk} \leftarrow \mathbf{KEM}.\mathsf{KG}(\mathsf{sk})$;

$T_{\mathsf{eq}} \leftarrow T_{\mathsf{eq}} \cup \{\mathsf{pk} - x \cdot \diamondsuit = 0\}$;

return x; ..... Case 2

$\underline{\mathcal{S}.\mathcal{H}_{\mathsf{seed}}(\mathsf{pk}, \mathsf{r})}$:

if $\exists(\mathsf{pk}||\mathsf{r}, \mathsf{seed}) \in T_{\mathcal{H}_{\mathsf{seed}}}$, then **return** seed; ..... Case 1

$(\mathsf{c}, \mathsf{k}) \leftarrow \mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$;

if $\exists(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c}) \in T_{\mathcal{P}}$:

if $\exists x \in \mathbb{Z}_N$ s.t. $(\mathsf{pk} - x \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$:

if $\exists(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask}) \in T_{\mathcal{H}_{\mathsf{mask}}}$ with $(\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$:

$\mathsf{seed} \leftarrow \mathsf{mask} \oplus \tilde{\mathsf{c}}_2$; $T_{\mathcal{H}_{\mathsf{seed}}} \leftarrow T_{\mathcal{H}_{\mathsf{seed}}} \cup \{\mathsf{pk}||\mathsf{r}, \mathsf{seed}\}$;

return seed; ..... Case 2

$\mathsf{seed} \leftarrow \{0,1\}^\lambda$; $T_{\mathcal{H}_{\mathsf{seed}}} \leftarrow T_{\mathcal{H}_{\mathsf{seed}}} \cup \{\mathsf{pk}||\mathsf{r}, \mathsf{seed}\}$;

return seed; ..... Case 3

$\underline{\mathcal{S}.\mathcal{H}_y(\mathsf{seed})}$:

if $\exists(\mathsf{seed}, y) \in T_{\mathcal{H}_y}$, then **return** y; ..... Case 1

$y \xleftarrow{\$} \mathbb{Z}_N$; $T_{\mathcal{H}_y} \leftarrow T_{\mathcal{H}_y} \cup \{(\mathsf{seed}, y)\}$;

if $\exists(\mathsf{pk}||\mathsf{r}, \mathsf{seed}) \in T_{\mathcal{H}_{\mathsf{seed}}}$:

$(\mathsf{c}, \mathsf{k}) \leftarrow \mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$;

if $\exists(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c}) \in T_{\mathcal{P}}$: $T_{\mathsf{eq}} \leftarrow T_{\mathsf{eq}} \cup \{\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0\}$;

for any $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c}) \in T_{\mathcal{P}}$ and $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask}) \in T_{\mathcal{H}_{\mathsf{mask}}}$ s.t. $\mathsf{seed} = \tilde{\mathsf{c}}_2 \oplus \mathsf{mask}$:

if $\exists x \in \mathbb{Z}_N$ s.t. $(\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$:

if $\exists(\mathsf{sk}, x) \in T_{\mathcal{H}_x}$ s.t. $\mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c}) \in \mathcal{K}$:

$T_{\mathsf{eq}} \leftarrow T_{\mathsf{eq}} \cup \{\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0\}$;

if $\exists(\mathsf{sk}, \mathsf{pk}) \in T^*_{\mathsf{sk}}$ s.t. $\mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c}) \in \mathcal{K}$:

if $(\mathsf{pk} - x \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$: $T_{\mathsf{eq}} \leftarrow T_{\mathsf{eq}} \cup \{\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0\}$;

return y; ..... Case 2

$\underline{\mathcal{S}.\mathcal{H}_{\mathsf{mask}}(\tilde{\mathsf{c}}_1||\mathsf{str})}$:

if $\exists(\tilde{\mathsf{c}}_1||\mathsf{str},\mathsf{mask}) \in T_{\mathcal{H}_{\mathsf{mask}}}$, then **return** mask; ..... Case 1

for any $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2,\mathsf{c}) \in T_{\mathcal{P}}$:

if $\exists(\mathsf{pk}||\mathsf{r},\mathsf{seed}) \in T_{\mathcal{H}_{\mathsf{seed}}}$ s.t. $\mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk},\mathsf{r}) = (\mathsf{c},*)$:

if $\exists(\mathsf{seed},y) \in T_{\mathcal{H}_y}$ s.t. $(\mathsf{str} - y \cdot \mathsf{pk} = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$:

$\mathsf{mask} = \mathsf{seed} \oplus \tilde{\mathsf{c}}_2$;

$T_{\mathcal{H}_{\mathsf{mask}}} \leftarrow T_{\mathcal{H}_{\mathsf{mask}}} \cup \{(\tilde{\mathsf{c}}_1||\mathsf{str},\mathsf{mask})\}$;

return mask; ..... Case 2

if $\exists x \in \mathbb{Z}_N$ s.t. $(\mathsf{pk} - x \cdot \diamondsuit = 0), (\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$:

$\mathsf{mask} = \mathsf{seed} \oplus \tilde{\mathsf{c}}_2$;

$T_{\mathcal{H}_{\mathsf{mask}}} \leftarrow T_{\mathcal{H}_{\mathsf{mask}}} \cup \{(\tilde{\mathsf{c}}_1||\mathsf{str},\mathsf{mask})\}$;

return mask; ..... Case 3

$\mathsf{mask} \xleftarrow{\$} \{0,1\}^\lambda$; $T_{\mathcal{H}_{\mathsf{mask}}} \leftarrow T_{\mathcal{H}_{\mathsf{mask}}} \cup \{(\tilde{\mathsf{c}}_1||\mathsf{str},\mathsf{mask})\}$;

return mask; ..... Case 4

$\underline{\mathcal{S}.\mathcal{H}_{\mathsf{key}}(\mathsf{pk}||\mathsf{seed})}$:

if $\exists(\mathsf{pk}||\mathsf{seed},\mathsf{k}) \in T_{\mathcal{H}_{\mathsf{key}}}$, then **return** k; ..... Case 1

if $\mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk},\mathsf{r}_{\mathsf{test}}) = \perp_{\mathsf{pk}}$:

$\mathsf{k} \xleftarrow{\$} \mathcal{K}$; $T_{\mathcal{H}_{\mathsf{key}}} \leftarrow T_{\mathcal{H}_{\mathsf{key}}} \cup \{(\mathsf{pk}||\mathsf{seed},\mathsf{k})\}$;

return k; ..... Case 2

if $\exists(\mathsf{pk}||\mathsf{r},\mathsf{seed}) \in T_{\mathcal{H}_{\mathsf{seed}}}$:

$(\mathsf{c},\mathsf{k}) \leftarrow \mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk},\mathsf{r})$;

$T_{\mathcal{H}_{\mathsf{key}}} \leftarrow T_{\mathcal{H}_{\mathsf{key}}} \cup \{(\mathsf{pk}||\mathsf{seed},\mathsf{k})\}$;

return k; ..... Case 3

if $\exists(\mathsf{pk}||\mathsf{seed},\mathsf{r}) \in T^*_{\mathsf{r}}$:

$(\mathsf{c},\mathsf{k}) \leftarrow \mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk},\mathsf{r})$;

$T_{\mathcal{H}_{\mathsf{key}}} \leftarrow T_{\mathcal{H}_{\mathsf{key}}} \cup \{(\mathsf{pk}||\mathsf{seed},\mathsf{k})\}$;

return k; ..... Case 4

for any $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2,\mathsf{c}) \in T_{\mathcal{P}}$ and $(\tilde{\mathsf{c}}_1||\mathsf{str},\mathsf{mask}) \in T_{\mathcal{H}_{\mathsf{mask}}}$ s.t. $\mathsf{seed} = \tilde{\mathsf{c}}_2 \oplus \mathsf{mask}$:

if $\exists x \in \mathbb{Z}_N$ s.t. $(\mathsf{pk} - x \cdot \diamondsuit = 0), (\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$:

if $\exists(\mathsf{sk},x) \in T_{\mathcal{H}_x}$ s.t. $\mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk},\mathsf{c}) \in \mathcal{K}$:

$\mathsf{k} \leftarrow \mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk},\mathsf{c})$;

$T_{\mathcal{H}_{\mathsf{key}}} \leftarrow T_{\mathcal{H}_{\mathsf{key}}} \cup \{(\mathsf{pk}||\mathsf{seed},\mathsf{k})\}$;

return k; ..... Case 5

if $\exists(\mathsf{sk},\mathsf{pk}) \in T^*_{\mathsf{sk}}$ s.t. $\mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk},\mathsf{c}) \in \mathcal{K}$:
$\mathsf{k} \leftarrow \mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk},\mathsf{c})$;
$T_{\mathcal{H}_{\mathsf{key}}} \leftarrow T_{\mathcal{H}_{\mathsf{key}}} \cup \{(\mathsf{pk}||\mathsf{seed},\mathsf{k})\}$;
return k; ..... Case 6

$\mathsf{r} \xleftarrow{\$} \{0,1\}^\lambda$; $(\mathsf{c},\mathsf{k}) \leftarrow \mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk},\mathsf{r})$;

$T_{\mathcal{H}_{\mathsf{key}}} \leftarrow T_{\mathcal{H}_{\mathsf{key}}} \cup \{(\mathsf{pk}||\mathsf{seed},\mathsf{k})\}$; $T^*_{\mathsf{r}} \leftarrow T^*_{\mathsf{r}} \cup \{(\mathsf{pk}||\mathsf{seed},\mathsf{r})\}$;

return k; ..... Case 7

$\underline{\mathcal{S}.\mathcal{P}(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2)}$:

if $\exists(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2,\mathsf{c}) \in T_{\mathcal{P}}$, then **return** c; ..... Case 1

if $\mathbf{KEM}.\mathsf{Encaps}(\tilde{\mathsf{c}}_1,\mathsf{r}_{\mathsf{test}}) = \perp_{\mathsf{pk}}$:
for attempt $= 1$ to λ do:
$\mathsf{c} \xleftarrow{\$} \{0,1\}^{\ell+\lambda}$;
if $\mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk}_{\mathsf{test}},\mathsf{c}) = \perp_{\mathsf{c}}$ **then break**;
$T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2,\mathsf{c})\}$;
return c; ..... Case 2

if $\exists(\mathsf{seed},y) \in T_{\mathcal{H}_y}$ s.t. $(\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$:
$\mathsf{mask} \leftarrow \mathsf{seed} \oplus \tilde{\mathsf{c}}_2$;
if $\exists(\tilde{\mathsf{c}}_1||\mathsf{str},\mathsf{mask}) \in T_{\mathcal{H}_{\mathsf{mask}}}$ with $(\mathsf{str} - y \cdot \mathsf{pk} = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$:
if $\exists(\mathsf{pk}||\mathsf{r},\mathsf{seed}) \in T_{\mathcal{H}_{\mathsf{seed}}}$: $(\mathsf{c},\mathsf{k}) \leftarrow \mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk},\mathsf{r})$;
$T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2,\mathsf{c})\}$;
return c; ..... Case 3
if $\exists(\mathsf{pk}||\mathsf{seed},\mathsf{r}) \in T^*_{\mathsf{r}}$: $(\mathsf{c},\mathsf{k}) \leftarrow \mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk},\mathsf{r})$;
$T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2,\mathsf{c})\}$;
return c; ..... Case 4
else: $\mathsf{r} \xleftarrow{\$} \{0,1\}^\lambda$; $(\mathsf{c},\mathsf{k}) \leftarrow \mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk},\mathsf{r})$;
$T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2,\mathsf{c})\}$; $T^*_{\mathsf{r}} \leftarrow T^*_{\mathsf{r}} \cup \{(\mathsf{pk}||\mathsf{seed},\mathsf{r})\}$;
return c; ..... Case 5

$\mathsf{sk} \xleftarrow{\$} \mathbb{Z}_N$; $\mathsf{pk} \leftarrow \mathbf{KEM}.\mathsf{KG}(\mathsf{sk})$;

$\mathsf{r} \xleftarrow{\$} \{0,1\}^\lambda$; $(\mathsf{c},\mathsf{k}) \leftarrow \mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk},\mathsf{r})$;

$T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2,\mathsf{c})\}$; $T^*_{\mathsf{c}} \leftarrow T^*_{\mathsf{c}} \cup \{(\mathsf{sk},\mathsf{r})\}$;

return c; ..... Case 6

$\underline{\mathcal{S}.\mathcal{P}^{-1}(\mathsf{c})}$:

if $\exists(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c}) \in T_{\mathcal{P}}$, then **return** $\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2$; ..... Case 7

if $\mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk}_{\mathsf{test}}, \mathsf{c}) = \perp_{\mathsf{c}}$:

for attempt $= 1$ to λ do:

$\tilde{\mathsf{c}}_1 \xleftarrow{\$} \{0,1\}^{\ell}$;

if $\mathbf{KEM}.\mathsf{Encaps}(\tilde{\mathsf{c}}_1, \mathsf{r}_{\mathsf{test}}) = \perp_{\mathsf{pk}}$ **then break**;

$\tilde{\mathsf{c}}_2 \xleftarrow{\$} \{0,1\}^{\lambda}$; $T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c})\}$;

return $\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2$; ..... Case 8

if $\exists(\mathsf{pk}||\mathsf{r}, \mathsf{seed}) \in T_{\mathcal{H}_{\mathsf{seed}}}$ s.t. $\mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r}) = (\mathsf{c}, *)$:

if $\exists(\mathsf{seed}, y) \in T_{\mathcal{H}_y}$:

if $\exists(\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$:

if $\exists(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask}) \in T_{\mathcal{H}_{\mathsf{mask}}}$ s.t. $(\mathsf{str} - y \cdot \mathsf{pk} = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$:

$\tilde{\mathsf{c}}_2 \leftarrow \mathsf{mask} \oplus \mathsf{seed}$; $T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c})\}$;

return $\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2$; ..... Case 9

else: $\tilde{\mathsf{c}}_2 \xleftarrow{\$} \{0,1\}^{\lambda}$; $T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c})\}$;

return $\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2$; ..... Case 10

else: $\mathsf{sk} \xleftarrow{\$} \mathbb{Z}_N$; $\tilde{\mathsf{c}}_1 \leftarrow \mathbf{KEM}.\mathsf{KG}(\mathsf{sk})$;

$T^*_{\mathsf{sk}} \leftarrow T^*_{\mathsf{sk}} \cup \{(\mathsf{sk}, \tilde{\mathsf{c}}_1)\}$; $T_{\mathsf{eq}} \leftarrow T_{\mathsf{eq}} \cup \{\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0\}$;

$\tilde{\mathsf{c}}_2 \xleftarrow{\$} \{0,1\}^{\lambda}$; $T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c})\}$;

return $\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2$; ..... Case 11

$\mathsf{sk} \xleftarrow{\$} \mathbb{Z}_N$; $\tilde{\mathsf{c}}_1 \leftarrow \mathbf{KEM}.\mathsf{KG}(\mathsf{sk})$;

$T^*_{\mathsf{sk}} \leftarrow T^*_{\mathsf{sk}} \cup \{(\mathsf{sk}, \tilde{\mathsf{c}}_1)\}$;

$\tilde{\mathsf{c}}_2 \xleftarrow{\$} \{0,1\}^{\lambda}$; $T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c})\}$;

return $\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2$; ..... Case 12

$\mathcal{S}.\mathcal{G}^{\mathsf{label}}(x)$:

if $\exists(\mathsf{str} - x \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$, then **return** str; ..... Case 1

else:

$\mathsf{sk} \xleftarrow{\$} \mathbb{Z}_N$; $\mathsf{str} \leftarrow \mathbf{KEM}.\mathsf{KG}(\mathsf{sk})$; $T^*_{\mathsf{sk}} \leftarrow T^*_{\mathsf{sk}} \cup \{(\mathsf{sk}, \mathsf{str})\}$;

$T_{\mathsf{eq}} \leftarrow T_{\mathsf{eq}} \cup \{\mathsf{str} - x \cdot \diamondsuit = 0\}$;

return str; ..... Case 2

$\mathcal{S}.\mathcal{G}^{\mathsf{add}}(\mathsf{str}_1, \mathsf{str}_2)$:

$\phi_1 \leftarrow \mathbf{KEM}.\mathsf{Encaps}(\mathsf{str}_1, \mathsf{r}_{\mathsf{test}})$;

$\phi_2 \leftarrow \mathbf{KEM}.\mathsf{Encaps}(\mathsf{str}_2, \mathsf{r}_{\mathsf{test}})$;

if $\phi_1 = \perp_{\mathsf{pk}} \vee \phi_2 = \perp_{\mathsf{pk}}$: **return** $\perp$; ..... Case 1

if $\exists(\mathsf{str} - \mathsf{str}_1 - \mathsf{str}_2 = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$, then **return** str; ..... Case 2

$\mathsf{sk} \xleftarrow{\$} \mathbb{Z}_N$; $\mathsf{str} \leftarrow \mathbf{KEM}.\mathsf{KG}(\mathsf{sk})$; $T^*_{\mathsf{sk}} \leftarrow T^*_{\mathsf{sk}} \cup \{(\mathsf{sk}, \mathsf{str})\}$;

$T_{\mathsf{eq}} \leftarrow T_{\mathsf{eq}} \cup \{\mathsf{str} - \mathsf{str}_1 - \mathsf{str}_2 = 0\}$;

return str. ..... Case 3

We first observe that if the differentiator $\mathcal{D}$ issues q queries to the honest and adversarial interfaces, then the simulator $\mathcal{S}$ makes at most $O(\lambda \cdot q)$ queries to the ideal KEM $\mathbf{KEM}$. This ensures that the simulator $\mathcal{S}$ described above is efficient.

Next, we establish that the view of $\mathcal{D}$ in the real world is indistinguishable from its view in the ideal world. To this end, we construct a sequence of indistinguishable hybrid games transitioning from the ideal game $\mathbf{Ideal}_{\mathcal{S}}^{\mathbf{KEM}}$ to the real game $\mathbf{Real}^{\Pi^{\mathcal{O}}_{\mathsf{KEM}}}$. Specifically,

1. **Ideal Game**. We begin with the ideal game $\mathbf{Ideal}_{\mathcal{S}}^{\mathbf{KEM}}$, in which $\mathcal{D}$ interacts with the honest interface $\mathbf{KEM}$ and the adversarial interface $\mathcal{S}^{\mathbf{KEM}}$.

2. **Game $\mathbf{G}_1$**. A relay algorithm $\mathcal{F}$ is introduced as an intermediary between $\mathcal{D}$ and ideal KEM. Here, $\mathcal{D}$ interacts with $\mathcal{F}^{\mathbf{KEM}}$ and $\mathcal{S}^{\mathbf{KEM}}$, where $\mathcal{F}$ simply forwards queries to $\mathbf{KEM}$ and returns the corresponding outputs.

3. **Game $\mathbf{G}_2$**. We refine the simulator by introducing several failure conditions on its query–response pairs. Whenever a failure condition is triggered, the modified simulator $\mathcal{S}_{\mathsf{fail}}$ aborts explicitly. In this game, $\mathcal{D}$ interacts with $\mathcal{F}^{\mathbf{KEM}}$ and $\mathcal{S}_{\mathsf{fail}}^{\mathbf{KEM}}$.

4. **Game $\mathbf{G}_3$**. The relay algorithm $\mathcal{F}$ is replaced with the construction Π_{KEM}, as defined earlier. The construction Π_{KEM} employs $\mathcal{S}_{\mathsf{fail}}$ in place of the ideal model. The central goal is to show that, except with negligible probability of triggering a failure condition (from Game 2), the responses generated by Π_{KEM} remain consistent with those of $\mathbf{KEM}$. In this game, $\mathcal{D}$ interacts with $\Pi_{\mathsf{KEM}}^{\mathcal{S}_{\mathsf{fail}}}$ and $\mathcal{S}_{\mathsf{fail}}^{\mathbf{KEM}}$.

5. **Game $\mathbf{G}_4$**. The simulator is further modified to sample its responses independently of the ideal KEM. Moreover, the new simulator $\mathcal{S}_{\mathsf{samp}}$ no longer checks for failure conditions. In this game, $\mathcal{D}$ interacts with $\Pi_{\mathsf{KEM}}^{\mathcal{S}_{\mathsf{samp}}}$ and $\mathcal{S}_{\mathsf{samp}}$.

6. **Real Game**. Finally, we arrive at the real game $\mathbf{Real}^{\Pi^{\mathcal{O}}_{\mathsf{KEM}}}$, in which the simulator $\mathcal{S}_{\mathsf{samp}}$ from Game 4 is replaced with the real oracle $\mathcal{O}$. Thus, $\mathcal{D}$ interacts with $\Pi_{\mathsf{KEM}}^{\mathcal{O}}$ and $\mathcal{O}$.

In what follows, we give a detailed description of each hybrid game and demonstrate that the statistical distance between any pair of consecutive games is negligible.

Ideal Game. In this game, the differentiator $\mathcal{D}$ interacts with the ideal KEM and the simulator $\mathcal{S}$, as defined previously. The game outputs 1 if and only if $\mathcal{D}$ outputs 1 after this interaction. Formally,

$$\Pr[\mathbf{Ideal}_{\mathcal{S}}^{\mathbf{KEM}}(\mathcal{D}) = 1] = \Pr[\mathcal{D}^{\mathbf{KEM}, \mathcal{S}^{\mathbf{KEM}}}(1^\lambda) = 1].$$

B.1 Transition from Ideal to $\mathbf{G}_1$

Game 1. This game is identical to the ideal game, except that the differentiator $\mathcal{D}$ interacts with a relay algorithm $\mathcal{F} = (\mathcal{F}.\mathsf{KG}, \mathcal{F}.\mathsf{Encaps}, \mathcal{F}.\mathsf{Decaps})$ rather than directly with the ideal KEM. The relay algorithm $\mathcal{F}$ forwards all queries to the ideal KEM and provides the following two interfaces:

- $\mathcal{F}.\mathsf{KG}(\mathsf{sk})$: query $\mathsf{pk} \leftarrow \mathbf{KEM}.\mathsf{KG}(\mathsf{sk})$ and return pk.
- $\mathcal{F}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$: query $(\mathsf{c}, \mathsf{k}) \leftarrow \mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$ and return (c, k).
- $\mathcal{F}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c})$: query $\mathsf{k} \leftarrow \mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c})$ and return k.

Since all queries issued by $\mathcal{D}$ are answered exactly as in the ideal game, its view in Game 1 is identical to that in Ideal Game. Defining $\mathbf{G}_1$ as Game 1, which outputs 1 if and only if $\mathcal{D}$ outputs 1 after interacting with $\mathcal{F}$ and $\mathcal{S}$, we obtain

$$\begin{aligned}\Pr[\mathbf{G}_1(\mathcal{D}) = 1] &= \Pr[\mathcal{D}^{\mathcal{F}^{\mathbf{KEM}}, \mathcal{S}^{\mathbf{KEM}}}(1^\lambda) = 1] \\ &= \Pr[\mathbf{Ideal}_{\mathcal{S}}^{\mathbf{KEM}}(\mathcal{D}) = 1].\end{aligned}$$

B.2 Transition from $\mathbf{G}_1$ to $\mathbf{G}_2$

Game 2. In this game, the simulator is modified to include conditional checks before returning partial responses. These checks are specifically designed to capture correlations among the simulator's outputs that could potentially be exploited by the differentiator $\mathcal{D}$. If the modified simulator $\mathcal{S}_{\mathsf{fail}}$ generates a response that triggers any of the failure conditions listed below, it explicitly aborts instead of returning that response.

In response to a $\mathcal{H}_x$ query with input sk, the simulator samples a value $x \in \mathbb{Z}_N$ uniformly as the candidate response. Let $\mathsf{pk} \leftarrow \mathbf{KEM}.\mathsf{KG}(\mathsf{sk})$. The simulator then checks the following failure conditions:

- $\mathsf{C}_{1\text{-}\mathcal{H}_x}(\mathsf{sk}, \mathsf{pk})$: The pair $(\mathsf{sk}, \mathsf{pk})$ already appears in T^*_{sk}.
- $\mathsf{C}_{2\text{-}\mathcal{H}_x}(\mathsf{sk})$: The value sk already appears in T^*_{c}.
- $\mathsf{C}_{3\text{-}\mathcal{H}_x}(\mathsf{pk}, x)$: There exists $(\mathsf{str} - x \cdot \diamondsuit = 0)$ in the span of T_{eq} with $\mathsf{str} \neq \mathsf{pk}$.
- $\mathsf{C}_{4\text{-}\mathcal{H}_x}(x)$: There exists $(\mathsf{str}_1 - x \cdot \mathsf{str}_2 = 0)$ in the span of T_{eq}.
- $\mathsf{C}_{5\text{-}\mathcal{H}_x}(\mathsf{pk}, x)$: There exists $(\mathsf{str}_1 - \alpha \cdot \mathsf{str}_2 - \beta \cdot \mathsf{pk} - \gamma \cdot \diamondsuit = 0)$ in the span of T_{eq} such that $\beta \cdot x + \gamma = 0$.
- $\mathsf{C}_{6\text{-}\mathcal{H}_x}(\mathsf{pk}, x)$: There exists $(\mathsf{str} - \alpha \cdot \mathsf{pk} - \beta \cdot \diamondsuit = 0)$ in the span of T_{eq} such that condition $\mathsf{C}_{3\text{-}\mathcal{H}_x}(\mathsf{str}, \alpha \cdot x + \beta)$ holds.
- $\mathsf{C}_{7\text{-}\mathcal{H}_x}(\mathsf{pk}, x)$: There exists $(\mathsf{str} - \alpha \cdot \mathsf{pk} - \beta \cdot \diamondsuit = 0)$ in the span of T_{eq} such that condition $\mathsf{C}_{4\text{-}\mathcal{H}_x}(\alpha \cdot x + \beta)$ holds.
- $\mathsf{C}_{8\text{-}\mathcal{H}_x}(\mathsf{pk}, x)$: There exists $(\mathsf{str} - \alpha \cdot \mathsf{pk} - \beta \cdot \diamondsuit = 0)$ in the span of T_{eq} such that the value $(\alpha \cdot x + \beta)$ already appears in $T_{\mathcal{H}_y}$.
- $\mathsf{C}_{9\text{-}\mathcal{H}_x}(\mathsf{pk}, x)$: There exists $(\mathsf{str} - \alpha \cdot \mathsf{pk} - \beta \cdot \diamondsuit = 0)$ in the span of T_{eq} such that condition $\mathsf{C}_{5\text{-}\mathcal{H}_x}(\mathsf{str}, \alpha \cdot x + \beta)$ holds.

In response to a $\mathcal{H}_{\mathsf{seed}}$ query with input pk and r, if the response is not already determined by the relationships among other interfaces, the simulator samples seed uniformly at random as the candidate response, and then checks the following failure conditions:

- $\mathsf{C}_{1\text{-}\mathcal{H}_{\mathsf{seed}}}(\mathsf{r})$: The value r already appears in T^*_{r} or T^*_{c}.
- $\mathsf{C}_{2\text{-}\mathcal{H}_{\mathsf{seed}}}(\mathsf{seed})$: The string seed already appears in $T_{\mathcal{H}_{\mathsf{seed}}}$, $T_{\mathcal{H}_{\mathsf{key}}}$ or T^*_{r}.

In response to a $\mathcal{H}_y$ query on input seed, the simulator samples $y \in \mathbb{Z}_N$ uniformly at random as the candidate response. If there exists $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c}) \in T_{\mathcal{P}}$ associated with seed, the simulator additionally inserts the equation $(\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0)$ into T_{eq}. It then checks the following failure conditions:

- $\mathsf{C}_{1\text{-}\mathcal{H}_y}(y)$: The value y already appears in $T_{\mathcal{H}_y}$.
- $\mathsf{C}_{2\text{-}\mathcal{H}_y}(\tilde{\mathsf{c}}_1, y)$: There exists $(\mathsf{str} - y \cdot \diamondsuit = 0)$ in the span of T_{eq} with $\tilde{\mathsf{c}}_1 \neq \mathsf{str}$.
- $\mathsf{C}_{3\text{-}\mathcal{H}_y}(y)$: There exists $(\mathsf{str}_1 - y \cdot \mathsf{str}_2 = 0)$ in the span of T_{eq}.
- $\mathsf{C}_{4\text{-}\mathcal{H}_y}(\tilde{\mathsf{c}}_1, y)$: There exists $(\mathsf{str}_1 - \alpha \cdot \mathsf{str}_2 - \beta \cdot \tilde{\mathsf{c}}_1 - \gamma \cdot \diamondsuit = 0)$ in the span of T_{eq} such that $\beta \cdot y + \gamma = 0$.
- $\mathsf{C}_{5\text{-}\mathcal{H}_y}(\tilde{\mathsf{c}}_1, y)$: There exists $(\mathsf{str} - \alpha \cdot \tilde{\mathsf{c}}_1 - \beta \cdot \diamondsuit = 0)$ in the span of T_{eq} such that condition $\mathsf{C}_{1\text{-}\mathcal{H}_y}(\alpha \cdot y + \beta)$ holds.
- $\mathsf{C}_{6\text{-}\mathcal{H}_y}(\tilde{\mathsf{c}}_1, y)$: There exists $(\mathsf{str} - \alpha \cdot \tilde{\mathsf{c}}_1 - \beta \cdot \diamondsuit = 0)$ in the span of T_{eq} such that condition $\mathsf{C}_{2\text{-}\mathcal{H}_y}(\mathsf{str}, \alpha \cdot y + \beta)$ holds.
- $\mathsf{C}_{7\text{-}\mathcal{H}_y}(\tilde{\mathsf{c}}_1, y)$: There exists $(\mathsf{str} - \alpha \cdot \tilde{\mathsf{c}}_1 - \beta \cdot \diamondsuit = 0)$ in the span of T_{eq} such that condition $\mathsf{C}_{3\text{-}\mathcal{H}_y}(\alpha \cdot y + \beta)$ holds.

In response to a $\mathcal{H}_{\mathsf{mask}}$ query with input $\tilde{\mathsf{c}}_1$ and str, if the response is not already determined by the relationships among other interfaces, the simulator samples mask uniformly at random as the candidate response, and then checks the following failure conditions:

- $\mathsf{C}_{1\text{-}\mathcal{H}_{\mathsf{mask}}}(\mathsf{mask})$: The string mask already appears in $T_{\mathcal{H}_{\mathsf{mask}}}$.
- $\mathsf{C}_{2\text{-}\mathcal{H}_{\mathsf{mask}}}(\mathsf{mask})$: There exist seed in $T_{\mathcal{H}_{\mathsf{seed}}}$, $T_{\mathcal{H}_y}$, or $T_{\mathcal{H}_{\mathsf{key}}}$ and $\tilde{\mathsf{c}}_2$ in $T_{\mathcal{P}}$ such that $\mathsf{seed} \oplus \mathsf{mask} = \tilde{\mathsf{c}}_2$.

In response to a $\mathcal{H}_{\mathsf{key}}$ query with input $(\mathsf{pk}, \mathsf{seed})$, if pk is valid but the response is not already determined by the relationships among other interfaces, the simulator generates a fresh k by invoking $\mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$, where r is sampled uniformly at random, and then checks the following failure condition:

- $\mathsf{C}_{1\text{-}\mathcal{H}_{\mathsf{key}}}(\mathsf{r})$: The value r already appears in $T_{\mathcal{H}_{\mathsf{seed}}}$, T^*_{r}, or T^*_{c}.

In response to a $\mathcal{P}$ query with input $\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2$, the simulator first checks the validity of $\tilde{\mathsf{c}}_1$ using $\mathbf{KEM}.\mathsf{Encaps}$. If $\tilde{\mathsf{c}}_1$ is invalid, the simulator samples an invalid ciphertext c as the candidate response and then checks the following failure conditions:

- $\mathsf{C}_{1\text{-}\mathcal{P}}(\mathsf{c})$: The generated c already appears in $T_{\mathcal{P}}$.
- $\mathsf{C}_{2\text{-}\mathcal{P}}(\mathsf{c})$: The generated c is valid, i.e., $\mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk}_{\mathsf{test}}, \mathsf{c}) \neq \perp_{\mathsf{c}}$.

For a valid $\tilde{c}_1$, if there exist seed and mask with $\tilde{c}_1 = \mathsf{seed} \oplus \mathsf{mask}$ such that the specific correlations are satisfied and corresponding to a pk. Then the response should be a ciphertext under pk. If there is no corresponding randomness r, the simulator uniformly samples r, generates response c by querying $\mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$, and then checks the following failure conditions:

- $\mathsf{C}_{3\text{-}\mathcal{P}}(\mathsf{r})$: The value r already appears in $T_{\mathcal{H}_{\mathsf{seed}}}$, T^*_{r}, or T^*_{c}.

Otherwise, the simulator generates a fresh c by sampling sk and r uniformly, and then checks the failure conditions $\mathsf{C}_{3\text{-}\mathcal{P}}$ and:

- $\mathsf{C}_{4\text{-}\mathcal{P}}(\mathsf{sk})$: The value sk already appears in $T_{\mathcal{H}_x}$.

In response to a $\mathcal{P}^{-1}$ query with input c, the simulator first checks the validity of c using $\mathbf{KEM}.\mathsf{Decaps}$. If c is invalid, the simulator samples an invalid $\tilde{c}_1$ and $\tilde{c}_2$ as the candidate response and then checks the following failure conditions:

- $\mathsf{C}_{1\text{-}\mathcal{P}^{-1}}(\tilde{c}_2)$: The generated $\tilde{c}_2$ already appears in $T_{\mathcal{P}}$.

- $\mathsf{C}_{2\text{-}\mathcal{P}^{-1}}(\tilde{c}_1)$: The generated $\tilde{c}_1$ is valid, i.e., $\mathbf{KEM}.\mathsf{Encaps}(\tilde{c}_1, \mathsf{r}_{\mathsf{test}}) \neq \perp_{\mathsf{pk}}$.

For a valid c, if only $\tilde{c}_1$ is already determined by the relationships among other interfaces, the simulator uniformly samples $\tilde{c}_2$ and then checks the failure conditions $\mathsf{C}_{1\text{-}\mathcal{P}^{-1}}(\tilde{c}_2)$ and:

- $\mathsf{C}_{3\text{-}\mathcal{P}^{-1}}(\tilde{c}_2)$: There exist seed in $T_{\mathcal{H}_{\mathsf{seed}}}$, $T_{\mathcal{H}_y}$, or $T_{\mathcal{H}_{\mathsf{key}}}$ and mask in $T_{\mathcal{H}_{\mathsf{mask}}}$ such that $\mathsf{seed} \oplus \mathsf{mask} = \tilde{c}_2$.

If both $\tilde{c}_1$ and $\tilde{c}_2$ are not determined, the simulator generates fresh $\tilde{c}_1$ and $\tilde{c}_2$ with $\tilde{c}_1 \leftarrow \mathbf{KEM}.\mathsf{KG}(\mathsf{sk})$ where sk is sampled uniformly at random. And then checks the failure conditions $\mathsf{C}_{1\text{-}\mathcal{P}^{-1}}(\tilde{c}_2)$, $\mathsf{C}_{3\text{-}\mathcal{P}^{-1}}(\tilde{c}_2)$, and:

- $\mathsf{C}_{4\text{-}\mathcal{P}^{-1}}(\tilde{c}_1)$: The generated $\tilde{c}_1$ already appears in T_{eq}, $T_{\mathcal{P}}$, or $T_{\mathcal{H}_{\mathsf{mask}}}$.

- $\mathsf{C}_{5\text{-}\mathcal{P}^{-1}}(\mathsf{sk})$: The value sk already appears in $T_{\mathcal{H}_x}$.

- $\mathsf{C}_{6\text{-}\mathcal{P}^{-1}}(\mathsf{sk})$: The exist $(\tilde{c}_1 || \tilde{c}_2, \mathsf{c})$ in $T_{\mathcal{P}}$ with $\mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c}) \in \mathcal{K}$.

In response to a $\mathcal{G}$ query for which no corresponding entry exists in $\mathsf{Span}(T_{\mathsf{eq}})$, the simulator generates a fresh public key by uniformly sampling sk as the candidate response. Let $\mathsf{str} \leftarrow \mathbf{KEM}.\mathsf{KG}(\mathsf{sk})$. The simulator then checks the following failure conditions:

- $\mathsf{C}_{1\text{-}\mathcal{G}}(\mathsf{str})$: The generated str already appears in T_{eq}, $T_{\mathcal{P}}$, or $T_{\mathcal{H}_{\mathsf{mask}}}$.

- $\mathsf{C}_{2\text{-}\mathcal{G}}(\mathsf{sk})$: The value sk already appears in $T_{\mathcal{H}_x}$.

Intuitively, each of the conditions above corresponds to an unwanted collision or algebraic relation between a freshly sampled element and information already recorded in the simulator's tables; each such event can be bounded by a negligible probability. A formal probabilistic analysis is deferred to Appendix B.7.

Lemma B.1. $\Pr[\text{Failure Conditions Hold}] \leq O\left(q^4/N + q^3/2^\lambda\right)$.

The differentiator can distinguish between Game 1 and Game 2 only when the following two events occur:

- In Game 1, the simulator $\mathcal{S}$ does not fail, even when its response satisfies one of the failure conditions above;

- In Game 2, the simulator $\mathcal{S}_{\mathsf{fail}}$ does fail.

Let $\mathbf{G}_2$ denoted Game 2, which outputs 1 if and only if $\mathcal{D}$ outputs 1 after interacting with $\mathcal{F}$ and $\mathcal{S}_{\mathsf{fail}}$. Formally,

$$\Pr[\mathbf{G}_2(\mathcal{D})] = \Pr[\mathcal{D}^{\mathcal{F}^{\mathbf{KEM}},\mathcal{S}_{\mathsf{fail}}^{\mathbf{KEM}}}(1^\lambda) = 1].$$

Then

$$\begin{aligned}|\Pr[\mathbf{G}_2(\mathcal{D}) = 1] - \Pr[\mathbf{G}_1(\mathcal{D}) = 1]| &\leq 2 \cdot \Pr[\text{Failure Conditions Hold}] \\ &\leq O\left(\frac{q^4}{N} + \frac{q^3}{2^\lambda}\right).\end{aligned}$$

B.3 Properties under the absence of failure conditions

Before describing Game 3, we state several propositions that record basic observations and will be useful in the proofs of the subsequent lemmas.

Proposition B.1. *Suppose that conditions* $\mathsf{C}_{1,2\text{-}\mathcal{H}_x}$, $\mathsf{C}_{1\text{-}\mathcal{H}_{\mathsf{seed}}}$, $\mathsf{C}_{1\text{-}\mathcal{H}_{\mathsf{key}}}$, $\mathsf{C}_{3,4\text{-}\mathcal{P}}$, $\mathsf{C}_{4,5\text{-}\mathcal{P}^{-1}}$, *and* $\mathsf{C}_{1,2\text{-}\mathcal{G}}$ *do not hold. Then for every equation* $(\mathsf{str} - x \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$, *there must exist either a pair* (sk, x) *in* $T_{\mathcal{H}_x}$ *or a pair* $(\mathsf{sk}, \mathsf{str})$ *in* T^*_{sk} *such that* $\mathsf{str} = \mathbf{KEM}.\mathsf{KG}(\mathsf{sk})$.

Proof. As described for the simulator, the equations recorded in T_{eq} can be classified into five types:

- **Type 1:** An equation of the form $(\mathsf{str} - x \cdot \diamondsuit = 0)$, introduced when processing a $\mathcal{G}^{\mathsf{label}}$ query, where $\mathsf{str} = \mathbf{KEM}.\mathsf{KG}(\mathsf{sk})$ with sk sampled uniformly at random from $\mathbb{Z}_N$.
- **Type 2:** An equation of the form $(\mathsf{str} - \mathsf{str}_1 - \mathsf{str}_2 = 0)$, introduced when processing a $\mathcal{G}^{\mathsf{add}}$ query, where $\mathsf{str} = \mathbf{KEM}.\mathsf{KG}(\mathsf{sk})$ with sk sampled uniformly at random from $\mathbb{Z}_N$.
- **Type 3:** An equation of the form $(\mathsf{pk} - x \cdot \diamondsuit = 0)$, introduced when processing a $\mathcal{H}_x$ query, where x is sampled uniformly at random from $\mathbb{Z}_N$.
- **Type 4:** An equation of the form $(\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0)$, introduced when processing a $\mathcal{H}_y$ query, where y is sampled uniformly at random from $\mathbb{Z}_N$.
- **Type 5:** An equation of the form $(\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0)$, introduced when processing a $\mathcal{P}^{-1}$ query, where $\tilde{\mathsf{c}}_1 = \mathbf{KEM}.\mathsf{KG}(\mathsf{sk})$ with sk sampled uniformly at random from $\mathbb{Z}_N$.

For an equation Eq of the form $(\mathsf{str} - x \cdot \diamondsuit = 0)$, corresponding to Type 1 (with Type 5 being analogous), the pair $(\mathsf{sk}, \mathsf{str})$ with $\mathsf{str} = \mathbf{KEM}.\mathsf{KG}(\mathsf{sk})$ is recorded in T^*_{sk}. Provided that str does not already appear in T_{eq} (i.e., conditions $\mathsf{C}_{1\text{-}\mathcal{G}}$ and $\mathsf{C}_{4\text{-}\mathcal{P}^{-1}}$ do not hold), this insertion does not assign discrete logarithms to any other group elements.

For an equation Eq of the form $(\mathsf{str} - \mathsf{str}_1 - \mathsf{str}_2 = 0)$, corresponding to Type 2, if both str_1 and str_2 already have discrete logarithms, the situation is analogous to Type 1. Otherwise, provided that condition $\mathsf{C}_{1\text{-}\mathcal{G}}$ does not hold, inserting Eq does not assign new discrete logarithms to the group elements in T_{eq}.

For an equation Eq of the form $(\mathsf{pk} - x \cdot \diamondsuit = 0)$, corresponding to Type 3, the pair (sk, x) is recorded in $T_{\mathcal{H}_x}$ with $\mathsf{str} = \mathbf{KEM}.\mathsf{KG}(\mathsf{sk})$. Inserting Eq may also assign discrete logarithms to other group elements. If this occurs, and provided that no sk appears simultaneously in both $T_{\mathcal{H}_x}$ and T^*_{sk} (i.e., conditions $\mathsf{C}_{1\text{-}\mathcal{H}_x}$, $\mathsf{C}_{5\text{-}\mathcal{P}^{-1}}$, and $\mathsf{C}_{2\text{-}\mathcal{G}}$ do not hold), then there must exist an equation

$$\left(\widetilde{\mathsf{str}} - \widetilde{\mathsf{str}}_1 - \widetilde{\mathsf{str}}_2 = 0\right) \in T_{\mathsf{eq}}$$

such that $\widetilde{\mathsf{str}}_1 = \mathsf{pk}$ and $(\widetilde{\mathsf{str}}_2 - \tilde{x}_2 \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$. In particular, inserting Eq yields

$$\left(\widetilde{\mathsf{str}} - (x + \tilde{x}_2) \cdot \diamondsuit = 0\right) \in \mathsf{Span}(T_{\mathsf{eq}}),$$

so that $\widetilde{\mathsf{str}}$ acquires a discrete logarithm and $(\widetilde{\mathsf{sk}}, \widetilde{\mathsf{str}})$ is added to T^*_{sk}. Moreover, $\widetilde{\mathsf{str}}$ may in turn enable further group elements to acquire discrete logarithms in the same manner, with each such element likewise associated with a private key recorded in T^*_{sk}.

For an equation Eq of the form $(\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0)$, corresponding to Type 4, there must exist a pair $(\tilde{\mathsf{c}}_1 || \tilde{\mathsf{c}}_2, \mathsf{c})$ in $T_{\mathcal{P}}$ with c corresponding to the input seed of the $\mathcal{H}_y$ query. Such a pair cannot originate from a $\mathcal{P}$ query; otherwise, the query-response tuple (seed, y) must already be recorded in $T_{\mathcal{H}_y}$, or there exists an r that appears simultaneously in both $T_{\mathcal{H}_{\mathsf{seed}}}$ and T^*_{c} (i.e., conditions $\mathsf{C}_{1\text{-}\mathcal{H}_{\mathsf{seed}}}$, $\mathsf{C}_{1\text{-}\mathcal{H}_{\mathsf{key}}}$ or $\mathsf{C}_{3\text{-}\mathcal{P}}$ are triggered), or an sk that appears simultaneously in both T^*_{c} and $T_{\mathcal{H}_x}$ or T^*_{sk} (i.e., conditions $\mathsf{C}_{2\text{-}\mathcal{H}_x}$ or $\mathsf{C}_{4\text{-}\mathcal{P}}$ are triggered). If $(\tilde{\mathsf{c}}_1 || \tilde{\mathsf{c}}_2, \mathsf{c})$ is inserted via a $\mathcal{P}^{-1}$ query, then $\tilde{\mathsf{c}}_1$ must have been generated by uniformly sampling sk, in which case the simulator simultaneously records $(\mathsf{sk}, \tilde{\mathsf{c}}_1)$ in T^*_{sk}. Moreover, if the insertion of Eq also assigns discrete logarithms to other group elements, the situation is analogous to that described for Type 3. □

□

Proposition B.2. *Suppose that conditions* $\mathsf{C}_{1,2,3,6\text{-}\mathcal{H}_x}$, $\mathsf{C}_{1\text{-}\mathcal{H}_{\mathsf{seed}}}$, $\mathsf{C}_{2,6\text{-}\mathcal{H}_y}$, $\mathsf{C}_{1\text{-}\mathcal{H}_{\mathsf{key}}}$, $\mathsf{C}_{3,4\text{-}\mathcal{P}}$, $\mathsf{C}_{4,5\text{-}\mathcal{P}^{-1}}$, *and* $\mathsf{C}_{1,2\text{-}\mathcal{G}}$ *do not hold. Then* T_{eq} *simulates a proper view of the generic group.*

Proof. A proper simulation of the generic group with respect to T_{eq} requires that the following two bad events do not occur:

- DLog-Coll: There exist $(\mathsf{str}_1 - x \cdot \diamondsuit = 0)$ and $(\mathsf{str}_2 - x \cdot \diamondsuit = 0)$ in $\mathsf{Span}(T_{\mathsf{eq}})$ with $\mathsf{str}_1 \neq \mathsf{str}_2$.
- Multi-DLog: There exist $(\mathsf{str} - x_1 \cdot \diamondsuit = 0)$ and $(\mathsf{str} - x_2 \cdot \diamondsuit = 0)$ in $\mathsf{Span}(T_{\mathsf{eq}})$ with $x_1 \neq x_2$.

Write $\mathsf{Cons}(T_{\mathsf{eq}})$ to denote that neither bad event occurs for T_{eq}. Clearly $\mathsf{Cons}(T_{\mathsf{eq}})$ holds when T_{eq} is empty or contains a single equation. Assume $\mathsf{Cons}(T_{\mathsf{eq}})$ holds prior to inserting a new equation Eq. We show that, provided no failure condition is triggered, $\mathsf{Cons}(T_{\mathsf{eq}} \cup \mathsf{Eq})$ still holds.

If Eq is generated by $\mathcal{P}^{-1}$ or $\mathcal{G}^{\mathsf{label}}$ in the form $(\mathsf{str} - x \cdot \diamondsuit = 0)$, or by $\mathcal{G}^{\mathsf{add}}$ in the form $(\mathsf{str} - \mathsf{str}_1 - \mathsf{str}_2 = 0)$, then, as long as the newly generated str does not already appear in T_{eq} (i.e., conditions $\mathsf{C}_{1\text{-}\mathcal{G}}$ and $\mathsf{C}_{4\text{-}\mathcal{P}^{-1}}$ are not met)—the insertion of Eq introduces no new algebraic relations among the existing group elements recorded in T_{eq}. Hence neither DLog-Coll nor Multi-DLog can arise in this case.

By contrast, if Eq is inserted by $\mathcal{H}_x(\mathsf{sk})$ as $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ with x sampled uniformly at random, then the insertion may directly cause either DLog-Coll or Multi-DLog. In the former event the failure condition $\mathsf{C}_{3\text{-}\mathcal{H}_x}$ is triggered. In the latter event Proposition B.1 implies the existence of $(\mathsf{sk}, \mathsf{pk})$ in T^*_{sk}; because sk is recorded in $T_{\mathcal{H}_x}$ as well, condition $\mathsf{C}_{1\text{-}\mathcal{H}_x}$ is forced to hold.

Furthermore, the insertion of Eq may also endow additional group elements with random discrete logarithms. As noted in Proposition B.1, there must exist an equation

$$(\mathsf{str} - \alpha \cdot \mathsf{pk} - \beta = 0) \in \mathsf{Span}(T_{\mathsf{eq}}),$$

so that the insertion of Eq induces

$$(\mathsf{str} - (\alpha \cdot x + \beta) \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}}).$$

The random discrete logarithm thus assigned to str can collide with existing discrete logarithms; such a collision triggers the failure condition $\mathsf{C}_{6\text{-}\mathcal{H}_x}$.

Similarly, if Eq is inserted by $\mathcal{H}_y(\mathsf{r})$ in the form $(\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0)$, as noted in Proposition B.1, there must exist a pair $(\tilde{\mathsf{c}}_1 || \tilde{\mathsf{c}}_2, \mathsf{c}) \in T_{\mathcal{P}}$ that was produced by a $\mathcal{P}^{-1}$ query sampling a valid $\tilde{\mathsf{c}}_1$ uniformly. Provided that $\mathsf{C}_{1\text{-}\mathcal{H}_x}$, $\mathsf{C}_{4\text{-}\mathcal{P}^{-1}}$ and $\mathsf{C}_{1\text{-}\mathcal{G}}$ do not hold, $\tilde{\mathsf{c}}_1$ has no preexisting discrete logarithm in T_{eq} prior to the insertion of Eq. Inserting Eq may directly induce a DLog-Coll (triggering $\mathsf{C}_{2\text{-}\mathcal{H}_y}$), or may assign random discrete logarithms to other group elements which collide with existing ones (triggering $\mathsf{C}_{6\text{-}\mathcal{H}_y}$).

In all cases, therefore, provided no failure condition is triggered, T_{eq} simulates a proper view of the generic group. □ □

Proposition B.3. *Suppose that none of the failure conditions holds. Then there do not exist two distinct pairs* $(\tilde{\mathsf{c}}_1 || \tilde{\mathsf{c}}_2, \mathsf{c})$ *and* $(\tilde{\mathsf{c}}'_1 || \tilde{\mathsf{c}}'_2, \mathsf{c})$ *in* $T_{\mathcal{P}}$ *with* $\tilde{\mathsf{c}}_1 || \tilde{\mathsf{c}}_2 \neq \tilde{\mathsf{c}}'_1 || \tilde{\mathsf{c}}'_2$.

Proof. For any pair $(\tilde{c}_1||\tilde{c}_2, c)$ inserted by the $\mathcal{P}(\tilde{c}_1||\tilde{c}_2)$ query, rather than Case 1 where the corresponding response is already recorded in $T_{\mathcal{P}}$, the simulator proceeds according to one of the remaining three cases, which correspond to those already specified in the simulator's description.

Case 2. If $\tilde{c}_1$ is invalid, the simulator repeatedly samples ciphertexts c uniformly at random until an invalid one is obtained, making at most λ attempts.

Case 3, 4. Suppose there exist pk and $seed$ such that $seed$ corresponds to $\tilde{c}_1$, and pk together with $seed$ determine a $mask$ with $\tilde{c}_2 = seed \oplus mask$. If the corresponding randomness r is already recorded in $T_{\mathcal{H}_{seed}}$ or T^*_r, then the simulator returns the ciphertext c by querying $\mathbf{KEM}.\mathsf{Encaps}(pk, r)$.

Case 5. Under the same pk and $seed$ as in Case 3, 4, if no corresponding randomness r exists, the simulator samples r uniformly at random, generates c via $\mathbf{KEM}.\mathsf{Encaps}(pk, r)$, and records $(pk||seed, r)$ in T^*_r.

Case 6. If none of the above cases applies, the simulator samples sk and r uniformly at random to generate a ciphertext c. It then records (sk, r) in T^*_c.

If the pair $(\tilde{c}_1||\tilde{c}_2, c)$ inserted by the $\mathcal{P}^{-1}(c)$ query, rather than Case 7 where the corresponding response is already recorded in $T_{\mathcal{P}}$, the simulator proceeds according to one of the remaining five cases, which correspond to those already specified in the simulator's description.

Case 8. If c is invalid, the simulator repeatedly samples public key $\tilde{c}_1$ uniformly at random until an invalid one is obtained, making at most λ attempts.

Case 9. If there exist $(pk||r, seed) \in T_{\mathcal{H}_{seed}}$ such that $\mathbf{KEM}.\mathsf{Encaps}(pk, r) = (c, *)$, and if $\tilde{c}_1$ together with a corresponding $mask$ are defined with respect to $(pk, seed)$, then the simulator responds with $\tilde{c}_1||(seed \oplus mask)$.

Case 10. Given a determined $\tilde{c}_1$ as in Case 9, if no corresponding $\tilde{c}_2$ exists, the simulator samples $\tilde{c}_2$ uniformly at random.

Case 11, 12. If none of the above cases applies, the simulator samples both valid $\tilde{c}_1$ and $\tilde{c}_2$ uniformly at random and returns them.

Suppose there exist two distinct pairs $(\tilde{c}_1||\tilde{c}_2, c)$ and $(\tilde{c}'_1||\tilde{c}'_2, c)$ in $T_{\mathcal{P}}$. Then $(\tilde{c}_1||\tilde{c}_2, c)$ must have been inserted first, either by a $\mathcal{P}$ or $\mathcal{P}^{-1}$ query, and only afterwards $(\tilde{c}'_1||\tilde{c}'_2, c)$ must have inserted by a $\mathcal{P}$ query. Any such collision necessarily implies that one of the failure conditions is triggered. Specifically,

- If $(\tilde{c}_1||\tilde{c}_2, c)$ is first inserted by either $\mathcal{P}$ or $\mathcal{P}^{-1}$, and then $(\tilde{c}'_1||\tilde{c}'_2, c)$ is inserted by Case 2, the randomly sampled c by Case 2 must collide with an existing ciphertext in $T_{\mathcal{P}}$ or being valid (i.e., conditions $\mathsf{C}_{1\text{-}\mathcal{P}}$ or $\mathsf{C}_{2\text{-}\mathcal{P}}$ are triggered).
- If $(\tilde{c}_1||\tilde{c}_2, c)$ is first inserted by Case 2, and later $(\tilde{c}'_1||\tilde{c}'_2, c)$ is inserted by Cases 3, 4, 5, or 6, this indicates that in Case 2 the simulator still produced a valid ciphertext even after λ iterations (i.e., condition $\mathsf{C}_{2\text{-}\mathcal{P}}$ is triggered).
- If $(\tilde{c}_1||\tilde{c}_2, c)$ is first inserted by Cases 3, 9, 10, or 11, and later $(\tilde{c}'_1||\tilde{c}'_2, c)$ is inserted by Case 3, then—since each random oracle query produces a unique response—an inconsistency must arise in T_{eq}, contradicting Proposition B.2.
- If $(\tilde{c}_1||\tilde{c}_2, c)$ is first inserted by Cases 4, 5, or 6, and later $(\tilde{c}'_1||\tilde{c}'_2, c)$ is inserted by Case 3, there must exist one r that simultaneously appears in both $T_{\mathcal{H}_{seed}}$ and T^*_r or T^*_c (i.e., conditions $\mathsf{C}_{1\text{-}\mathcal{H}_{seed}}$, $\mathsf{C}_{1\text{-}\mathcal{H}_{key}}$, or $\mathsf{C}_{3\text{-}\mathcal{P}}$ are triggered).
- If $(\tilde{c}_1||\tilde{c}_2, c)$ is first inserted by Cases 3, 9, ,10, or 11 and later $(\tilde{c}'_1||\tilde{c}'_2, c)$ is inserted by Cases 4, 5, or 6, conditions $\mathsf{C}_{1\text{-}\mathcal{H}_{seed}}$, $\mathsf{C}_{1\text{-}\mathcal{H}_{key}}$, or $\mathsf{C}_{3\text{-}\mathcal{P}}$ are triggered as well.
- If $(\tilde{c}_1||\tilde{c}_2, c)$ is first inserted by Case 4, and later $(\tilde{c}'_1||\tilde{c}'_2, c)$ is inserted by Case 4, then—since each random oracle query produces a unique response—an inconsistency must arise in T_{eq}, contradicting Proposition B.2.

- If $(\tilde{c}_1||\tilde{c}_2, c)$ is first inserted by Case 5, and later $(\tilde{c}'_1||\tilde{c}'_2, c)$ is inserted by Case 4, then—provided no r-collision in T^*_r-an inconsistency must arise in T_{eq}, contradicting Proposition B.2.
- If $(\tilde{c}_1||\tilde{c}_2, c)$ is first inserted by Case 6, and later $(\tilde{c}'_1||\tilde{c}'_2, c)$ is inserted by Case 4, there must exist one r that simultaneously appears in both T^*_r and T^*_c (i.e., conditions $C_{1\text{-}\mathcal{H}_{key}}$, or $C_{3\text{-}\mathcal{P}}$ are triggered).
- If $(\tilde{c}_1||\tilde{c}_2, c)$ is first inserted by Cases 4, 5, or 6, and later $(\tilde{c}'_1||\tilde{c}'_2, c)$ is inserted by Case 5 or 6, There exists a collision on r either within T^*_r, within table T^*_c, or between T^*_r and T^*_c (i.e., conditions $C_{1\text{-}\mathcal{H}_{key}}$, or $C_{3\text{-}\mathcal{P}}$ are triggered).

□
□

Proposition B.4. *Suppose that none of the failure conditions holds. Then there do not exist two distinct pairs* $(\tilde{c}_1||\tilde{c}_2, c)$ *and* $(\tilde{c}_1||\tilde{c}_2, c')$ *in* $T_{\mathcal{P}}$ *with* $c \neq c'$.

Proof. Suppose there exist two distinct pairs $(\tilde{c}_1||\tilde{c}_2, c)$ and $(\tilde{c}_1||\tilde{c}_2, c')$ in $T_{\mathcal{P}}$. Then $(\tilde{c}_1||\tilde{c}_2, c)$ must have been inserted first, either by a $\mathcal{P}$ or $\mathcal{P}^{-1}$ query, and only afterwards $(\tilde{c}_1||\tilde{c}_2, c')$ must have inserted by a $\mathcal{P}^{-1}$ query. Any such collision necessarily implies that one of the failure conditions is triggered. Specifically,

- If $(\tilde{c}_1||\tilde{c}_2, c)$ is first inserted by either $\mathcal{P}$ or $\mathcal{P}^{-1}$, and then $(\tilde{c}_1||\tilde{c}_2, c')$ is inserted by Case 8, the randomly sampled $\tilde{c}_2$ by Case 8 must collide with an existing string in $T_{\mathcal{P}}$ or $\tilde{c}_1$ being valid (i.e., conditions $C_{1,2\text{-}\mathcal{P}^{-1}}$ are triggered).
- If $(\tilde{c}_1||\tilde{c}_2, c)$ is first inserted by Case 8, and later $(\tilde{c}_1||\tilde{c}_2, c')$ is inserted by Cases 9, 10, 11, or 12, this indicates that in Case 8 the simulator still produced a valid public key even after λ iterations (i.e., condition $C_{2\text{-}\mathcal{P}^{-1}}$ is triggered).
- If $(\tilde{c}_1||\tilde{c}_2, c)$ is first inserted by Cases 3 or 9, and later $(\tilde{c}_1||\tilde{c}_2, c')$ is inserted by Case 9, then, provided T_{eq} is consistent, there must exist either seed-collision in $T_{\mathcal{H}_{seed}}$ or y-collision in $T_{\mathcal{H}_y}$ (i.e., conditions $C_{2\text{-}\mathcal{H}_{seed}}$ or $C_{1\text{-}\mathcal{H}_y}$ are triggered).
- If $(\tilde{c}_1||\tilde{c}_2, c)$ is first inserted by either $\mathcal{P}$ or $\mathcal{P}^{-1}$, and then $(\tilde{c}_1||\tilde{c}_2, c')$ is inserted by Cases 10, 11, or 12, the randomly sampled $\tilde{c}_2$ by $\mathcal{P}^{-1}$ must collide with an existing string in $T_{\mathcal{P}}$ (i.e., condition $C_{1\text{-}\mathcal{P}^{-1}}$ is triggered).
- If $(\tilde{c}_1||\tilde{c}_2, c)$ is first inserted by Case 6, and later $(\tilde{c}_1||\tilde{c}_2, c')$ is inserted by Case 9. When inserting $(\tilde{c}_1||\tilde{c}_2, c')$ there already exists $(\tilde{c}_1||\mathsf{str}, \mathsf{mask})$ in $T_{\mathcal{H}_{mask}}$ with mask sampled uniformly, and it holds that there exists $\tilde{c}_2$ in $T_{\mathcal{P}}$ and seed in $T_{\mathcal{H}_{seed}}$ with $\tilde{c}_2 = \mathsf{seed} \oplus \mathsf{mask}$ (i.e., condition $C_{2\text{-}\mathcal{H}_{mask}}$ is triggered).
- If $(\tilde{c}_1||\tilde{c}_2, c)$ is first inserted by Cases 4 or 5, and later $(\tilde{c}_1||\tilde{c}_2, c')$ is inserted by Case 9, then, provided T_{eq} is consistent, no y-collision in $T_{\mathcal{H}_y}$, and no mask-collision in $T_{\mathcal{H}_{mask}}$, the uniformly sample seed by $\mathcal{H}_{seed}$ must collide with the existing one in T^*_r (i.e., condition $C_{2\text{-}\mathcal{H}_{seed}}$ is triggered).
- If $(\tilde{c}_1||\tilde{c}_2, c)$ is first inserted by Cases 10, 11 or 12, and later $(\tilde{c}_1||\tilde{c}_2, c')$ is inserted by Case 9, then either the uniformly sampled mask collides with the xor of existing seed and $\tilde{c}_2$, or the uniformly sampled $\tilde{c}_2$ collides with the xor of existing seed and mask (i.e., conditions $C_{2\text{-}\mathcal{H}_{mask}}$ or $C_{3\text{-}\mathcal{P}^{-1}}$ are triggered).

□
□

Proposition B.5. *Suppose that none of the failure conditions hold. Then for any tuple* (sk, x) *in* $T_{\mathcal{H}_x}$ *and equation* $(\mathsf{str} - x \cdot \diamondsuit = 0)$ *in the span of* T_{eq}, *we have* $\mathsf{str} = \mathbf{KEM}.\mathsf{KG}(\mathsf{sk})$.

Proof. When the pair (sk, x) is recorded in $T_{\mathcal{H}_x}$, the simulator simultaneously inserts $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ into T_{eq} with $\mathsf{pk} = \mathbf{KEM}.\mathsf{KG}(\mathsf{sk})$. Suppose, for the sake of contradiction, that $\mathsf{pk} \neq \mathsf{str}$. Then both $(\mathsf{str} - x \cdot \diamondsuit = 0)$ and $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ lie in the span of T_{eq}, which implies that T_{eq} fails to simulate a proper view of the

generic group. Since no failure conditions are assumed to hold, this contradicts Proposition B.2. Therefore, it must be the case that $\mathsf{str} = \mathbf{KEM}.\mathsf{KG}(\mathsf{sk})$, as claimed. □ □

To facilitate the subsequent analysis, we use the symbol "$\prec$" to indicate the order in which elements are recorded in different tables by the simulator. For instance, when Item_1 is recorded in T_1 prior to Item_2 being recorded in T_2, we write

$$\mathsf{Item}_1 \in \mathsf{T}_1 \prec \mathsf{Item}_2 \in \mathsf{T}_2.$$

If the context is sufficiently clear, this can be abbreviated as $\mathsf{Item}_1 \prec \mathsf{Item}_2$.

Proposition B.6. *Suppose that none of the failure conditions hold. Then for any tuple* $(\mathsf{pk}||\mathsf{r}, \mathsf{seed}) \in T_{\mathcal{H}_{\mathsf{seed}}}$, $(\mathsf{seed}, y) \in T_{\mathcal{H}_y}$, *equations* $(\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0)$ *and* $(\mathsf{str} - y \cdot \mathsf{pk} = 0)$ *in* $\mathsf{Span}(T_{\mathsf{eq}})$, *tuple* $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask}) \in T_{\mathcal{H}_{\mathsf{mask}}}$, *and* $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c}) \in T_{\mathcal{P}}$ *with* $\tilde{\mathsf{c}}_2 = \mathsf{seed} \oplus \mathsf{mask}$, *it holds that*

$$\mathsf{c} = \mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r}).$$

Proof. We argue that, provided no failure conditions are triggered, each item must be inserted into its corresponding table in a specific sequence, which guarantees that $\mathsf{c} = \mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$.

First, suppose $(\mathsf{pk}||\mathsf{r}, \mathsf{seed})$ is inserted by Case 2 of $\mathcal{H}_{\mathsf{seed}}$. Then $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c})$ already exists with $\mathsf{c} = \mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$. Otherwise, if $(\mathsf{pk}||\mathsf{r}, \mathsf{seed})$ is inserted by Case 3, where seed is sampled uniformly at random, then—provided that seed does not already appear in $T_{\mathcal{H}_y}$ and conditions $\mathsf{C}_{2\text{-}\mathcal{H}_y}$ and $\mathsf{C}_{3\text{-}\mathcal{H}_y}$ are not triggered—we obtain

$$(\mathsf{pk}||\mathsf{r}, \mathsf{seed}) \prec (\mathsf{seed}, y) \prec (\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0), (\mathsf{str} - y \cdot \mathsf{pk} = 0).$$

If $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask}) \prec \mathsf{Eq}_1 := (\mathsf{str} - y \cdot \mathsf{pk} = 0)$, then Eq_1 is clearly an equation determined by str, pk, and y that already appears in the table. There are two types to consider:

- Type 1. Eq_1 is induced by the insertion of $(\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0)$.
- Type 2. Eq_1 is induced by the insertion of an independent equation.

For Type 1, the equation $(\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0)$ must be inserted simultaneously when processing $\mathcal{H}_y$ query of (seed, y), and there must already exist $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c})$ with $\mathsf{c} = \mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$. For Type 2, if Eq_1 is induced by the insertion of an equation from $\mathcal{P}^{-1}$ or $\mathcal{G}$, then the uniformly sampled group element must already appear in $T_{\mathcal{H}_{\mathsf{seed}}}$, $T_{\mathcal{H}_{\mathsf{mask}}}$, or T_{eq}, thereby triggering conditions $\mathsf{C}_{4\text{-}\mathcal{P}^{-1}}$ or $\mathsf{C}_{1\text{-}\mathcal{G}}$. Else if Eq_1 is induced by the insertion of an equation $(\widetilde{\mathsf{str}} - \tilde{x} \cdot \diamondsuit = 0)$ from $\mathcal{H}_x$ or $\mathcal{H}_y$, there must exist an equation

$$(\mathsf{str} - y \cdot \mathsf{pk} - \alpha \cdot \widetilde{\mathsf{str}} - \beta \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$$

such that the uniformly sampled $\tilde{x}$ satisfies $\alpha \cdot \tilde{x} + \beta = 0$, thereby triggering conditions $\mathsf{C}_{5\text{-}\mathcal{H}_x}$ or $\mathsf{C}_{4\text{-}\mathcal{H}_y}$.

Similarly, if $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask}) \prec \mathsf{Eq}_2 := (\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0)$, then Eq_2 is a specific equation that can only be induced by the insertion of an independent equation. Specifically, if Eq_2 is induced by an equation from $\mathcal{P}^{-1}$ or $\mathcal{G}$, then the failure conditions $\mathsf{C}_{4\text{-}\mathcal{P}^{-1}}$ or $\mathsf{C}_{1\text{-}\mathcal{G}}$ are triggered. If Eq_2 is induced by an equation $(\widetilde{\mathsf{str}} - \tilde{x} \cdot \diamondsuit = 0)$ from $\mathcal{H}_x$ or $\mathcal{H}_y$, then there must exist

$$(\tilde{\mathsf{c}}_1 - \alpha \cdot \widetilde{\mathsf{str}} - \beta \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}}),$$

so that inserting $(\widetilde{\mathsf{str}} - \tilde{x} \cdot \diamondsuit = 0)$ implies

$$(\tilde{\mathsf{c}}_1 - (\alpha \cdot \tilde{x} + \beta) \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}}),$$

with $y = \alpha \cdot \tilde{x} + \beta$. Consequently, the failure conditions $\mathsf{C}_{8\text{-}\mathcal{H}_x}$ or $\mathsf{C}_{5\text{-}\mathcal{H}_y}$ are triggered.

Therefore, assuming that no failure conditions hold, we obtain the sequence

$$(\mathsf{pk}||\mathsf{r}, \mathsf{seed}) \prec (\mathsf{seed}, y) \prec (\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0), (\mathsf{str} - y \cdot \mathsf{pk} = 0) \prec (\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask}).$$

Clearly, if $(\tilde{c}_1||\mathsf{str},\mathsf{mask})$ is inserted by Types 2 or 3 of $\mathcal{H}_{\mathsf{mask}}$, then there already exists $(\tilde{c}_1||\tilde{c}_2,\mathsf{c})$ with $\mathsf{c} = \mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk},\mathsf{r})$. If $(\tilde{c}_1||\mathsf{str},\mathsf{mask})$ is inserted by Type 4 of $\mathcal{H}_{\mathsf{mask}}$, where mask is sampled uniformly at random, then—provided that condition $\mathsf{C}_{2\text{-}\mathcal{H}_{\mathsf{mask}}}$ is not triggered—we have

$$(\tilde{c}_1||\mathsf{str},\mathsf{mask}) \prec (\tilde{c}_1||\tilde{c}_2,\mathsf{c}).$$

If $(\tilde{c}_1||\tilde{c}_2,\mathsf{c})$ is inserted by $\mathcal{P}$, the simulator ensures

$$\mathsf{c} = \mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk},\mathsf{r}).$$

If it is inserted by $\mathcal{P}^{-1}$ with $\mathsf{c} \neq \mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk},\mathsf{r})$, this would imply the existence of a c-collision in $T_{\mathcal{P}}$, which is excluded by Proposition B.4.

Thus, by exhausting all possibilities, we conclude that as long as no failure conditions occur, the simulator always guarantees $\mathsf{c} = \mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk},\mathsf{r})$, as claimed. □

□

Proposition B.7. *Suppose that none of the failure conditions hold. Then for any tuple* $(\mathsf{pk}||\mathsf{r},\mathsf{seed}) \in T_{\mathcal{H}_{\mathsf{seed}}}$, $(\mathsf{seed},y) \in T_{\mathcal{H}_y}$, *tuple* $(\tilde{c}_1||\tilde{c}_2,\mathsf{c}) \in T_{\mathcal{P}}$ *with* $\mathsf{c} = \mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk},\mathsf{r})$, *and equation* $(\tilde{c}'_1 - y \cdot \diamondsuit = 0)$ *in* $\mathsf{Span}(T_{\mathsf{eq}})$, *it holds that* $\tilde{c}_1 = \tilde{c}'_1$.

Proof. We argue that, provided no failure conditions are triggered, each item must be inserted into its corresponding table in a specific sequence, which guarantees that $\tilde{c}_1 = \tilde{c}'_1$.

If the tuple $(\mathsf{pk}||\mathsf{r},\mathsf{seed})$ is inserted by Case 2 of $\mathcal{H}_{\mathsf{seed}}$, then there already exist $(\tilde{c}_1||\tilde{c}_2,\mathsf{c}) \in T_{\mathcal{P}}$ and the corresponding $(\tilde{c}_1||\mathsf{str},\mathsf{mask}) \in T_{\mathcal{H}_{\mathsf{mask}}}$, with both $\tilde{c}_1||\tilde{c}_2$ and mask sampled uniformly at random. Provided that failure conditions $\mathsf{C}_{2\text{-}\mathcal{H}_{\mathsf{mask}}}$ and $\mathsf{C}_{3\text{-}\mathcal{P}^{-1}}$ are not triggered, we obtain

$$(\tilde{c}_1||\tilde{c}_2,\mathsf{c}) \prec (\tilde{c}_1||\mathsf{str},\mathsf{mask}) \prec (\mathsf{pk}||\mathsf{r},\mathsf{seed}).$$

We also have

$$(\mathsf{seed},y) \prec (\tilde{c}'_1 - y \cdot \diamondsuit = 0).$$

If (seed,y) is inserted after $(\tilde{c}_1||\mathsf{str},\mathsf{mask})$, then when processing $\mathcal{H}_y(\mathsf{seed},y)$ the simulator must simultaneously insert $(\tilde{c}_1 - y \cdot \diamondsuit = 0)$ into T_{eq}. Since T_{eq} is consistent by Proposition B.2, it follows that $\tilde{c}_1 = \tilde{c}'_1$. Otherwise, if $(\mathsf{seed},y) \prec (\tilde{c}_1||\mathsf{str},\mathsf{mask})$, the failure condition $\mathsf{C}_{2\text{-}\mathcal{H}_{\mathsf{mask}}}$ is triggered.

If the tuple $(\mathsf{pk}||\mathsf{r},\mathsf{seed})$ is inserted by Case 3 of $\mathcal{H}_{\mathsf{seed}}$, where seed is randomly sampled, then

$$(\mathsf{pk}||\mathsf{r},\mathsf{seed}) \prec (\mathsf{seed},y) \prec (\tilde{c}'_1 - y \cdot \diamondsuit = 0).$$

Suppose (seed,y) is inserted after $(\tilde{c}_1||\tilde{c}_2,\mathsf{c})$. Then the simulator simultaneously inserts $(\tilde{c}_1 - y \cdot \diamondsuit = 0)$ into T_{eq}, and by Proposition B.2, we again have $\tilde{c}_1 = \tilde{c}'_1$. Alternatively, if $(\mathsf{seed},y) \prec (\tilde{c}_1||\tilde{c}_2,\mathsf{c})$, then when inserting $(\tilde{c}_1||\tilde{c}_2,\mathsf{c})$ there must already exist $(\tilde{c}_1 - y \cdot \diamondsuit = 0)$, unless conditions $\mathsf{C}_{1\text{-}\mathcal{H}_{\mathsf{seed}}}$ or $\mathsf{C}_{3\text{-}\mathcal{P}}$ are triggered. Proposition B.2 then ensures that $\tilde{c}_1 = \tilde{c}'_1$.

Thus, by exhausting all possibilities, we conclude that as long as no failure conditions occur, the simulator always guarantees $\tilde{c}_1 = \tilde{c}'_1$, as claimed. □

□

Proposition B.8. *Suppose that none of the failure conditions hold. Then for any tuple* $(\mathsf{sk},x) \in T_{\mathcal{H}_x}$, $(\mathsf{seed},y) \in T_{\mathcal{H}_y}$, $(\tilde{c}_1||\mathsf{str},\mathsf{mask}) \in T_{\mathcal{H}_{\mathsf{mask}}}$, $(\tilde{c}_1||\tilde{c}_2,\mathsf{c}) \in T_{\mathcal{P}}$, *and equations* $(\tilde{c}'_1 - y \cdot \diamondsuit = 0)$ *and* $(\mathsf{str} - x \cdot \tilde{c}_1 = 0)$ *in* $\mathsf{Span}(T_{\mathsf{eq}})$, *if* $\mathsf{seed} \oplus \mathsf{mask} = \tilde{c}_2$ *and* $\mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk},\mathsf{c}) \neq \perp$, *then it must hold that* $\tilde{c}_1 = \tilde{c}'_1$.

Proof. The simulator records (sk,x) in $T_{\mathcal{H}_x}$, where x is sampled uniformly at random. Provided that failure condition $\mathsf{C}_{4\text{-}\mathcal{H}_x}$ does not hold, we have

$$(\mathsf{sk},x) \prec (\mathsf{str} - x \cdot \tilde{c}_1 = 0).$$

Suppose $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c})$ is inserted by Case 3, 4, or 5 of $\mathcal{P}$, or by Case 9, 10, or 11 of Case $\mathcal{P}^{-1}$. Then $(\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0)$ already exists, and Proposition B.2 ensures that $\tilde{\mathsf{c}}_1 = \tilde{\mathsf{c}}'_1$. If $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c})$ is inserted by Case 6 of $\mathcal{P}$, the private key sk must appear simultaneously in both $T_{\mathcal{H}_x}$ and T^*_{c}, thereby triggering $\mathsf{C}_{1\text{-}\mathcal{H}_x}$ or $\mathsf{C}_{4\text{-}\mathcal{P}}$. Alternatively, if $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c})$ is inserted by Case 12, then

$$(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c}) \prec (\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0).$$

Apparently, if $(\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0)$ is induced prior to the insertion of $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask})$, then $(\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0)$ must eventually exist, and Proposition B.2 again yields $\tilde{\mathsf{c}}_1 = \tilde{\mathsf{c}}'_1$. Otherwise, suppose instead that $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask}) \prec (\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0)$. Let $\mathsf{Eq} := (\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0)$. There are two possible types of induction for Eq:

- Type 1. Eq is induced by the insertion of independent equation.
- Type 2. Eq is induced by the insertion of $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ from $\mathcal{H}_x(\mathsf{sk})$.

For Type 1, Eq is fully determined by existing tuples. If it is induced by an insertion from $\mathcal{P}^{-1}$ or $\mathcal{G}$, then the corresponding uniformly sampled group element must already appear in $T_{\mathcal{H}_{\mathsf{seed}}}$, $T_{\mathcal{H}_{\mathsf{mask}}}$, or T_{eq}, which triggers $\mathsf{C}_{4\text{-}\mathcal{P}^{-1}}$ or $\mathsf{C}_{1\text{-}\mathcal{G}}$. If instead Eq is induced by inserting $(\widetilde{\mathsf{str}} - \tilde{x} \cdot \diamondsuit = 0)$ from $\mathcal{H}_x$ or $\mathcal{H}_y$, then there must exist an equation

$$(\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 - \alpha \cdot \widetilde{\mathsf{str}} - \beta \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$$

forcing the uniformly sampled $\tilde{x}$ to satisfy $\alpha \cdot \tilde{x} + \beta = 0$, thereby triggering $\mathsf{C}_{5\text{-}\mathcal{H}_x}$ or $\mathsf{C}_{4\text{-}\mathcal{H}_y}$.

For Type 2, there must exist an equation

$$(\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 - \alpha \cdot \mathsf{pk} - \beta \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}}).$$

If, at the time of inserting $(\mathsf{pk} - x \cdot \diamondsuit = 0)$, no discrete logarithm of $\tilde{\mathsf{c}}_1$ has been fixed (i.e., no equation of the form $(\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0)$ exists in $\mathsf{Span}(T_{\mathsf{eq}})$), then the uniformly sampled x must satisfy $\alpha \cdot x + \beta = 0$, which triggers $\mathsf{C}_{5\text{-}\mathcal{H}_x}$. On the other hand, if the discrete logarithm of $\tilde{\mathsf{c}}_1$ has already been fixed, it must be $(\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0)$ inserted by $\mathcal{H}_y(\mathsf{seed})$, in which case Proposition B.2 guarantees that $\tilde{\mathsf{c}}_1 = \tilde{\mathsf{c}}'_1$.

Thus, by exhausting all possibilities, we conclude that as long as no failure conditions occur, the simulator always guarantees $\tilde{\mathsf{c}}_1 = \tilde{\mathsf{c}}'_1$, as claimed. □

□

Proposition B.9. *Suppose that none of the failure conditions hold. Then for any tuple* $(\mathsf{sk}, \mathsf{pk}) \in T^*_{\mathsf{sk}}$, $(\mathsf{seed}, y) \in T_{\mathcal{H}_y}$, $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask}) \in T_{\mathcal{H}_{\mathsf{mask}}}$, $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c}) \in T_{\mathcal{P}}$, *and equations* $(\mathsf{pk} - x \cdot \diamondsuit = 0)$, $(\tilde{\mathsf{c}}'_1 - y \cdot \diamondsuit = 0)$ *and* $(\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0)$ *in* $\mathsf{Span}(T_{\mathsf{eq}})$, *if* $\mathsf{seed} \oplus \mathsf{mask} = \tilde{\mathsf{c}}_2$ *and* $\mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c}) \neq \perp$, *then it must hold that* $\tilde{\mathsf{c}}_1 = \tilde{\mathsf{c}}'_1$.

Proof. By Proposition B.1, whenever $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ lies in $\mathsf{Span}(T_{\mathsf{eq}})$, there already exists a corresponding pair $(\mathsf{sk}, \mathsf{pk})$ in T^*_{sk}. Under the non-occurrence of failure condition $\mathsf{C}_{5\text{-}\mathcal{P}^{-1}}$, we have

$$(\mathsf{sk}, \mathsf{pk}) \prec (\mathsf{pk} - x \cdot \diamondsuit = 0), (\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c}).$$

If $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c})$ is inserted through Case 3, 4, or 5 of $\mathcal{P}$, or through Case 9, 10, or 11 of $\mathcal{P}^{-1}$, then $(\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0)$ already exists, and Proposition B.2 guarantees that $\tilde{\mathsf{c}}_1 = \tilde{\mathsf{c}}'_1$. If it is inserted through Case 6 of $\mathcal{P}$, the private key sk must simultaneously appear in both $T_{\mathcal{H}_x}$ and T^*_{c}, thereby triggering $\mathsf{C}_{1\text{-}\mathcal{H}_x}$ or $\mathsf{C}_{4\text{-}\mathcal{P}}$. Alternatively, if $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c})$ is inserted through Case 12, then—under the non-occurrence of conditions $\mathsf{C}_{5\text{-}\mathcal{H}_x}$, $\mathsf{C}_{4\text{-}\mathcal{H}_y}$, $\mathsf{C}_{4\text{-}\mathcal{P}^{-1}}$, and $\mathsf{C}_{1\text{-}\mathcal{G}}$—we obtain

$$(\mathsf{sk}, \mathsf{pk}) \prec (\mathsf{pk} - x \cdot \diamondsuit = 0), (\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c}) \prec (\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0).$$

As in Proposition B.8, the absence of failure conditions ensures that $(\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0) \prec (\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask})$. Therefore, $(\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0)$ must exist, and Proposition B.2 again yields $\tilde{\mathsf{c}}_1 = \tilde{\mathsf{c}}'_1$. □

□

Proposition B.10. *Suppose that none of the failure conditions hold. Then for any tuple* $(\mathsf{pk}||\mathsf{r}, \mathsf{seed}) \in T_{\mathcal{H}_{\mathsf{seed}}}$, $(\mathsf{seed}, y) \in T_{\mathcal{H}_y}$, $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask}) \in T_{\mathcal{H}_{\mathsf{mask}}}$, $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c}) \in T_{\mathcal{P}}$ *with* $\mathsf{c} = \mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$, *and equation* $(\mathsf{str} - y \cdot \mathsf{pk} = 0)$ *in* $\mathsf{Span}(T_{\mathsf{eq}})$, *it holds that* $\mathsf{seed} \oplus \mathsf{mask} = \tilde{\mathsf{c}}_2$.

Proof. If the tuple $(\mathsf{pk}||\mathsf{r}, \mathsf{seed})$ is inserted by Case 2 of $\mathcal{H}_{\mathsf{seed}}$, then $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask})$ already exists in $T_{\mathcal{H}_{\mathsf{mask}}}$ and $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c})$ already exists in $T_{\mathcal{P}}$, such that $\mathsf{seed} \oplus \mathsf{mask} = \tilde{\mathsf{c}}_2$. If $(\mathsf{pk}||\mathsf{r}, \mathsf{seed})$ is inserted by Case 3 of $\mathcal{H}_{\mathsf{seed}}$, where seed is sampled uniformly at random, then

$$(\mathsf{pk}||\mathsf{r}, \mathsf{seed}) \prec (\mathsf{seed}, y) \prec (\mathsf{str} - y \cdot \mathsf{pk} = 0).$$

Apparently, if $(\mathsf{str} - y \cdot \mathsf{pk} = 0)$ is induced before $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask})$, the simulator ensures that $\mathsf{seed} \oplus \mathsf{mask} = \tilde{\mathsf{c}}_2$. Otherwise, suppose $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask}) \prec (\mathsf{str} - y \cdot \mathsf{pk} = 0)$. Let $\mathsf{Eq} := (\mathsf{str} - y \cdot \mathsf{pk} = 0)$. There are two possible types of induction for Eq:

- Type 1. Eq is induced by the insertion of independent equation.
- Type 2. Eq is induced by the insertion of $(\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0)$ from $\mathcal{H}_y(\mathsf{seed})$.

For Type 1, as previously established, one of the failure conditions $\mathsf{C}_{4\text{-}\mathcal{P}^{-1}}$, $\mathsf{C}_{1\text{-}\mathcal{G}}$, $\mathsf{C}_{5\text{-}\mathcal{H}_x}$, or $\mathsf{C}_{4\text{-}\mathcal{H}_y}$ must be triggered. For Type 2, there must exist an equation

$$(\mathsf{str} - y \cdot \mathsf{pk} - \alpha \cdot \tilde{\mathsf{c}}_1 - \beta \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}}).$$

If, at the time of inserting $(\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0)$, no equation of the form $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ exists in $\mathsf{Span}(T_{\mathsf{eq}})$, then the random value y must satisfy $\alpha \cdot y + \beta = 0$, thereby triggering $\mathsf{C}_{5\text{-}\mathcal{H}_y}$. On the other hand, if a discrete logarithm x of pk has already been determined, then $(\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0)$ must also exist. By Proposition B.1, this implies the existence of either (sk, x) in $T_{\mathcal{H}_x}$ or $(\mathsf{sk}, \mathsf{pk})$ in T^*_{sk}. Furthermore, Propositions B.11 and B.12 together ensure that in this case $\mathsf{seed} \oplus \mathsf{mask} = \tilde{\mathsf{c}}_2$ must hold.

Thus, by exhausting all possibilities, we conclude that as long as no failure conditions occur, the simulator always guarantees $\mathsf{seed} \oplus \mathsf{mask} = \tilde{\mathsf{c}}_2$, as claimed. □

□

Proposition B.11. *Suppose that none of the failure conditions hold. Then for any tuple* $(\mathsf{pk}||\mathsf{r}, \mathsf{seed}) \in T_{\mathcal{H}_{\mathsf{seed}}}$, $(\mathsf{sk}, x) \in T_{\mathcal{H}_x}$, $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask}) \in T_{\mathcal{H}_{\mathsf{mask}}}$, $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c}) \in T_{\mathcal{P}}$ *with* $\mathsf{c} = \mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$, *and equation* $(\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0)$ *in* $\mathsf{Span}(T_{\mathsf{eq}})$, *it holds that* $\mathsf{seed} \oplus \mathsf{mask} = \tilde{\mathsf{c}}_2$.

Proof. For the tuple $(\mathsf{pk}||\mathsf{r}, \mathsf{seed})$, if it is inserted by Case 2 of $\mathcal{H}_{\mathsf{seed}}$, then $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask})$ already exists in $T_{\mathcal{H}_{\mathsf{mask}}}$ and $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c})$ already exists in $T_{\mathcal{P}}$, such that $\mathsf{seed} \oplus \mathsf{mask} = \tilde{\mathsf{c}}_2$. Otherwise, seed is sampled uniformly at random by $\mathcal{H}_{\mathsf{seed}}(\mathsf{pk}, \mathsf{r})$.

For $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c})$, if it is inserted by Case 3 of $\mathcal{P}$ or Case 9 of $\mathcal{P}^{-1}$, then $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask})$ already exists in $T_{\mathcal{H}_{\mathsf{mask}}}$ and $(\mathsf{pk}||\mathsf{r}, \mathsf{seed})$ already exists in $T_{\mathcal{H}_{\mathsf{seed}}}$, such that $\mathsf{seed} \oplus \mathsf{mask} = \tilde{\mathsf{c}}_2$. If it is inserted by Case 4, 5, or 6 of $\mathcal{P}$, the value r must simultaneously appear in both $T_{\mathcal{H}_{\mathsf{seed}}}$ and T^*_{r} or T^*_{c}, thereby triggering conditions $\mathsf{C}_{1\text{-}\mathcal{H}_{\mathsf{seed}}}$ or $\mathsf{C}_{3\text{-}\mathcal{P}}$. Otherwise, $\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2$ is sampled randomly, and there is

$$(\mathsf{sk}, x), (\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c}) \prec (\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0).$$

Suppose that $(\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0)$ is induced before the insertion of $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask})$, with $(\mathsf{pk}||\mathsf{r}, \mathsf{seed})$ being inserted through Type 3 of $\mathcal{H}_{\mathsf{seed}}$. Then we have

$$(\mathsf{pk}||\mathsf{r}, \mathsf{seed}), (\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0) \prec (\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask}).$$

In this case, the simulator ensures that $\mathsf{mask} = \mathsf{seed} \oplus \tilde{\mathsf{c}}_2$.

In contrast, if $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask}) \prec (\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0)$, let $\mathsf{Eq} := (\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0)$. There are two possible types of induction for Eq:

- Type 1. Eq is induced by the insertion of independent equation.
- Type 2. Eq is induced by the insertion of $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ from $\mathcal{H}_x(\mathsf{sk})$.

For Type 1, as previously established, one of the failure conditions $\mathsf{C}_{4\text{-}\mathcal{P}^{-1}}$, $\mathsf{C}_{1\text{-}\mathcal{G}}$, $\mathsf{C}_{5\text{-}\mathcal{H}_x}$, or $\mathsf{C}_{4\text{-}\mathcal{H}_y}$ must be triggered. For Type 2, there must exist an equation

$$(\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 - \alpha \cdot \mathsf{pk} - \beta \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}}).$$

Suppose that, when inserting $(\mathsf{pk} - x \cdot \diamondsuit = 0)$, there is no $(\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0)$ in $\mathsf{Span}(T_{\mathsf{eq}})$. In this case, the random discrete logarithm x must satisfy $\alpha \cdot x + \beta = 0$, thereby triggering condition $\mathsf{C}_{5\text{-}\mathcal{H}_x}$.

On the other hand, if a discrete logarithm y of $\tilde{\mathsf{c}}_1$ already exists, then it must have been inserted by $\mathcal{H}_y(\mathsf{seed})$, and $(\mathsf{str} - y \cdot \mathsf{pk} = 0)$ must also appear in $\mathsf{Span}(T_{\mathsf{eq}})$. Thus, Proposition B.10 ensures that $\mathsf{seed} \oplus \mathsf{mask} = \tilde{\mathsf{c}}_2$.

Thus, by exhausting all possibilities, we conclude that as long as no failure conditions occur, the simulator always guarantees $\mathsf{seed} \oplus \mathsf{mask} = \tilde{\mathsf{c}}_2$, as claimed. □

□

Proposition B.12. *Suppose that none of the failure conditions hold. Then for any tuple* $(\mathsf{pk}||\mathsf{r}, \mathsf{seed}) \in T_{\mathcal{H}_{\mathsf{seed}}}$, $(\mathsf{sk}, \mathsf{pk}) \in T^*_{\mathsf{sk}}$, $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask}) \in T_{\mathcal{H}_{\mathsf{mask}}}$, $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c}) \in T_{\mathcal{P}}$ *with* $\mathsf{c} = \mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$, *and equations* $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ *and* $(\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0)$ *in* $\mathsf{Span}(T_{\mathsf{eq}})$, *it holds that* $\mathsf{seed} \oplus \mathsf{mask} = \tilde{\mathsf{c}}_2$.

Proof. For the tuple $(\mathsf{pk}||\mathsf{r}, \mathsf{seed})$, if it is inserted by Case 2 of $\mathcal{H}_{\mathsf{seed}}$, then $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask})$ already exists in $T_{\mathcal{H}_{\mathsf{mask}}}$ and $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c})$ already exists in $T_{\mathcal{P}}$, such that $\mathsf{seed} \oplus \mathsf{mask} = \tilde{\mathsf{c}}_2$. Otherwise, seed is sampled uniformly at random by $\mathcal{H}_{\mathsf{seed}}(\mathsf{pk}, \mathsf{r})$.

For $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c})$, if it is inserted by Case 3 of $\mathcal{P}$ or Case 9 of $\mathcal{P}^{-1}$, then $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask})$ already exists in $T_{\mathcal{H}_{\mathsf{mask}}}$ and $(\mathsf{pk}||\mathsf{r}, \mathsf{seed})$ already exists in $T_{\mathcal{H}_{\mathsf{seed}}}$, such that $\mathsf{seed} \oplus \mathsf{mask} = \tilde{\mathsf{c}}_2$. If it is inserted by Case 4, 5, or 6 of $\mathcal{P}$, the value r must simultaneously appear in both $T_{\mathcal{H}_{\mathsf{seed}}}$ and either T^*_{r} or T^*_{c}, thereby triggering conditions $\mathsf{C}_{1\text{-}\mathcal{H}_{\mathsf{seed}}}$ or $\mathsf{C}_{3\text{-}\mathcal{P}}$. Otherwise, $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c})$ may be inserted by Case 10, 11, or 12 of $\mathcal{P}^{-1}$.

Consider Case 10 of $\mathcal{P}^{-1}$. Provided that failure conditions $\mathsf{C}_{4\text{-}\mathcal{P}^{-1}}$ and $\mathsf{C}_{1\text{-}\mathcal{G}}$ are not triggered, we have

$$(\mathsf{sk}, \mathsf{pk}) \prec (\mathsf{pk}||\mathsf{r}, \mathsf{seed}), (\mathsf{pk} - x \cdot \diamondsuit = 0) \prec (\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c}), (\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0).$$

Suppose the equation $(\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0)$ is recorded prior to the insertion of $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask})$. In this case, the simulator ensures that $\mathsf{seed} \oplus \mathsf{mask} = \tilde{\mathsf{c}}_2$. In contrast, if $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask}) \prec (\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0)$, let $\mathsf{Eq} := (\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0)$. There are two possible types of induction for Eq:

- Type 1. Eq is induced by the insertion of independent equation.
- Type 2. Eq is induced by the insertion of $(\mathsf{pk} - x \cdot \diamondsuit = 0)$.

For Type 1, as previously established, one of the failure conditions $\mathsf{C}_{4\text{-}\mathcal{P}^{-1}}$, $\mathsf{C}_{1\text{-}\mathcal{G}}$, $\mathsf{C}_{5\text{-}\mathcal{H}_x}$, or $\mathsf{C}_{4\text{-}\mathcal{H}_y}$ must be triggered. For Type 2, since $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c})$ is inserted by Case 10, there must also exist $(\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0)$, and consequently $(\mathsf{str} - y \cdot \mathsf{pk} = 0)$ appears in $\mathsf{Span}(T_{\mathsf{eq}})$. Thus, Proposition B.10 ensures that $\mathsf{seed} \oplus \mathsf{mask} = \tilde{\mathsf{c}}_2$.

Consider Case 11 or 12 or $\mathcal{P}^{-1}$. Provided that the failure conditions $\mathsf{C}_{4\text{-}\mathcal{P}^{-1}}$ and $\mathsf{C}_{1\text{-}\mathcal{G}}$ are not triggered, we have

$$(\mathsf{sk}, \mathsf{pk}) \prec (\mathsf{pk}||\mathsf{r}, \mathsf{seed}), (\mathsf{pk} - x \cdot \diamondsuit = 0), (\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c})$$

and

$$(\mathsf{pk} - x \cdot \diamondsuit = 0), (\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c}) \prec (\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0).$$

Suppose the equation $(\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0)$ is recorded prior to the insertion of $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask})$. In this case, the simulator ensures that $\mathsf{seed} \oplus \mathsf{mask} = \tilde{\mathsf{c}}_2$. In contrast, if $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask}) \prec (\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0)$, then by an analysis identical to that of Case 10 of $\mathcal{P}^{-1}$ above, we again conclude that $\mathsf{seed} \oplus \mathsf{mask} = \tilde{\mathsf{c}}_2$, provided no failure conditions are triggered.

Thus, by exhausting all possibilities, we conclude that as long as no failure conditions occur, the simulator always guarantees $\mathsf{seed} \oplus \mathsf{mask} = \tilde{\mathsf{c}}_2$, as claimed. □ □

Proposition B.13. *Suppose that none of the failure conditions hold. Then for any tuple* $(\mathsf{pk}||\mathsf{r}, \mathsf{seed}) \in T_{\mathcal{H}_{\mathsf{seed}}}$ *and tuple* $(\mathsf{pk}||\mathsf{seed}, \mathsf{k}) \in T_{\mathcal{H}_{\mathsf{key}}}$*, it must hold that*

$$(*, \mathsf{k}) \leftarrow \mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r}).$$

Proof. For the tuple $(\mathsf{pk}||\mathsf{r}, \mathsf{seed})$, if it is inserted by Case 2 of $\mathcal{H}_{\mathsf{seed}}$, then, provided that failure condition $\mathsf{C}_{2\text{-}\mathcal{H}_{\mathsf{mask}}}$ is not triggered, when inserting $(\mathsf{pk}||\mathsf{seed}, \mathsf{k})$ there already exist a corresponding sk and c such that

$$\mathsf{pk} \leftarrow \mathbf{KEM}.\mathsf{KG}(\mathsf{sk}), \quad (\mathsf{c}, *) \leftarrow \mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r}),$$

and by Proposition B.14 and Proposition B.15 the simulator ensures

$$\mathsf{k} = \mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c}).$$

If $(\mathsf{pk}||\mathsf{r}, \mathsf{seed})$ is inserted by Case 3 of $\mathcal{H}_{\mathsf{seed}}$, where seed is sampled uniformly at random, then, provided that failure condition $\mathsf{C}_{2\text{-}\mathcal{H}_{\mathsf{seed}}}$ is not triggered, we have

$$(\mathsf{pk}||\mathsf{r}, \mathsf{seed}) \prec (\mathsf{pk}||\mathsf{seed}, \mathsf{k}),$$

and the simulator guarantees

$$(*, \mathsf{k}) \leftarrow \mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r}).$$

□
□

Proposition B.14. *Suppose that none of the failure conditions occur. Then for any tuples* $(\mathsf{sk}, x) \in T_{\mathcal{H}_x}$, $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask}) \in T_{\mathcal{H}_{\mathsf{mask}}}$, $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c}) \in T_{\mathcal{P}}$, $(\mathsf{pk}||\mathsf{seed}, \mathsf{k}) \in T_{\mathcal{H}_{\mathsf{key}}}$*, and equation* $(\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0)$ *in* $\mathsf{Span}(T_{\mathsf{eq}})$*, if*

$$\mathsf{seed} \oplus \mathsf{mask} = \tilde{\mathsf{c}}_2, \ \mathsf{pk} = \mathbf{KEM}.\mathsf{KG}(\mathsf{sk}), \ \mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c}) \neq \perp,$$

then it must hold that

$$\mathsf{k} = \mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c}).$$

Proof. For the tuple $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c})$, if it is inserted by Case 3 of $\mathcal{P}$ or by Cases 9, 10, or 11 of $\mathcal{P}^{-1}$, then there exists $(\mathsf{pk}||\mathsf{r}, \mathsf{seed})$ with seed sampled uniformly at random and $(\mathsf{pk}, \mathsf{r})$ corresponds to c. As established in Proposition B.13, we have

$$(*, \mathsf{k}) \leftarrow \mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r}).$$

If $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c})$ is inserted by Case 4 of $\mathcal{P}$, then there exists $(\mathsf{pk}||\mathsf{seed}, \mathsf{r}) \in T^*_{\mathsf{r}}$, where $(\mathsf{pk}, \mathsf{r})$ corresponds to c, and it can only be inserted through $\mathcal{H}_{\mathsf{key}}(\mathsf{pk}||\mathsf{seed})$. Therefore,

$$(\mathsf{c}, \mathsf{k}) = \mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r}).$$

Similarly, if $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c})$ is inserted by Case 5 of $\mathcal{P}$, then necessarily

$$(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c}) \prec (\mathsf{pk}||\mathsf{seed}, \mathsf{k}),$$

where the simulator simultaneously inserts $(\mathsf{pk}||\mathsf{seed}, \mathsf{r})$ when inserting $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c})$. Thus, both k and c correspond to the same $(\mathsf{pk}, \mathsf{r})$.

If $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c})$ is inserted by Case 6 of $\mathcal{P}$, then there exists a private key sk that appears simultaneously in both $T_{\mathcal{H}_x}$ and T^*_{c}, thereby triggering $\mathsf{C}_{1\text{-}\mathcal{H}_x}$ or $\mathsf{C}_{4\text{-}\mathcal{P}}$.

If $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c})$ is inserted by Case 12 of $\mathcal{P}^{-1}$, where $\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2$ is sampled uniformly at random, then provided no failure conditions occur, we have

$$(\mathsf{sk}, x), (\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c}) \prec (\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0).$$

Suppose the equation $(\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0)$ is recorded prior to the insertion of $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask})$. If $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask}) \prec (\mathsf{pk}||\mathsf{seed}, \mathsf{k})$, then

$$\mathsf{k} = \mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c}).$$

Conversely, if $(\mathsf{pk}||\mathsf{seed}, \mathsf{k}) \prec (\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask})$, then there exists $(\mathsf{pk}, \mathsf{r}, \mathsf{seed})$; provided that $\mathsf{C}_{2\text{-}\mathcal{H}_{\mathsf{mask}}}$ is not triggered, Proposition B.13 guarantees

$$(*, \mathsf{k}) \leftarrow \mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r}).$$

In contrast, if $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask}) \prec (\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0)$, let $\mathsf{Eq} := (\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0)$. There are two possible types of induction for Eq:

- Type 1. Eq is induced by the insertion of independent equation.
- Type 2. Eq is induced by the insertion of $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ from $\mathcal{H}_x(\mathsf{sk})$.

For Type 1, as previously established, one of the failure conditions $\mathsf{C}_{4\text{-}\mathcal{P}^{-1}}$, $\mathsf{C}_{1\text{-}\mathcal{G}}$, $\mathsf{C}_{5\text{-}\mathcal{H}_x}$, or $\mathsf{C}_{4\text{-}\mathcal{H}_y}$ must be triggered. For Type 2, there must exist an equation

$$(\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 - \alpha \cdot \mathsf{pk} - \beta \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}}).$$

If when inserting $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ there is no $(\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0)$ in $\mathsf{Span}(T_{\mathsf{eq}})$, then the random discrete logarithm x must satisfy $\alpha \cdot x + \beta = 0$, thereby triggering $\mathsf{C}_{5\text{-}\mathcal{H}_x}$. On the other hand, if a discrete logarithm y of $\tilde{\mathsf{c}}_1$ already exists, then it must have been inserted by $\mathcal{H}_y(\mathsf{seed})$ with $(\mathsf{pk}||\mathsf{r}, \mathsf{seed}) \in T_{\mathcal{H}_{\mathsf{seed}}}$. In this case, Proposition B.13 ensures that $\mathsf{k} = \mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c})$. □
□

Proposition B.15. *Suppose that none of the failure conditions occur. Then for any tuples* $(\mathsf{sk}, \mathsf{pk}) \in T^*_{\mathsf{sk}}$, $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask}) \in T_{\mathcal{H}_{\mathsf{mask}}}$, $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c}) \in T_{\mathcal{P}}$, $(\mathsf{pk}||\mathsf{seed}, \mathsf{k}) \in T_{\mathcal{H}_{\mathsf{key}}}$, *and equations* $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ *and* $(\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0)$ *in* $\mathsf{Span}(T_{\mathsf{eq}})$, *if*

$$\mathsf{seed} \oplus \mathsf{mask} = \tilde{\mathsf{c}}_2, \ \mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c}) \neq \perp,$$

then it must hold that

$$\mathsf{k} = \mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c}).$$

The proof is almost identical to that of Proposition B.14.

Proposition B.16. *Suppose that none of the failure conditions occur. Then for any tuples* $(\mathsf{sk}, x) \in T_{\mathcal{H}_x}$, $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask}) \in T_{\mathcal{H}_{\mathsf{mask}}}$, $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c}) \in T_{\mathcal{P}}$, $(\mathsf{seed}, y) \in T_{\mathcal{H}_y}$, *and equations* $(\tilde{\mathsf{c}}'_1 - y \cdot \diamondsuit = 0)$ *and* $(\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0)$ *in* $\mathsf{Span}(T_{\mathsf{eq}})$, *if*

$$\mathsf{seed} \oplus \mathsf{mask} = \tilde{\mathsf{c}}_2 \textit{ and } \mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c}) = \perp,$$

then it must hold that $\tilde{\mathsf{c}}_1 \neq \tilde{\mathsf{c}}'_1$.

Proof. By Proposition B.8, for $(\mathsf{sk}', x') \in T_{\mathcal{H}_x}$ with $\mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk}', \mathsf{c}) \in \mathcal{K}$ and the corresponding tuples existing, we have $(\tilde{\mathsf{c}}_1 - y' \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$. If $\tilde{\mathsf{c}}_1 = \tilde{\mathsf{c}}'_1$ and there is no y-collision in $T_{\mathcal{H}_y}$, then a group element would possess two distinct discrete logarithms, contradicting Proposition B.2. □
□

Proposition B.17. *Suppose that none of the failure conditions occur. Then for any tuples* $(\mathsf{seed}, y) \in T_{\mathcal{H}_y}$, $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask}) \in T_{\mathcal{H}_{\mathsf{mask}}}$, $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c}) \in T_{\mathcal{P}}$, $(\mathsf{pk}||\mathsf{seed}, \mathsf{k}) \in T_{\mathcal{H}_{\mathsf{key}}}$, *and equations* $(\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0)$ *and* $(\mathsf{str} - y \cdot \mathsf{pk} = 0)$ *in* $\mathsf{Span}(T_{\mathsf{eq}})$, *if* $\mathsf{seed} \oplus \mathsf{mask} = \tilde{\mathsf{c}}_2$, *then it must hold that*

$$\mathsf{k} = \mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c}),$$

where $\mathsf{pk} = \mathbf{KEM}.\mathsf{KG}(\mathsf{sk})$.

Proof. Assume that the failure conditions $\mathsf{C}_{2\text{-}\mathcal{H}_y}$ and $\mathsf{C}_{3\text{-}\mathcal{H}_y}$ are not triggered. Then we have

$$(\mathsf{seed}, y) \prec (\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0), (\mathsf{str} - y \cdot \mathsf{pk} = 0).$$

First, suppose that $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask})$ is recorded prior to the induction of $(\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0)$. If $(\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0)$ is inserted by $\mathcal{H}_y(\mathsf{seed})$, then Propositions B.13, B.14, and B.15 already guarantee that

$$\mathsf{k} = \mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c}).$$

Otherwise, if $(\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0)$ is induced by any other independent equation, it follows (as previously established) that one of the failure conditions $\mathsf{C}_{5\text{-}\mathcal{H}_x}$, $\mathsf{C}_{4\text{-}\mathcal{H}_y}$, $\mathsf{C}_{4\text{-}\mathcal{P}^{-1}}$, or $\mathsf{C}_{1\text{-}\mathcal{G}}$ must be triggered.

Next, suppose $(\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0) \prec (\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask})$. Assuming that the conditions $\mathsf{C}_{4\text{-}\mathcal{P}^{-1}}$, $\mathsf{C}_{1\text{-}\mathcal{G}}$, $\mathsf{C}_{5\text{-}\mathcal{H}_x}$, and $\mathsf{C}_{4\text{-}\mathcal{H}_y}$ are not triggered, we obtain

$$(\mathsf{seed}, y) \prec (\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0), (\mathsf{str} - y \cdot \mathsf{pk} = 0) \prec (\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask}).$$

If $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask})$ is inserted by Case 2 or Case 3 of $\mathcal{H}_{\mathsf{mask}}$, then there exists $(\mathsf{pk}||\mathsf{r}, \mathsf{seed})$ corresponding to c, and Proposition B.13 ensures that

$$\mathsf{k} = \mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c}).$$

If instead $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask})$ is inserted by Case 4 of $\mathcal{H}_{\mathsf{mask}}$, then mask is sampled uniformly at random. Provided that condition $\mathsf{C}_{2\text{-}\mathcal{H}_{\mathsf{mask}}}$ is not triggered, we further have

$$(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask}) \prec (\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c}).$$

If $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c})$ is inserted by Cases 9, 10, or 11 of $\mathcal{P}^{-1}$, then there exists $(\mathsf{pk}||\mathsf{r}, \mathsf{seed})$ corresponding to c, and Proposition B.13 again guarantees that

$$\mathsf{k} = \mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c}).$$

If instead $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c})$ is inserted by $\mathcal{P}$, then there must exist $(\mathsf{pk}, \mathsf{r})$ corresponding to both c and k, which implies $\mathsf{k} = \mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c})$. □

□

B.4 Transition from $\mathbf{G}_2$ to $\mathbf{G}_3$

Game 3. In this game, the simulator $\mathcal{S}_{\mathsf{fail}}$ remains unchanged. Instead, the relay algorithm $\mathcal{F} = (\mathcal{F}.\mathsf{KG}, \mathcal{F}.\mathsf{Encaps}, \mathcal{F}.\mathsf{Decaps})$ is replaced with the construction $\Pi_{\mathsf{KEM}} = (\Pi_{\mathsf{KEM}}.\mathsf{KG}, \Pi_{\mathsf{KEM}}.\mathsf{Encaps}, \Pi_{\mathsf{KEM}}.\mathsf{Decaps})$. Notably, Π_{KEM} does not query the ideal KEM directly; rather, it interacts with $\mathcal{S}_{\mathsf{fail}}$, thereby ensuring that the honest interfaces are consistent with the adversarial interfaces. Consequently, the differentiator can distinguish between Game 2 and Game 3 only if it detects an inconsistency between $\mathcal{F}$ and $\mathcal{S}_{\mathsf{fail}}$ in Game 2.

Let $\mathbf{G}_3$ denote Game 3, which outputs 1 if and only if $\mathcal{D}$ outputs 1 after interacting with Π_{KEM} and $\mathcal{S}_{\mathsf{fail}}$. Formally,

$$\Pr[\mathbf{G}_3(\mathcal{D}) = 1] = \Pr[\mathcal{D}^{\Pi_{\mathsf{KEM}}^{\mathcal{S}_{\mathsf{fail}}}, \mathcal{S}_{\mathsf{fail}}^{\mathbf{KEM}}}(1^\lambda) = 1].$$

Lemma B.2. *Let q denote the number of queries made by the differentiator to both the honest and adversarial interfaces. Then,*

$$|\Pr[\mathbf{G}_3(\mathcal{D}) = 1] - \Pr[\mathbf{G}_2(\mathcal{D}) = 1]| \leq O\left(\frac{(\lambda \cdot q)^4}{N} + \frac{(\lambda \cdot q)^3}{2^\lambda}\right).$$

Proof. To prove that the views in Game 2 and Game 3 are statistically close, we argue that the differentiator can only exploit an inconsistency if the simulator $\mathcal{S}_{\mathsf{fail}}$ fails in at least one of the two games.

For Game 3, recall that the honest interfaces produce their responses by querying the adversarial interfaces, namely the simulator $\mathcal{S}_{\mathsf{fail}}$. Hence, as long as the simulator provides consistent answers to identical queries across all invocations, the responses of the honest interface necessarily coincide with those of the

adversarial interface. For random oracle queries, this consistency holds trivially. Propositions B.3 and B.4 guarantee that a permutation query cannot produce more than one distinct response, while Proposition B.2 ensures the consistency of T_{eq}, i.e., that the responses to $\mathcal{G}$ queries are proper.

For Game 2, any inconsistency between the honest interface $\mathcal{F}$ and the adversarial interface $\mathcal{S}_{\mathsf{fail}}$ can occur only when the response generated by $\mathcal{F}$ deviates from that of the simulator implementing the Π_{KEM} construction. We now establish that, provided $\mathcal{S}_{\mathsf{fail}}$ does not fail, such an inconsistency cannot arise.

Inconsistency with respect to $\mathcal{F}$.KG. For any public-key generation query on input sk, the relay algorithm $\mathcal{F}$ always returns $\mathsf{pk} \leftarrow \mathbf{KEM}.\mathsf{KG}(\mathsf{sk})$. Thus, an inconsistency w.r.t. $\mathcal{F}$.KG arises if and only if $\mathcal{S}_{\mathsf{fail}}$ responds with some str such that $\mathsf{str} \neq \mathsf{pk}$.

Concretely, when the differentiator generates the public key for sk through $\mathcal{S}_{\mathsf{fail}}$ according to the Π_{KEM} construction, it queries $\mathcal{H}_x(\mathsf{sk})$ to obtain x, and then queries for the label str of x. This ensures that (sk, x) in $T_{\mathcal{H}_x}$ and $(\mathsf{str} - x \cdot \diamondsuit = 0)$ in the span of T_{eq}. By Proposition B.5, we deduce that

$$\mathsf{str} = \mathbf{KEM}.\mathsf{KG}(\mathsf{sk})$$

unless the simulator $\mathcal{S}_{\mathsf{fail}}$ fails.

Inconsistency with respect to $\mathcal{F}$.Encaps. For any encapsulation query on input pk and r, the relay algorithm $\mathcal{F}$ always returns $\mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$. Consistency therefore requires that the following two hold:

1. If pk is invalid with respect to the ideal KEM, then $\mathcal{S}_{\mathsf{fail}}.\mathcal{G}^{\mathsf{add}}$ on input pk must return $\perp$.
2. Otherwise, let $(\mathsf{c}, \mathsf{k}) \leftarrow \mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$, the simulator $\mathcal{S}_{\mathsf{fail}}$ must return the same ciphertext c and key k.

Based on the description of the simulator, when processing an addition query with inputs $(\mathsf{str}_1, \mathsf{str}_2)$, the simulator determines their validity by querying $\mathbf{KEM}.\mathsf{Encaps}$. Therefore, for any public key pk that is invalid with respect to the ideal KEM, we must have

$$\mathcal{S}_{\mathsf{fail}}.\mathcal{G}^{\mathsf{add}}(\mathsf{pk}, \mathsf{pk}) = \perp.$$

Consider any private key $\mathsf{sk} \in \mathbb{Z}_N$, with $\mathsf{pk} \leftarrow \mathbf{KEM}.\mathsf{KG}(\mathsf{sk})$, and any randomness r. When the differentiator derives the ciphertext c corresponding $(\mathsf{pk}, \mathsf{r})$, it must query $\mathcal{P}$ with $\tilde{\mathsf{c}}_1 || \tilde{\mathsf{c}}_2$, where both $\tilde{\mathsf{c}}_1$ and $\tilde{\mathsf{c}}_2$ correspond to $(\mathsf{pk}, \mathsf{r})$. Specifically, the relationship between $(\mathsf{pk}, \mathsf{r})$ and $\tilde{\mathsf{c}}_1 || \tilde{\mathsf{c}}_2$ must contain:

- There exist $(\mathsf{pk} || \mathsf{r}, \mathsf{seed}) \in T_{\mathcal{H}_{\mathsf{seed}}}$ and $(\mathsf{seed}, y) \in T_{\mathcal{H}_y}$.
- The equation $(\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0)$ lies in the span of T_{eq}.
- There exists str with equation $(\mathsf{str} - y \cdot \mathsf{pk} = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$ and $(\tilde{\mathsf{c}}_1 || \mathsf{str}, \mathsf{mask}) \in T_{\mathcal{H}_{\mathsf{mask}}}$ such that $\mathsf{seed} \oplus \mathsf{mask} = \tilde{\mathsf{c}}_2$.

In this case, Proposition B.6 ensures that

$$(\mathsf{c}, *) = \mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r}).$$

When the differentiator derives the key k corresponding to $(\mathsf{pk}, \mathsf{r})$, it must query $\mathcal{H}_{\mathsf{key}}$ with $(\mathsf{pk}, \mathsf{seed})$, where seed corresponds to $(\mathsf{pk}, \mathsf{r})$. Specifically, the relationship between $(\mathsf{pk}, \mathsf{r})$ and seed must fall into one of the following associations:

- Form 1: There exists $(\mathsf{pk} || \mathsf{r}, \mathsf{seed})$ in $T_{\mathcal{H}_{\mathsf{seed}}}$.
- Form 2: There exists $(\tilde{\mathsf{c}}_1 || \tilde{\mathsf{c}}_2, \mathsf{c}) \in T_{\mathcal{P}}$ with c corresponding to $(\mathsf{pk}, \mathsf{r})$, both $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ and $(\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0)$ lying in $\mathsf{Span}(T_{\mathsf{eq}})$, and $(\tilde{\mathsf{c}}_1 || \mathsf{str}, \mathsf{mask}) \in T_{\mathcal{H}_{\mathsf{mask}}}$, such that $\mathsf{seed} = \mathsf{mask} \oplus \tilde{\mathsf{c}}_2$.

Let $(\mathsf{c}, \mathsf{k}) \leftarrow \mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$. For Form 1, by Proposition B.13, it follows that

$$\mathcal{H}_{\mathsf{key}}(\mathsf{pk}, \mathsf{seed}) = \mathsf{k}.$$

For Form 2, by Proposition B.1, the presence of $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ in the sapn of T_{eq} implies that either $(\mathsf{sk}, x) \in T_{\mathcal{H}_x}$ or $(\mathsf{sk}, \mathsf{pk}) \in T^*_{\mathsf{sk}}$. In either case, Proposition B.14 and Proposition B.15 guarantee that

$$\mathcal{H}_{\mathsf{key}}(\mathsf{pk}, \mathsf{seed}) = \mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c}) = \mathsf{k}.$$

Furthermore, the differentiator may also query $\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2 \leftarrow \mathcal{P}^{-1}(\mathsf{c})$. In this case, the simulator must return the same $\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2$.

Inconsistency with respect to $\tilde{\mathsf{c}}_1$. For any valid public key pk and randomness r, let $(\mathsf{c}, \mathsf{k}) \leftarrow \mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$ and suppose $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c}) \in T_{\mathcal{P}}$. In this case, the differentiator must query $y \leftarrow \mathcal{H}_y(\mathsf{seed})$, where seed corresponds to $(\mathsf{pk}, \mathsf{r})$, and it must hold that $\mathcal{G}^{\mathsf{label}}(y) = \tilde{\mathsf{c}}_1$. Here the relationship between $(\mathsf{pk}, \mathsf{r})$ and seed must fall into one of the following associations:

- Form 1: There exists $(\mathsf{pk}||\mathsf{r}, \mathsf{seed})$ in $T_{\mathcal{H}_{\mathsf{seed}}}$.
- Form 2: There exists $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask}) \in T_{\mathcal{H}_{\mathsf{mask}}}$ such that both $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ and $(\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0)$ lie in $\mathsf{Span}(T_{\mathsf{eq}})$ and $\mathsf{seed} = \mathsf{mask} \oplus \tilde{\mathsf{c}}_2$.

For Form 1, by Proposition B.7, it follows that

$$(\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}}).$$

For Form 2, by Proposition B.1, the presence of $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ in the sapn of T_{eq} implies that either $(\mathsf{sk}, x) \in T_{\mathcal{H}_x}$ or $(\mathsf{sk}, \mathsf{pk}) \in T^*_{\mathsf{sk}}$. In either case, Proposition B.8 and Proposition B.9 guarantee this consistency.

Inconsistency with respect to $\tilde{\mathsf{c}}_2$. For any $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c})$ in $T_{\mathcal{P}}$ with c corresponding to $(\mathsf{pk}, \mathsf{r})$, the differentiator must query $\mathsf{seed} \leftarrow \mathcal{H}_{\mathsf{seed}}(\mathsf{pk}, \mathsf{r})$ and $\mathsf{mask} \leftarrow \mathcal{H}_{\mathsf{mask}}(\tilde{\mathsf{c}}_1||\mathsf{str})$. It must hold that $\mathsf{seed} \oplus \mathsf{mask} = \tilde{\mathsf{c}}_2$. Here str must fall into one of the following associations:

- Form 1: There exists (seed, y) in $T_{\mathcal{H}_y}$ with $(\mathsf{str} - y \cdot \mathsf{pk} = 0)$ lying in $\mathsf{Span}(T_{\mathsf{eq}})$.
- Form 2: The equations $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ and $(\mathsf{str} - x \cdot \tilde{\mathsf{c}}_1 = 0)$ lie in $\mathsf{Span}(T_{\mathsf{eq}})$.

For Form 1, by Proposition B.10, it follows that

$$\mathsf{seed} \oplus \mathsf{mask} = \tilde{\mathsf{c}}_2$$

For Form 2, by Proposition B.1, the presence of $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ in the sapn of T_{eq} implies that either $(\mathsf{sk}, x) \in T_{\mathcal{H}_x}$ or $(\mathsf{sk}, \mathsf{pk}) \in T^*_{\mathsf{sk}}$. In either case, Proposition B.11 and Proposition B.12 guarantee this consistency.

Inconsistency with respect to $\mathcal{F}.\mathsf{Decaps}$. For any decapsulation query with input sk and c, the relay algorithm $\mathcal{F}$ always returns $\mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c})$. Consistency therefore requires that the following two hold:

1. If c is invalid with respect to the ideal KEM, and there exists $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c})$ in $T_{\mathcal{P}}$, then $\mathcal{S}_{\mathsf{fail}}.\mathcal{G}^{\mathsf{add}}$ on input $\tilde{\mathsf{c}}_1$ must return $\perp$.
2. Otherwise, letting $\mathsf{k} \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c})$, the simulator $\mathcal{S}_{\mathsf{fail}}$ must return the same k.

If there exists $(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c}) \in T_{\mathcal{P}}$ with $\tilde{\mathsf{c}}_1$ valid and c invalid, then it must be inserted by $\mathcal{P}^{-1}$, and condition $\mathsf{C}_{2\text{-}\mathcal{P}^{-1}}$ is necessarily triggered. Otherwise, if $\tilde{\mathsf{c}}$ is also invalid, then, as noted earlier,

$$\mathcal{S}_{\mathsf{fail}}.\mathcal{G}^{\mathsf{add}}(\tilde{\mathsf{c}}, \tilde{\mathsf{c}}) = \perp.$$

Now consider any private key $\mathsf{sk} \in \mathbb{Z}_N$ and any valid ciphertext c. When the differentiator derives the key k corresponding to $(\mathsf{sk}, \mathsf{c})$, it must query $\mathcal{H}_{\mathsf{key}}$ with $(\mathsf{pk}, \mathsf{seed})$, where $\mathsf{pk} = \mathbf{KEM}.\mathsf{KG}(\mathsf{sk})$ and seed

corresponds to $(\mathsf{pk}, \mathsf{r})$ of c. If the differentiator knows r, the consistency of $\mathbf{KEM}.\mathsf{Encaps}$ already ensures that

$$\mathcal{H}_{\mathsf{key}}(\mathsf{pk}, \mathsf{seed}) = \mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c}).$$

Otherwise, the differentiator must generates c through $\mathcal{S}_{\mathsf{fail}}.\mathcal{P}(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2)$, and there exist the following associations:

- The exists $(\mathsf{seed}, y) \in T_{\mathcal{H}_y}$.
- The equation $(\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0)$ lies in the span of T_{eq}.
- There exists str with equation $(\mathsf{str} - y \cdot \mathsf{pk} = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$ and $(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask}) \in T_{\mathcal{H}_{\mathsf{mask}}}$ such that $\mathsf{seed} \oplus \mathsf{mask} = \tilde{\mathsf{c}}_2$.

In such case, Proposition B.17 ensures that

$$\mathcal{H}_{\mathsf{key}}(\mathsf{pk}, \mathsf{seed}) = \mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c}).$$

Hence, as long as the simulator $\mathcal{S}_{\mathsf{fail}}$ does not fail in either Game 2 or Game 3, the differentiator's view is statistically indistinguishable between the two games. Let q_{hon} and q_{adv} denote the number of queries made to the honest and adversarial interfaces, with $q_{\mathrm{hon}} + q_{\mathrm{adv}} \leq q$. Furthermore, each query to the honest interface in Game 3 requires at most λ queries to the simulator. Then,

$$\begin{aligned} |\Pr[\mathbf{G}_3(\mathcal{D}) = 1] - \Pr[\mathbf{G}_2(\mathcal{D}) = 1]| &\leq 2 \cdot \Pr[\mathcal{S}_{\mathsf{fail}} \text{ fails}] \\ &\leq O\left(\frac{(\lambda \cdot q)^4}{N} + \frac{(\lambda \cdot q)^3}{2^\lambda}\right). \end{aligned}$$

□
□

B.5 Transition from $\mathbf{G}_3$ to $\mathbf{G}_4$

Game 4. In this game, we modify the simulator such that its responses no longer depend on the ideal KEM. Specifically, the new simulator $\mathcal{S}_{\mathsf{samp}}$ implements each interface using lazy sampling, and all failure conditions are removed from $\mathcal{S}_{\mathsf{samp}}$.

By $\mathsf{Coin} \xleftarrow{\$} \mathrm{Ber}(p)$ we denote that a single bit Coin is sampled from a Bernoulli distribution with success probability p (i.e., $\Pr[\mathsf{Coin} = 1] = p$). We write $\mathsf{Ele}(T_{\mathsf{eq}})$ for the set of all group elements appearing in T_{eq}.

Simulator $\mathcal{S}_{\mathsf{samp}}$

$\mathcal{S}_{\mathsf{samp}}.\mathcal{H}_x(\mathsf{sk})$:

if $\exists(\mathsf{sk}, x) \in T_{\mathcal{H}_x}$, then **return** x;

$x \xleftarrow{\$} \mathbb{Z}_N$; $T_{\mathcal{H}_x} \leftarrow T_{\mathcal{H}_x} \cup \{(\mathsf{sk}, x)\}$;

return x;

$\mathcal{S}_{\mathsf{samp}}.\mathcal{H}_{\mathsf{seed}}(\mathsf{pk}, \mathsf{r})$:

if $\exists(\mathsf{pk}||\mathsf{r}, \mathsf{seed}) \in T_{\mathcal{H}_{\mathsf{seed}}}$, then **return** seed;

$\mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$; $T_{\mathcal{H}_{\mathsf{seed}}} \leftarrow T_{\mathcal{H}_{\mathsf{seed}}} \cup \{\mathsf{pk}||\mathsf{r}, \mathsf{seed}\}$;

return seed;

$\underline{\mathcal{S}_{\mathsf{samp}}.\mathcal{H}_y(\mathsf{seed}):}$

if $\exists(\mathsf{seed}, y) \in T_{\mathcal{H}_y}$, then **return** y;

$y \xleftarrow{\$} \mathbb{Z}_N$; $T_{\mathcal{H}_y} \leftarrow T_{\mathcal{H}_y} \cup \{(\mathsf{seed}, y)\}$;

return y;

$\underline{\mathcal{S}_{\mathsf{samp}}.\mathcal{H}_{\mathsf{mask}}(\tilde{\mathsf{c}}_1||\mathsf{str}):}$

if $\exists(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask}) \in T_{\mathcal{H}_{\mathsf{mask}}}$, then **return** mask;

$\mathsf{mask} \xleftarrow{\$} \{0,1\}^\lambda$; $T_{\mathcal{H}_{\mathsf{mask}}} \leftarrow T_{\mathcal{H}_{\mathsf{mask}}} \cup \{(\tilde{\mathsf{c}}_1||\mathsf{str}, \mathsf{mask})\}$;

return mask;

$\underline{\mathcal{S}_{\mathsf{samp}}.\mathcal{H}_{\mathsf{key}}(\mathsf{pk}, \mathsf{seed}):}$

if $\exists(\mathsf{pk}||\mathsf{seed}, \mathsf{k}) \in T_{\mathcal{H}_{\mathsf{key}}}$, then **return** k;

$\mathsf{k} \xleftarrow{\$} \mathcal{K}$; $T_{\mathcal{H}_{\mathsf{key}}} \leftarrow T_{\mathcal{H}_{\mathsf{key}}} \cup \{(\mathsf{pk}||\mathsf{seed}, \mathsf{k})\}$;

return k;

$\underline{\mathcal{S}_{\mathsf{samp}}.\mathcal{P}(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2):}$

if $\exists(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c}) \in T_{\mathcal{P}}$, then **return** c;

$\mathsf{c} \xleftarrow{\$} \{0,1\}^{\ell+\lambda}$; $T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c})\}$;

return c;

$\underline{\mathcal{S}_{\mathsf{samp}}.\mathcal{P}^{-1}(\mathsf{c}):}$

if $\exists(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c}) \in T_{\mathcal{P}}$, then **return** $\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2$;

$\tilde{\mathsf{c}}_1 \xleftarrow{\$} \{0,1\}^\ell$; $\tilde{\mathsf{c}}_2 \xleftarrow{\$} \{0,1\}^\lambda$; $T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2, \mathsf{c})\}$;

return $\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2$;

$\underline{\mathcal{S}_{\mathsf{samp}}.\mathcal{G}^{\mathsf{label}}(x):}$

if $\exists(\mathsf{str} - x \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$, then **return** str;

$\mathsf{str} \xleftarrow{\$} \{0,1\}^\ell$; $T_{\mathsf{eq}} \leftarrow T_{\mathsf{eq}} \cup \{\mathsf{str} - x \cdot \diamondsuit = 0\}$;

return str;

$\underline{\mathcal{S}_{\mathsf{samp}}.\mathcal{G}^{\mathsf{add}}(\mathsf{str}_1, \mathsf{str}_2):}$

if $\mathsf{str}_1 \in T_\perp \vee \mathsf{str}_2 \in T_\perp$: return $\perp$;

if $\exists(\mathsf{str} - \mathsf{str}_1 - \mathsf{str}_2 = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$, then **return** str;

for str_i that does not appear in T_{eq} with $i \in \{1,2\}$:

$\mathsf{Coin} \leftarrow \mathrm{Ber}(\frac{N-|\mathsf{Ele}(T_{\mathsf{eq}})|}{2^\ell-|\mathsf{Ele}(T_{\mathsf{eq}})|-|T_\perp|})$;

if $\mathsf{Coin} = 0$: $T_\perp \leftarrow T_\perp \cup \{\mathsf{str}_i\}$; return $\perp$;

$\mathsf{str} \xleftarrow{\$} \{0,1\}^\ell$; $T_{\mathsf{eq}} \leftarrow T_{\mathsf{eq}} \cup \{\mathsf{str} - \mathsf{str}_1 - \mathsf{str}_2 = 0\}$;

return str.

We now show that the views of the differentiator in Game 3 and Game 4 are statistically close. Recall that the only difference between the two games lies in how the simulator answers queries. Thus, the proof reduces to analyzing, for each interface, whether the responses generated by $\mathcal{S}_{\mathsf{samp}}$ faithfully reproduce the distribution enforced by $\mathcal{S}_{\mathsf{fail}}$, and whether any statistical deviation can only occur through rare collision events.

Responses from the $\mathcal{H}_x$ and $\mathcal{H}_y$ interfaces. For both $\mathcal{H}_x$ and $\mathcal{H}_y$ queries, the responses generated by $\mathcal{S}_{\mathsf{fail}}.\mathcal{H}_x$ and $\mathcal{S}_{\mathsf{samp}}.\mathcal{H}_x$ are sampled uniformly from $\mathbb{Z}_N$. Hence, their output distributions are identical.

Response from the $\mathcal{H}_{\mathsf{seed}}$ interface. For a $\mathcal{H}_{\mathsf{seed}}$ query with input $(\mathsf{pk}, \mathsf{r})$, assuming no failure condition is triggered, the only bad event is that the sampled seed in Game 4 collides with an entry in $T_{\mathcal{H}_{\mathsf{seed}}}$ or $T_{\mathcal{H}_{\mathsf{key}}}$. The probability of such a collision is at most $q/2^\lambda$.

Response from the $\mathcal{H}_{\mathsf{mask}}$ interface. For a $\mathcal{H}_{\mathsf{mask}}$ query with input $(\mathsf{pk}, \mathsf{seed})$, assuming no failure condition is triggered, two types of bad events may occur: 1) the sampled mask in Game 4 collides with an entry in $T_{\mathcal{H}_{\mathsf{seed}}}$ or $T_{\mathcal{H}_{\mathsf{key}}}$, which happens with probability at most $q/2^\lambda$; 2) there exist seed in $T_{\mathcal{H}_{\mathsf{seed}}}$, $T_{\mathcal{H}_y}$, or $T_{\mathcal{H}_{\mathsf{key}}}$ and $\tilde{\mathsf{c}}_2$ in $T_{\mathcal{P}}$ such that $\mathsf{seed} = \mathsf{mask} \oplus \tilde{\mathsf{c}}_2$, which occurs with probability at most $q^2/2^\lambda$.

Response from the $\mathcal{H}_{\mathsf{key}}$ interface. For a $\mathcal{H}_{\mathsf{key}}$ query with input $(\mathsf{pk}, \mathsf{seed})$, the simulator in Game 4 generates the response k by sampling uniformly from $\mathcal{K}$ rather than invoking $\mathbf{KEM}.\mathsf{Encaps}$ or $\mathbf{KEM}.\mathsf{Decaps}$. Since the ideal KEM is itself a random mapping into $\mathcal{K}$, and provided that no failure condition is triggered in Game 3, the resulting output distributions are identical.

Response from the $\mathcal{P}$ interface. For a $\mathcal{P}$ query with input $\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2$, the only difference is that $\mathcal{S}_{\mathsf{samp}}$ generates the response c by sampling uniformly from $\{0,1\}^{\ell+\lambda}$ rather than invoking Encaps of $\mathbf{KEM}$. Since the ideal KEM is itself a random injection to $\{0,1\}^{\ell+\lambda}$, and provided that no failure condition is triggered in Game 3, the only bad event here is that the sample c in Game 4 collides with a recorded ciphertext in $T_{\mathcal{P}}$, which occurs with probability at most $q/2^{\ell+\lambda}$.

Response from the $\mathcal{P}^{-1}$ interface. For a $\mathcal{P}^{-1}$ query with input c, the only difference is that $\mathcal{S}_{\mathsf{samp}}$ generates the response $\tilde{\mathsf{c}}_1||\tilde{\mathsf{c}}_2$ by sampling uniformly at random. Assuming no failure condition is triggered in Game 3, three types of bad events may occur: 1) the sampled $\tilde{\mathsf{c}}_1$ collides with an entry in T_{eq}, $T_{\mathcal{P}}$, or $T_{\mathcal{H}_{\mathsf{mask}}}$, which happens with probability at most $2q/2^\ell$; 2) the sampled $\tilde{\mathsf{c}}_2$ collides with an entry in $T_{\mathcal{P}}$, which happens with probability at most $q/2^\lambda$; 3) there exist seed in $T_{\mathcal{H}_{\mathsf{seed}}}$, $T_{\mathcal{H}_y}$, or $T_{\mathcal{H}_{\mathsf{key}}}$ and mask in $T_{\mathcal{H}_{\mathsf{mask}}}$ such that $\mathsf{seed} \oplus \mathsf{mask} = \tilde{\mathsf{c}}_2$, which occurs with probability at most $q^2/2^\lambda$.

Response from the $\mathcal{G}^{\mathsf{label}}$ interface. For a labeling query with input x, the only difference is that $\mathcal{S}_{\mathsf{samp}}$ generates the group element by sampling uniformly from $\{0,1\}^\ell$ rather than invoking $\mathbf{KEM}.\mathsf{KG}$. Assuming that no failure condition is triggered, the only bad event here is that the sampled str falls into T_{eq} or $T_\perp$ (the set recording all currently considered invalid group encodings). The probability of this event is at most $3q/2^\ell$.

Response from the $\mathcal{G}^{\mathsf{add}}$ interface. For an addition query with input $(\mathsf{str}_1, \mathsf{str}_2)$, the simulator $\mathcal{S}_{\mathsf{samp}}$ decides whether an unknown encoding is valid by sampling from a Bernoulli distribution with probability $\frac{N-|\mathsf{Ele}(T_{\mathsf{eq}})|}{2^\ell-|\mathsf{Ele}(T_{\mathsf{eq}})|-|T_\perp|}$ of being valid, rather than querying $\mathbf{KEM}.\mathsf{Encaps}$. These two approaches yield identical distributions. Assuming no failure condition is triggered in Game 3, the only bad event is that the sampled response str lies in T_{eq} or $T_\perp$, which occurs with probability at most $3q/2^\ell$.

Therefore, if the simulator $\mathcal{S}_{\mathsf{fail}}$ does not fail in Game 3 and no response generated by $\mathcal{S}_{\mathsf{samp}}$ in Game 4 triggers any of the bad events above, the distinguisher's view remains statistically close between the two games. Let $\mathbf{G}_4$ denote Game 4, which outputs 1 if and only if the differentiator outputs 1 after interacting with Π_{KEM} and $\mathcal{S}_{\mathsf{samp}}$. Formally,

$$\Pr[\mathbf{G}_4(\mathcal{D}) = 1] = \Pr[\mathcal{D}^{\Pi_{\mathsf{KEM}}^{\mathcal{S}_{\mathsf{samp}}}, \mathcal{S}_{\mathsf{samp}}}(1^\lambda) = 1].$$

Then,

$$\begin{aligned}|\Pr[\mathbf{G}_4(\mathcal{D}) = 1] - \Pr[\mathbf{G}_3(\mathcal{D}) = 1]| &\leq O\left(\frac{(\lambda \cdot q)^4}{N} + \frac{(\lambda \cdot q)^3}{2^\lambda}\right) + q \cdot \frac{2q^2 + 12q}{2^\lambda} \\ &= O\left(\frac{(\lambda \cdot q)^4}{N} + \frac{(\lambda \cdot q)^3}{2^\lambda}\right).\end{aligned}$$

B.6 Transition from $\mathbf{G}_4$ to Real

Real Game. By replacing the simulator $\mathcal{S}_{\mathsf{samp}}$ in Game 4 with the real idealized oracles

$$\mathcal{O} = (\mathcal{H}_x, \mathcal{H}_{\mathsf{seed}}, \mathcal{H}_y, \mathcal{H}_{\mathsf{mask}}, \mathcal{H}_{\mathsf{key}}, \mathcal{P}, \mathcal{P}^{-1}, \mathcal{G}^{\mathsf{label}}, \mathcal{G}^{\mathsf{add}}),$$

we arrive at the final game of our proof, which we refer to as the real game. In this game, the differentiator $\mathcal{D}$ interacts with $\Pi_{\mathsf{KEM}}^{\mathcal{O}}$ and $\mathcal{O}$. The real game outputs 1 if and only if $\mathcal{D}$ outputs 1. Formally,

$$\Pr[\mathbf{Real}^{\Pi_{\mathsf{KEM}}^{\mathcal{O}}}(\mathcal{D}) = 1] = \Pr[\mathcal{D}^{\Pi_{\mathsf{KEM}}^{\mathcal{O}}, \mathcal{O}}(1^\lambda) = 1].$$

We now demonstrate that the views of the differentiator in Game 4 and the Real Game are statistically close.

For the random oracle queries $\mathcal{H}_x$, $\mathcal{H}_{\mathsf{seed}}$, $\mathcal{H}_y$, $\mathcal{H}_{\mathsf{mask}}$, and $\mathcal{H}_{\mathsf{key}}$, the responses in Game 4 and Real Game are uniformly distributed. For the random permutation queries $\mathcal{P}$ and $\mathcal{P}^{-1}$, the differentiator may observe a difference if the uniformly sampled response by $\mathcal{S}_{\mathsf{samp}}$ collides with the existing one in $T_{\mathcal{P}}$, each with probability at most $q/2^{\ell+\lambda}$. For labeling and addition queries, the simulator in Game 4 uses T_{eq} to emulate the algebraic structure of the GGM and determines the validity of group encodings via the Bernoulli distribution and $T_\perp$. Specifically, an unknown group encoding is deemed valid with probability

$$\frac{N - |\mathsf{Ele}(T_{\mathsf{eq}})|}{2^\ell - |\mathsf{Ele}(T_{\mathsf{eq}})| - |T_\perp|}.$$

Since $\mathsf{Ele}(T_{\mathsf{eq}})$ and $T_\perp$ contain all valid and invalid encodings known to the differentiator, the views in the two games are identical. For each equation in T_{eq}, two cases arise:

Case 1: The equation is of the form $(\mathsf{str} - x \cdot \diamondsuit = 0)$, where str is uniformly sampled. A distinguishing difference may occur if str collides with a group element in T_{eq} with known discrete logarithm or if $\mathsf{str} \in T_\perp$, each with probability at most $3q/2^\ell$.

Case 2: The equation is of the form $(\mathsf{str} - \mathsf{str}_1 - \mathsf{str}_2 = 0)$, where str is uniformly sampled. A difference is observed if str appears in either T_{eq} or $T_\perp$, which occurs with probability at most $3q/2^\ell$.

Therefore, we deduce that

$$\left|\Pr[\mathbf{Real}^{\Pi_{\mathsf{KEM}}^{\mathcal{O}}}(\mathcal{D}) = 1] - \Pr[\mathbf{G}_4(\mathcal{D}) = 1]\right| \leq O\left(\frac{q^2}{2^\ell}\right).$$

Combining all the games above, we conclude that:

$$\left|\Pr[\mathbf{Real}^{\Pi_{\mathsf{KEM}}^{\mathcal{O}}}(\mathcal{D}) = 1] - \Pr[\mathbf{Ideal}_{\mathcal{S}}^{\mathbf{KEM}}(\mathcal{D}) = 1]\right| \leq O\left(\frac{(\lambda \cdot q)^4}{N} + \frac{(\lambda \cdot q)^3}{2^\lambda}\right).$$

□

B.7 Bounding the failure conditions

Note that the simulator $\mathcal{S}_{\mathsf{fail}}$ in Games 2 and 3 may fail under various conditions. In this section, we analyze the probability of each such failure occurring.

Lemma B.1. $\Pr[\text{Failure Conditions Hold}] \leq O\left(q^4/N + q^3/2^\lambda\right)$.

Proof. As described in Appendix B.2, the simulator evaluates potential failure conditions whenever it handles a $\mathcal{H}_x$, $\mathcal{H}_{\mathsf{seed}}$, $\mathcal{H}_y$, $\mathcal{H}_{\mathsf{mask}}$, $\mathcal{H}_{\mathsf{key}}$, $\mathcal{P}$, or $\mathcal{G}$ query.

When processing a $\mathcal{H}_x$ query on input sk, the simulator samples a value $x \in \mathbb{Z}_N$ uniformly at random as the candidate response. Let $\mathsf{pk} \leftarrow \mathbf{KEM}.\mathsf{KG}(\mathsf{sk})$. Six distinct failure conditions arise:

Condition $\mathsf{C}_{1\text{-}\mathcal{H}_x}$. The simulator fails if the differentiator queries with some sk such that sk already appears in T^*_{sk}. Since the simulator makes at most q queries to $\mathcal{S}_{\mathsf{fail}}$, each table maintained by $\mathcal{S}_{\mathsf{fail}}$ contains at most q entries. Thus T^*_{sk} can hold at most q private keys, each sampled uniformly at random and hidden from the differentiator. Accordingly, the probability of this event is bounded by

$$\Pr[\mathsf{C}_{1\text{-}\mathcal{H}_x}] \leq \frac{q}{N}.$$

Condition $\mathsf{C}_{2\text{-}\mathcal{H}_x}$. The simulator fails if the differentiator queries with some sk such that sk already appears in T^*_{c}. Since each private key in T^*_{c} is sampled uniformly at random and remains hidden from the differentiator, the probability of this event is bounded by

$$\Pr[\mathsf{C}_{2\text{-}\mathcal{H}_x}] \leq \frac{q}{N}.$$

Condition $\mathsf{C}_{3\text{-}\mathcal{H}_x}$. The simulator fails if a group element already exists with x as its discrete logarithm. As noted earlier, there are at most q discrete logarithms of group elements in T_{eq}. Therefore, the probability that a uniformly sampled x collides with one of these discrete logarithms is bounded by

$$\Pr[\mathsf{C}_{3\text{-}\mathcal{H}_x}] \leq \frac{q}{N}.$$

Condition $\mathsf{C}_{4\text{-}\mathcal{H}_x}$. The simulator fails if there exist two group elements str_1 and str_2 such that the uniformly sampled x satisfies $(\mathsf{str}_1 - x \cdot \mathsf{str}_2 = 0)$ in $\mathsf{Span}(T_{\mathsf{eq}})$. Assuming Proposition B.2 holds, each pair $(\mathsf{str}_1, \mathsf{str}_2)$ can generate at most one equation of the form $(\mathsf{str}_1 - \alpha \cdot \mathsf{str}_2 = 0)$ in $\mathsf{Span}(T_{\mathsf{eq}})$. Hence, the probability that the uniformly sampled x collides with one of these values α is bounded by

$$\Pr\left[\mathsf{C}_{4\text{-}\mathcal{H}_x} | \text{Proposition B.2 holds}\right] \leq \frac{9q^2}{N}.$$

Condition $\mathsf{C}_{5\text{-}\mathcal{H}_x}$. The simulator fails if there exist two group elements str_1 and str_2 such that the equation $(\mathsf{str}_1 - \alpha \cdot \mathsf{str}_2 - \beta \cdot \mathsf{pk} - \gamma \cdot \diamondsuit = 0)$ lies in the span of T_{eq}, and the uniformly sampled x satisfies $\beta \cdot x + \gamma = 0$. For each pair $(\mathsf{str}_1, \mathsf{str}_2)$, if more than two equations of this form exist, then pk must have a discrete logarithm in T_{eq}. By Proposition B.1, this implies that $(\mathsf{sk}, \mathsf{pk})$ exists in T^*_{sk}, which triggers the failure condition $\mathsf{C}_{1\text{-}\mathcal{H}_x}$. Therefore, assuming Proposition B.1 holds, the probability is bounded by

$$\Pr\left[\mathsf{C}_{5\text{-}\mathcal{H}_x} | \text{Proposition B.1 holds}\right] \leq 9q^2 \cdot \frac{2}{N} = \frac{18q^2}{N}.$$

Condition $\mathsf{C}_{6,7,8,9\text{-}\mathcal{H}_x}$. The simulator fails if there exists a group element str such that the equation $(\mathsf{str} - \alpha \cdot \mathsf{pk} - \beta \cdot \diamondsuit = 0)$ lies in the span of T_{eq}, and the uniformly sampled x yields a value $(\alpha \cdot x + \beta)$ that triggers a condition analogous to $\mathsf{C}_{3\text{-}\mathcal{H}_x}$, $\mathsf{C}_{4\text{-}\mathcal{H}_x}$, $\mathsf{C}_{5\text{-}\mathcal{H}_x}$, or already appears in $T_{\mathcal{H}_y}$. As noted earlier, there are at most $3q$ such group elements str. Provided that Proposition B.1 holds, for each str there is at most one equation of this form, yielding at most $3q$ uniformly random values. Accordingly, the probabilities are bounded by:

$$\Pr\left[\mathsf{C}_{6\text{-}\mathcal{H}_x} | \text{Proposition B.1 holds}\right] \leq 3q \cdot \frac{q}{N} = \frac{3q^2}{N},$$

$$\Pr\left[\mathsf{C}_{7\text{-}\mathcal{H}_x} | \text{Proposition B.2 holds}\right] \leq 3q \cdot \frac{9q^2}{N} = \frac{27q^3}{N},$$

$$\Pr\left[\mathsf{C}_{8\text{-}\mathcal{H}_x} | \text{Proposition B.1 holds}\right] \leq 3q \cdot \frac{q}{N} = \frac{3q^2}{N},$$

and

$$\Pr\left[\mathsf{C}_{9\text{-}\mathcal{H}_x} | \text{Proposition B.1 holds}\right] \leq 3q \cdot \frac{18q^2}{N} = \frac{54q^3}{N},$$

When processing a $\mathcal{H}_{\mathsf{seed}}$ query with input pk and r, if the response is not already determined by the relationships among other interfaces, the simulator samples seed uniformly at random as the candidate response, and two distinct failure conditions arise:

Condition $\mathsf{C}_{1\text{-}\mathcal{H}_{\mathsf{seed}}}$. The simulator fails if the differentiator queries $\mathcal{H}_{\mathsf{seed}}$ on some r that already appears in T^*_{r}. Since each randomness in T^*_{r} is sampled uniformly at random and remains hidden from the differentiator, the probability of this event is bounded by

$$\Pr[\mathsf{C}_{1\text{-}\mathcal{H}_{\mathsf{seed}}}] \leq \frac{q}{2^\lambda}.$$

Condition $\mathsf{C}_{2\text{-}\mathcal{H}_{\mathsf{seed}}}$. The simulator fails if the uniformly sampled seed already exists in $T_{\mathcal{H}_{\mathsf{seed}}}$, $T_{\mathcal{H}_{\mathsf{key}}}$ or T^*_{r}. The probability is bounded by

$$\Pr[\mathsf{C}_{2\text{-}\mathcal{H}_{\mathsf{seed}}}] \leq \frac{q}{2^\lambda}.$$

When processing a $\mathcal{H}_y$ query with input seed, the simulator samples a value $y \in \mathbb{Z}_N$ uniformly at random as the candidate response. If there exists $(\tilde{\mathsf{c}}_1 || \tilde{\mathsf{c}}_2, \mathsf{c}) \in T_{\mathcal{P}}$ associated with seed, the simulator additionally inserts the equation $(\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0)$ into T_{eq}. Six distinct failure conditions arise:

Condition $\mathsf{C}_{1\text{-}\mathcal{H}_y}$. The simulator fails if the uniformly sampled value y already exists in $T_{\mathcal{H}_y}$. As noted previously, $T_{\mathcal{H}_y}$ contains at most q tuples. Thus, the probability that a random y collides with an existing value is bounded by

$$\Pr[\mathsf{C}_{1\text{-}\mathcal{H}_y}] \leq \frac{q}{N}.$$

Condition $\mathsf{C}_{2\text{-}\mathcal{H}_y}$. The simulator fails if a group element already exists with y as its discrete logarithm. As noted previously, T_{eq} contains at most q equations, leading to at most q discrete logarithms of group elements in T_{eq}. Therefore, the probability that a uniformly sampled y collides with one of these discrete logarithms is bounded by

$$\Pr[\mathsf{C}_{2\text{-}\mathcal{H}_y}] \leq \frac{q}{N}.$$

Condition $\mathsf{C}_{3\text{-}\mathcal{H}_y}$. The simulator fails if there exist two group elements str_1 and str_2 such that the uniformly sampled y satisfies $(\mathsf{str}_1 - y \cdot \mathsf{str}_2 = 0)$ in $\mathsf{Span}(T_{\mathsf{eq}})$. By Proposition B.2, for each pair $(\mathsf{str}_1, \mathsf{str}_2)$ there is at most one equation of the form $(\mathsf{str}_1 - \alpha \cdot \mathsf{str}_2 = 0)$ in $\mathsf{Span}(T_{\mathsf{eq}})$. Hence, the probability that the uniformly sampled y collides with one of these values α is bounded by

$$\Pr\left[\mathsf{C}_{3\text{-}\mathcal{H}_y} | \text{Proposition B.2 holds}\right] \leq \frac{9q^2}{N}.$$

Condition $\mathsf{C}_{4\text{-}\mathcal{H}_y}$. The simulator fails if there exist two group elements $\mathsf{str}_1, \mathsf{str}_2$ such that the equation $(\mathsf{str}_1 - \alpha \cdot \mathsf{str}_2 - \beta \cdot \tilde{\mathsf{c}}_1 - \gamma \cdot \diamondsuit = 0)$ lies in the span of T_{eq}, and the uniformly sampled y satisfies the linear relation $\beta \cdot y + \gamma = 0$. Similarly, By Proposition B.1, the probability is bounded by

$$\Pr\left[\mathsf{C}_{4\text{-}\mathcal{H}_y} | \text{Proposition B.1 holds}\right] \leq \frac{18q^2}{N}.$$

Conditions $\mathsf{C}_{5,6,7\text{-}\mathcal{H}_y}$. The simulator fails if there exists a group element str such that the equation $(\mathsf{str} - \alpha \cdot \tilde{\mathsf{c}}_1 - \beta \cdot \diamondsuit = 0)$ lies in the span of T_{eq}, and the uniformly sampled y yields a value $(\alpha \cdot y + \beta)$ that triggers

conditions analogous to $\mathsf{C}_{1\text{-}\mathcal{H}_y}$, $\mathsf{C}_{2\text{-}\mathcal{H}_y}$, or $\mathsf{C}_{3\text{-}\mathcal{H}_y}$. As noted earlier, there are at most $3q$ group elements str. Proposition B.1 ensures that, for each str, there is at most one equation of the form $(\mathsf{str} - \alpha \cdot \tilde{\mathsf{c}} - \beta \cdot \diamondsuit = 0)$. Accordingly, the probabilities are bounded by

$$\Pr[\mathsf{C}_{5\text{-}\mathcal{H}_y} | \text{Proposition B.1 holds}] \leq 3q \cdot \frac{q}{N} = \frac{3q^2}{N},$$

$$\Pr[\mathsf{C}_{6\text{-}\mathcal{H}_y} | \text{Proposition B.1 holds}] \leq 3q \cdot \frac{q}{N} = \frac{3q^2}{N},$$

and

$$\Pr[\mathsf{C}_{5\text{-}\mathcal{H}_y} | \text{Proposition B.2 holds}] \leq 3q \cdot \frac{9q^2}{N} = \frac{27q^3}{N}.$$

When processing a $\mathcal{H}_{\mathsf{mask}}$ query with input $\tilde{\mathsf{c}}_1$ and str, if the response is not already determined by the relationships among other interfaces, the simulator samples mask uniformly at random as the candidate response, and two distinct failure conditions arise:

Condition $\mathsf{C}_{1\text{-}\mathcal{H}_{\mathsf{mask}}}$. The simulator fails if the uniformly sampled mask already exists in $T_{\mathcal{H}_{\mathsf{mask}}}$. The probability is bounded by

$$\Pr[\mathsf{C}_{1\text{-}\mathcal{H}_{\mathsf{mask}}}] \leq \frac{q}{2^\lambda}.$$

Condition $\mathsf{C}_{2\text{-}\mathcal{H}_{\mathsf{mask}}}$. The simulator fails if there exist seed in $T_{\mathcal{H}_{\mathsf{seed}}}$, $T_{\mathcal{H}_y}$, or $T_{\mathcal{H}_{\mathsf{key}}}$ and $\tilde{\mathsf{c}}_2$ in $T_{\mathcal{P}}$, and the uniformly sampled mask satisfies $\mathsf{seed} \oplus \mathsf{mask} = \tilde{\mathsf{c}}_2$. The probability is bounded by

$$\Pr[\mathsf{C}_{2\text{-}\mathcal{H}_{\mathsf{mask}}}] \leq \frac{q^2}{2^\lambda}.$$

When processing a $\mathcal{H}_{\mathsf{key}}$ query with input pk and seed, if pk is valid but the response is not already determined by the relationships among other interfaces, the simulator generates a fresh k by invoking $\mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$, where r is sampled uniformly at random, and one failure condition arises:

Condition $\mathsf{C}_{1\text{-}\mathcal{H}_{\mathsf{mask}}}$. The simulator fails if the uniformly sampled r already appears in $T_{\mathcal{H}_{\mathsf{seed}}}$, T^*_{r}, or T^*_{c}. The probability is bounded by

$$\Pr[\mathsf{C}_{1\text{-}\mathcal{H}_{\mathsf{key}}}] \leq \frac{q}{2^\lambda}.$$

When processing a $\mathcal{P}$ query with input $\tilde{\mathsf{c}}_1 || \tilde{\mathsf{c}}_2$, the simulator first checks the validity of $\tilde{\mathsf{c}}_1$ using $\mathbf{KEM}.\mathsf{Encaps}$. If $\tilde{\mathsf{c}}_1$ is invalid, the simulator samples an invalid ciphertext c as the candidate response and two distinct failure conditions arise:

Condition $\mathsf{C}_{1\text{-}\mathcal{P}}$. The simulator fails if the uniformly sampled ciphertext c already exists in $T_{\mathcal{P}}$. The probability is bounded by

$$\Pr[\mathsf{C}_{1\text{-}\mathcal{P}}] \leq \frac{q}{2^{\ell+\lambda}}.$$

Condition $\mathsf{C}_{2\text{-}\mathcal{P}}$. The simulator fails if, even after λ iterations, the uniformly sampled ciphertext c remains valid. The probability is bounded by

$$\Pr[\mathsf{C}_{2\text{-}\mathcal{P}}] \leq \left(\frac{N \cdot 2^\lambda}{2^{\ell+\lambda}}\right)^\lambda \leq \frac{1}{2^\lambda}.$$

For a valid $\tilde{\mathsf{c}}_1$, if there exist seed and mask with $\tilde{\mathsf{c}}_1 = \mathsf{seed} \oplus \mathsf{mask}$ such that the specific correlations are satisfied and corresponding to a pk. Then the response should be a ciphertext under pk. If there is no corresponding randomness r, the simulator uniformly samples r, generates response c by querying $\mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$, and one failure condition arises:

Condition $\mathsf{C}_{3\text{-}\mathcal{P}}$. The simulator fails if the uniformly sampled value $\mathsf{r} \in \{0,1\}^\lambda$ already exists in $T_{\mathcal{H}_{\mathsf{seed}}}$, T^*_{r}, or T^*_{c}. The probability is bounded by

$$\Pr[\mathsf{C}_{3\text{-}\mathcal{P}}] \leq \frac{q}{2^\lambda}.$$

Otherwise, the simulator generates a fresh c by sampling sk and r uniformly, and one more failure condition arises:

Condition $\mathsf{C}_{4\text{-}\mathcal{P}}$. The simulator fails if the uniformly sampled value $\mathsf{sk} \in \mathbb{Z}_N$ already exists in $T_{\mathcal{H}_x}$. The probability is bounded by

$$\Pr\left[\mathsf{C}_{4\text{-}\mathcal{P}}\right] \leq \frac{q}{N}.$$

When processing a $\mathcal{P}^{-1}$ query with input c, the simulator first checks the validity of c using $\mathbf{KEM}.\mathsf{Decaps}$. If c is invalid, the simulator samples an invalid $\tilde{\mathsf{c}}_1$ and $\tilde{\mathsf{c}}_2$ as the candidate response and two distinct failure conditions arise:

Condition $\mathsf{C}_{1\text{-}\mathcal{P}^{-1}}$. The simulator fails if the uniformly sampled $\tilde{\mathsf{c}}_2 \in \{0,1\}^\lambda$ already exists in $T_{\mathcal{P}}$. The probability is bounded by

$$\Pr\left[\mathsf{C}_{1\text{-}\mathcal{P}^{-1}}\right] \leq \frac{q}{2^\lambda}.$$

Condition $\mathsf{C}_{2\text{-}\mathcal{P}^{-1}}$. The simulator fails if, even after λ iterations, the uniformly sampled public key $\tilde{\mathsf{c}}_1$ remains valid. The probability is bounded by

$$\Pr\left[\mathsf{C}_{2\text{-}\mathcal{P}^{-1}}\right] \leq \left(\frac{N}{2^\ell}\right)^\lambda \leq \frac{1}{2^\lambda}.$$

For a valid c, if only $\tilde{\mathsf{c}}_1$ is already determined by the relationships among other interfaces, the simulator uniformly samples $\tilde{\mathsf{c}}_2$, and one failure condition arises:

Condition $\mathsf{C}_{3\text{-}\mathcal{P}^{-1}}$. The simulator fails if there exist seed in $T_{\mathcal{H}_{\mathsf{seed}}}$, $T_{\mathcal{H}_y}$, or $T_{\mathcal{H}_{\mathsf{key}}}$ and mask in $T_{\mathcal{H}_{\mathsf{mask}}}$, and the uniformly sampled $\tilde{\mathsf{c}}_2$ satisfies $\mathsf{seed} \oplus \mathsf{mask} = \tilde{\mathsf{c}}_2$. The probability is bounded by

$$\Pr[\mathsf{C}_{3\text{-}\mathcal{P}^{-1}}] \leq \frac{q^2}{2^\lambda}.$$

If both $\tilde{\mathsf{c}}_1$ and $\tilde{\mathsf{c}}_2$ are not determined, the simulator generates fresh $\tilde{\mathsf{c}}_1$ and $\tilde{\mathsf{c}}_2$ with $\tilde{\mathsf{c}}_1 \leftarrow \mathbf{KEM}.\mathsf{KG}(\mathsf{sk})$ where sk is sampled uniformly at random, and three more distinct failure conditions arise:

Condition $\mathsf{C}_{4\text{-}\mathcal{P}^{-1}}$. The simulator fails if the randomly generated public key $\tilde{\mathsf{c}}_1$ already exists in T_{eq}, $T_{\mathcal{P}}$, or $T_{\mathcal{H}_{\mathsf{mask}}}$. The probability is bounded by

$$\Pr\left[\mathsf{C}_{4\text{-}\mathcal{P}^{-1}}\right] \leq \frac{3q}{N}.$$

Condition $\mathsf{C}_{5\text{-}\mathcal{P}^{-1}}$. The simulator fails if the uniformly sampled $\mathsf{sk} \in \mathbb{Z}_N$ already appears in $T_{\mathcal{H}_x}$. The probability is bounded by

$$\Pr\left[\mathsf{C}_{5\text{-}\mathcal{P}^{-1}}\right] \leq \frac{q}{N}.$$

Condition $\mathsf{C}_{6\text{-}\mathcal{P}^{-1}}$. The simulator fails if there exists $(\tilde{\mathsf{c}}_1 || \tilde{\mathsf{c}}_2, \mathsf{c})$ in $T_{\mathcal{P}}$ such that the uniformly sampled sk can decrypt c. Since each ciphertext corresponds to at most one proper private key, the probability is bounded by

$$\Pr\left[\mathsf{C}_{6\text{-}\mathcal{P}^{-1}}\right] \leq \frac{q}{N}.$$

When processing a $\mathcal{G}$ query with no corresponding entry in $\mathsf{Span}(T_{\mathsf{eq}})$, the simulator generates a fresh random public key by sampling sk uniformly as the candidate response. Let $\mathsf{str} \leftarrow \mathbf{KEM}.\mathsf{KG}(\mathsf{sk})$. Two distinct failure conditions may occur:

Condition $\mathsf{C}_{1\text{-}\mathcal{G}}$. The simulator fails if the randomly generated group element str already exists in T_{eq} or $T_{\mathcal{P}}$. The probability is bounded by

$$\Pr[\mathsf{C}_{1\text{-}\mathcal{G}}] \leq \frac{3q}{N}.$$

Condition $\mathsf{C}_{2\text{-}\mathcal{G}}$. The simulator fails if the sampled private key sk already exists in $T_{\mathcal{H}_x}$. Clearly, this probability is bounded by

$$\Pr[\mathsf{C}_{2\text{-}\mathcal{G}}] \leq \frac{q}{N}.$$

Combine together, we have

$$\Pr\left[\text{Failure Conditions Hold}\right] \leq= O\left(\frac{q^4}{N} + \frac{q^3}{2^\lambda}\right)$$

□
□

C Proof of Theorem 5.2

Theorem 5.2. $\Pi^{\mathcal{O}}_{\mathsf{PKE}}$ is indifferentiable from the ideal PKE with validity testing $\mathbf{PKE}$. More precisely, there exists a simulator $\mathcal{S}$ such that for all q-query differentiator $\mathcal{D}$, we have

$$\mathrm{Adv}^{\mathsf{indiff}}_{\Pi^{\mathcal{O}}_{\mathsf{PKE}},\mathbf{PKE},\mathcal{S}}(\mathcal{D}) \leq O\left(\frac{(\lambda\cdot q)^2}{N} + \frac{(\lambda\cdot q)^2}{2^\lambda}\right).$$

Here, the simulator makes at most $O(\lambda \cdot q)$ queries to its oracles.

Proof. According to the definition of indifferentiability, we have that, in the real world, the differentiator has three honest interfaces: $\Pi_{\mathsf{PKE}}.\mathsf{KG}$, $\Pi_{\mathsf{PKE}}.\mathsf{Enc}$, and $\Pi_{\mathsf{PKE}}.\mathsf{Dec}$, and adversarial interfaces including random oracles $\mathcal{H}$, random permutation $(\mathcal{P}, \mathcal{P}^{-1})$, key encapsulation mechanism $\mathbf{KEM}$, and authenticated encryption $(\mathcal{AE}, \mathcal{AD})$. Therefore, to complete the proof, we build an efficient simulator $\mathcal{S}$ in the ideal world, such that

1. $\mathcal{S}$ has access to the ideal PKE with validity testing $\mathbf{PKE}$;
2. $\mathcal{S}$ simulates those adversarial interfaces ($\mathcal{S}.\mathcal{H}$, $\mathcal{S}.\mathcal{P}$, $\mathcal{S}.\mathcal{P}^{-1}$, $\mathcal{S}.\mathbf{KEM}$, $\mathcal{S}.\mathcal{AE}$, $\mathcal{S}.\mathcal{AD}$).

To describe the simulator precisely, we first introduce some notation. Let $\mathsf{sk}_{\mathsf{test}} \in \mathcal{SK}$ be a fixed but otherwise arbitrary private key. Let $\mathsf{m}_{\mathsf{test}} \in \mathcal{M}$be a fixed but otherwise arbitrary message. Let $\mathsf{r}_{\mathsf{test}} \in \mathcal{R}$ be a fixed but otherwise arbitrary randomness. By $\mathsf{Coin} \xleftarrow{\$} \mathrm{Ber}(p)$, we mean that a single bit Coin is sampled from a Bernoulli distribution with probability p for $\mathsf{Coin} = 1$. We denote by $\mathsf{Ele}(T)$ the set of all ciphertexts $\mathsf{c}_\mathsf{k} \in \mathcal{C}_\mathsf{k}$ that appear in the table T.

In the following, we illustrate the full description of our simulator.

Simulator $\mathcal{S}$

$\mathcal{S}.\mathcal{H}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$:

if $\exists(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}}) \in T_{\mathcal{H}}$, then **return** $\tilde{\mathsf{r}}$; ..... Case 1

else:

$\tilde{\mathsf{r}} \xleftarrow{\$} \mathcal{R}$; $T_{\mathcal{H}} \leftarrow T_{\mathcal{H}} \cup \{(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}})\}$;

return $\tilde{\mathsf{r}}$. ..... Case 2

$\mathcal{S}.\mathbf{KEM}.\mathsf{KG}(\mathsf{sk})$:

return $\mathbf{PKE}.\mathsf{KG}(\mathsf{sk})$.

$\mathcal{S}.\mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk}, \tilde{\mathsf{r}})$:

if $\exists(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \in T_{\mathsf{KEM}}$, then **return** $(\mathsf{c_k}, \mathsf{k}||\mathsf{c})$; ..... Case 1

if $\mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m_{test}}, \mathsf{r_{test}}) = \perp_{\mathsf{pk}}$: **return** $\perp_{\mathsf{pk}}$; ..... Case 2

if $\exists(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}}) \in T_{\mathcal{H}}$:

$\mathsf{c} \leftarrow \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$;

if $\exists(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}) \in T_{\mathcal{P}}$:

if $\exists(\mathsf{sk}, \mathsf{pk}, \triangle, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \in T_{\mathsf{KEM}}$: replace $\triangle$ by $\tilde{\mathsf{r}}$;

return $(\mathsf{c_k}, \mathsf{k}||\mathsf{n})$; ..... Case 3

else: $\mathsf{k}||\mathsf{n} \xleftarrow{\$} \mathcal{K} \times \mathcal{N}$; $T_{\mathsf{KEM}} \leftarrow T_{\mathsf{KEM}} \cup \{(\triangle, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n})\}$;

return $(\mathsf{c_k}, \mathsf{k}||\mathsf{n})$; ..... Case 4

$(\mathsf{c_k}, \mathsf{k}, \mathsf{n}) \xleftarrow{\$} \mathcal{C_k} \times \mathcal{K} \times \mathcal{N}$; $T_{\mathsf{KEM}} \leftarrow T_{\mathsf{KEM}} \cup \{(\triangle, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n})\}$;

return $(\mathsf{c_k}, \mathsf{k}||\mathsf{n})$; ..... Case 5

$\mathcal{S}.\mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c_k})$:

if $\exists(\mathsf{sk}, *, *, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$, then **return** $\mathsf{k}||\mathsf{n}$; ..... Case 1

$\mathsf{pk} \leftarrow \mathbf{PKE}.\mathsf{KG}(\mathsf{sk})$;

if $\exists(\triangle, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \in T_{\mathsf{KEM}}$: replace $\triangle$ by sk;

return $\mathsf{k}||\mathsf{n}$; ..... Case 2

if $\exists(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}) \in T_{\mathcal{P}}$ s.t. $\mathbf{PKE}.\mathsf{Dec}(\mathsf{sk}, \mathsf{c}) \in \mathcal{M}$:

$(\mathsf{k}, \mathsf{n}) \xleftarrow{\$} \mathcal{K} \times \mathcal{N}$; $T_{\mathsf{KEM}} \leftarrow T_{\mathsf{KEM}} \cup \{(\mathsf{sk}, \mathsf{pk}, \triangle, \mathsf{c_k}, \mathsf{k}||\mathsf{n})\}$;

return $\mathsf{k}||\mathsf{n}$; ..... Case 3

if $\mathsf{c_k} \in T_{\perp}$, then **return** $\perp_{\mathsf{c}}$; ..... Case 4

if $\mathsf{c_k}$ does not appear in T_{KEM} and $T_{\mathcal{P}}$:

$b \xleftarrow{\$} \mathrm{Ber}(\frac{|\mathcal{R}| - |\mathsf{Ele}(T_{\mathsf{KEM}} \cup T_{\mathcal{P}} \cup T_{\perp})| + |\mathsf{Ele}(T_{\perp})|}{|\mathcal{C_k}| - |\mathsf{Ele}(T_{\mathsf{KEM}} \cup T_{\mathcal{P}} \cup T_{\perp})|})$;

if $b = 0$: $T_{\perp} \leftarrow T_{\perp} \cup \{\mathsf{c_k}\}$; **return** $\perp_{\mathsf{c}}$; ..... Case 5

$T_{\mathsf{KEM}} \leftarrow T_{\mathsf{KEM}} \cup \{(\mathsf{sk}, \mathsf{pk}, \triangle, \mathsf{c_k}, \perp)\}$;

return $\perp$; ..... Case 6

$\mathcal{S}.\mathcal{AE}(\mathsf{k}, \mathsf{n}, \mathsf{m})$:

if $\exists(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}) \in T_{\mathcal{AE}}$, then **return** $\mathsf{c_m}$; ..... Case 1

if $\exists(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \in T_{\mathsf{KEM}}$:

if $\exists(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}}) \in T_{\mathcal{H}}$:

$\mathsf{c} \leftarrow \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$;
if $\exists(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}) \in T_{\mathcal{P}}$: $T_{\mathcal{AE}} \leftarrow T_{\mathcal{AE}} \cup \{\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}\}$;
return $\mathsf{c_m}$; ..... Case 2

if $\exists(\mathsf{sk}, \mathsf{pk}, \triangle, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \in T_{\mathsf{KEM}}$:
if $\exists(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}) \in T_{\mathcal{P}}$ s.t. $\mathbf{PKE}.\mathsf{Dec}(\mathsf{sk}, \mathsf{c}) = \mathsf{m}$:
$T_{\mathcal{AE}} \leftarrow T_{\mathcal{AE}} \cup \{\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}\}$;
return $\mathsf{c_m}$; ..... Case 3

$\mathsf{c_m} \xleftarrow{\$} \mathcal{C}_{\mathsf{m}}$; $T_{\mathcal{AE}} \leftarrow T_{\mathcal{AE}} \cup \{(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})\}$;
return $\mathsf{c_m}$; ..... Case 4

$\underline{\mathcal{S}.\mathcal{AD}(\mathsf{k}, \mathsf{n}, \mathsf{c_m})}$:

if $\exists(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}) \in T_{\mathcal{AE}}$, then **return** m; ..... Case 1

if $\exists(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \in T_{\mathsf{KEM}}$:
if $\exists(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}}) \in T_{\mathcal{H}}$:
$\mathsf{c} \leftarrow \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$;
if $\exists(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}) \in T_{\mathcal{P}}$: $T_{\mathcal{AE}} \leftarrow T_{\mathcal{AE}} \cup \{(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})\}$;
return m; ..... Case 2

if $\exists(\mathsf{sk}, \mathsf{pk}, \triangle, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \in T_{\mathsf{KEM}}$:
if $\exists(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}) \in T_{\mathcal{P}}$: $\mathsf{m} \leftarrow \mathbf{PKE}.\mathsf{Dec}(\mathsf{sk}, \mathsf{c})$;
$T_{\mathcal{AE}} \leftarrow T_{\mathcal{AE}} \cup \{(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})\}$;
return m; ..... Case 3

$T_{\mathcal{AE}} \leftarrow T_{\mathcal{AE}} \cup \{(\mathsf{k}, \mathsf{n}, \perp, \mathsf{c_m})\}$;
return $\perp$; ..... Case 4

$\underline{\mathcal{S}.\mathcal{P}(\mathsf{c_k}||\mathsf{c_m})}$:

if $\exists(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}) \in T_{\mathcal{P}}$, then **return** c; ..... Case 1

if $\mathsf{c_k} \in T_{\perp}$:
for attempt $= 1$ to λ do: $\mathsf{c} \xleftarrow{\$} \mathcal{C}$;
if $\mathbf{PKE}.\mathsf{Dec}(\mathsf{sk_{test}}, \mathsf{c}) = \perp_{\mathsf{c}}$ **then break**;
$T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})\}$;
return c; ..... Case 2

if $\exists(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \in T_{\mathsf{KEM}}$:
if $\exists(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}) \in T_{\mathcal{AE}}$:
if $\exists(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}}) \in T_{\mathcal{H}}$: $\mathsf{c} \leftarrow \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$;

$T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})\}$;
return c; ..... Case 3

else: $\mathsf{r} \xleftarrow{\$} \mathcal{R}$; $\mathsf{c} \leftarrow \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$;
$T^*_{\mathsf{r}} \leftarrow T^*_{\mathsf{r}} \cup \{\mathsf{r}\}$; $T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})\}$;
return c; ..... Case 4

if $\exists(\mathsf{sk}, \mathsf{pk}, \triangle, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \in T_{\mathsf{KEM}}$:

if $\exists(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}) \in T_{\mathcal{AE}}$: $\mathsf{r} \xleftarrow{\$} \mathcal{R}$; $\mathsf{c} \leftarrow \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$;
$T^*_{\mathsf{r}} \leftarrow T^*_{\mathsf{r}} \cup \{\mathsf{r}\}$; $T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})\}$;
return c; ..... Case 5

if $\mathsf{c_k}$ does not appear in T_{KEM} and $T_{\mathcal{P}}$:

$b \xleftarrow{\$} \mathrm{Ber}(\frac{|\mathcal{R}|-|\mathsf{Ele}(T_{\mathsf{KEM}} \cup T_{\mathcal{P}} \cup T_{\perp})|+|\mathsf{Ele}(T_{\perp})|}{|\mathcal{C}_{\mathsf{k}}|-|\mathsf{Ele}(T_{\mathsf{KEM}} \cup T_{\mathcal{P}} \cup T_{\perp})|})$;
if $b = 0$: $T_{\perp} \leftarrow T_{\perp} \cup \{\mathsf{c_k}\}$;
for attempt $= 1$ to λ do:
$\mathsf{c} \xleftarrow{\$} \mathcal{C}$;
if $\mathbf{PKE}.\mathsf{Dec}(\mathsf{sk_{test}}, \mathsf{c}) = \perp_{\mathsf{c}}$ **then break**;
$T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})\}$;
return c; ..... Case 6

$(\mathsf{sk}, \mathsf{m}, \mathsf{r}) \xleftarrow{\$} \mathcal{SK} \times \mathcal{M} \times \mathcal{R}$; $\mathsf{c} \leftarrow \mathbf{PKE}.\mathsf{Enc}(\mathbf{PKE}.\mathsf{KG}(\mathsf{sk}), \mathsf{m}, \mathsf{r})$;
$T^*_{\mathsf{r}} \leftarrow T^*_{\mathsf{r}} \cup \{\mathsf{r}\}$; $T^*_{\mathsf{sk}} \leftarrow T^*_{\mathsf{sk}} \cup \{\mathsf{sk}\}$; $T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})\}$;
return c; ..... Case 7

$\underline{\mathcal{S}.\mathcal{P}^{-1}(\mathsf{c})}$:

if $\exists(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}) \in T_{\mathcal{P}}$, then **return** $\mathsf{c_k}||\mathsf{c_m}$; ..... Case 8

if $\mathbf{PKE}.\mathsf{Dec}(\mathsf{sk_{test}}, \mathsf{c}) = \perp_{\mathsf{c}}$: $(\mathsf{c_k}, \mathsf{c_m}) \xleftarrow{\$} \mathcal{C}_{\mathsf{k}} \times \mathcal{C}_{\mathsf{m}}$; $T_{\perp} \leftarrow T_{\perp} \cup \{\mathsf{c_k}\}$;
$T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})\}$;
return $\mathsf{c_k}||\mathsf{c_m}$; ..... Case 9

if $\exists(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}}) \in T_{\mathcal{H}}$ s.t. $\mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r}) = \mathsf{c}$:

if $\exists(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \in T_{\mathsf{KEM}}$:
if $\exists(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}) \in T_{\mathcal{AE}}$: $T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})\}$;
return $\mathsf{c_k}||\mathsf{c_m}$; ..... Case 10
else: $\mathsf{c_m} \xleftarrow{\$} \mathcal{C}_{\mathsf{m}}$; $T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})\}$;
return $\mathsf{c_k}||\mathsf{c_m}$; ..... Case 11

$(\mathsf{c_k}, \mathsf{c_m}) \xleftarrow{\$} \mathcal{C}_\mathsf{k} \times \mathcal{C}_\mathsf{m}$; $T_\mathcal{P} \leftarrow T_\mathcal{P} \cup \{(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})\}$;

return $\mathsf{c_k}||\mathsf{c_m}$. ..... Case 12

We immediately note that if the differentiator $\mathcal{D}$ makes q queries to the honest and adversarial interfaces, then the simulator $\mathcal{S}$ makes at most $O(\lambda \cdot q)$ number of queries to the ideal PKE with validity testing $\mathbf{PKE}$, indicating that the simulator $\mathcal{S}$ above is efficient.

Next, we establish that the view of $\mathcal{D}$ in the real world is indistinguishable from its view in the ideal world. To this end, we construct a sequence of indistinguishable hybrid games transitioning from the ideal game $\mathbf{Ideal}_\mathcal{S}^{\mathbf{PKE}}$ to the real game $\mathbf{Real}^{\Pi^\mathcal{O}_{\mathsf{PKE}}}$. Specifically,

1. **Ideal Game**. We begin with the ideal game $\mathbf{Ideal}_\mathcal{S}^{\mathbf{PKE}}$, in which $\mathcal{D}$ interacts with the honest interface $\mathbf{PKE}$ and the adversarial interface $\mathcal{S}^{\mathbf{PKE}}$.

2. **Game $\mathbf{G}_1$**. A relay algorithm $\mathcal{F}$ is introduced as an intermediary between $\mathcal{D}$ and ideal PKE. Here, $\mathcal{D}$ interacts with $\mathcal{F}^{\mathbf{PKE}}$ and $\mathcal{S}^{\mathbf{PKE}}$, where $\mathcal{F}$ simply forwards queries to $\mathbf{PKE}$ and returns the corresponding outputs.

3. **Game $\mathbf{G}_2$**. We refine the simulator by introducing several failure conditions on its query–response pairs. Whenever a failure condition is triggered, the modified simulator $\mathcal{S}_{\mathsf{fail}}$ aborts explicitly. In this game, $\mathcal{D}$ interacts with $\mathcal{F}^{\mathbf{PKE}}$ and $\mathcal{S}_{\mathsf{fail}}^{\mathbf{PKE}}$.

4. **Game $\mathbf{G}_3$**. The relay algorithm $\mathcal{F}$ is replaced with the construction Π_{PKE}, as defined earlier. The construction Π_{PKE} employs $\mathcal{S}_{\mathsf{fail}}$ in place of the ideal model. The central goal is to show that, except with negligible probability of triggering a failure condition (from Game 2), the responses generated by Π_{PKE} remain consistent with those of $\mathbf{PKE}$. In this game, $\mathcal{D}$ interacts with $\Pi_{\mathsf{PKE}}^{\mathcal{S}_{\mathsf{fail}}}$ and $\mathcal{S}_{\mathsf{fail}}^{\mathbf{PKE}}$.

5. **Game $\mathbf{G}_4$**. The simulator is further modified to sample its responses independently of the ideal PKE. Moreover, the new simulator $\mathcal{S}_{\mathsf{samp}}$ no longer checks for failure conditions. In this game, $\mathcal{D}$ interacts with $\Pi_{\mathsf{PKE}}^{\mathcal{S}_{\mathsf{samp}}}$ and $\mathcal{S}_{\mathsf{samp}}$.

6. **Real Game**. Finally, we arrive at the real game $\mathbf{Real}^{\Pi^\mathcal{O}_{\mathsf{PKE}}}$, in which the simulator $\mathcal{S}_{\mathsf{samp}}$ from Game 4 is replaced with the real oracle $\mathcal{O}$. Thus, $\mathcal{D}$ interacts with $\Pi_{\mathsf{PKE}}^{\mathcal{O}}$ and $\mathcal{O}$.

In what follows, we give a detailed description of each hybrid game and demonstrate that the statistical distance between any pair of consecutive games is negligible.

Ideal Game. In this game, the differentiator $\mathcal{D}$ interacts with the ideal PKE and the simulator $\mathcal{S}$, as defined previously. The game outputs 1 if and only if $\mathcal{D}$ outputs 1 after this interaction. Formally,

$$\Pr[\mathbf{Ideal}_\mathcal{S}^{\mathbf{PKE}}(\mathcal{D}) = 1] = \Pr[\mathcal{D}^{\mathbf{PKE},\mathcal{S}^{\mathbf{PKE}}}(1^\lambda) = 1].$$

C.1 Transition from Ideal to $\mathbf{G}_1$

Game 1. This game is identical to the ideal game, except that the differentiator $\mathcal{D}$ interacts with a relay algorithm $\mathcal{F} = (\mathcal{F}.\mathsf{KG}, \mathcal{F}.\mathsf{Enc}, \mathcal{F}.\mathsf{Dec})$ rather than directly with the ideal PKE. The relay algorithm $\mathcal{F}$ forwards all queries to the ideal PKE and provides the following two interfaces:

- $\mathcal{F}.\mathsf{KG}(\mathsf{sk})$: query $\mathsf{pk} \leftarrow \mathbf{PKE}.\mathsf{KG}(\mathsf{sk})$ and return pk.
- $\mathcal{F}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$: query $\mathsf{c} \leftarrow \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$ and return c.

- $\mathcal{F}.\mathsf{Dec}(\mathsf{sk},\mathsf{c})$: query $\mathsf{m} \leftarrow \mathbf{PKE}.\mathsf{Dec}(\mathsf{sk},\mathsf{c})$ and return m.

Since all queries issued by $\mathcal{D}$ are answered exactly as in the ideal game, its view in Game 1 is identical to that in Ideal Game. Defining $\mathbf{G}_1$ as Game 1, which outputs 1 if and only if $\mathcal{D}$ outputs 1 after interacting with $\mathcal{F}$ and $\mathcal{S}$, we obtain

$$\begin{aligned}\Pr[\mathbf{G}_1(\mathcal{D})=1] &= \Pr[\mathcal{D}^{\mathcal{F}^{\mathbf{PKE}},\mathcal{S}^{\mathbf{PKE}}}(1^\lambda)=1] \\ &= \Pr[\mathbf{Ideal}_{\mathcal{S}}^{\mathbf{PKE}}(\mathcal{D})=1].\end{aligned}$$

C.2 Transition from $\mathbf{G}_1$ to $\mathbf{G}_2$

Game 2. In this game, the simulator is modified to incorporate some conditional tests before returning partial responses. These conditions are specifically designed to capture correlations between the simulator's responses that could potentially be exploited by the differentiator $\mathcal{D}$. If the modified simulator $\mathcal{S}_{\mathsf{fail}}$ produces a response that triggers any of the failure conditions below, it explicitly fails instead of returning that response.

In response to a $\mathcal{H}$ query with input $(\mathsf{pk},\mathsf{m},\mathsf{r})$, the simulator samples a value $\tilde{\mathsf{r}} \in \mathcal{R}$ uniformly as the candidate response, and then checks the following failure conditions:

- $\mathsf{C}_{1\text{-}\mathcal{H}}(\tilde{\mathsf{r}})$: The value $\tilde{\mathsf{r}}$ already appears in T_{KEM} or $T_{\mathcal{H}}$.
- $\mathsf{C}_{2\text{-}\mathcal{H}}(\mathsf{r})$: The value r already appears in T^*_{r}.

In response to a $\mathbf{KEM}.\mathsf{KG}$ query with input sk, the simulator simply returns $\mathbf{PKE}.\mathsf{KG}(\mathsf{sk})$ and performs no failure checks.

In response to a $\mathbf{KEM}.\mathsf{Encaps}$ query with input $(\mathsf{pk},\tilde{\mathsf{r}})$, the simulator first verifies the validity of pk via $\mathbf{PKE}.\mathsf{Enc}$. If pk is invalid, the simulator returns $\perp_{\mathsf{pk}}$. If pk is valid and $\tilde{\mathsf{r}}$ is generated by $\mathcal{H}$ with corresponding c in $T_{\mathcal{P}}$, then a corresponding c_k is determined. In this case, if the differentiator has not yet decapsulated c_k using sk, the simulator samples k and n uniformly at random as candidate responses, and subsequently checks the following failure conditions:

- $\mathsf{C}_{1\text{-}\mathsf{Encaps}}(\mathsf{k}||\mathsf{n})$: The value $\mathsf{k}||\mathsf{n}$ already appears in T_{KEM} or $T_{\mathcal{AE}}$.

Otherwise, the simulator samples c_k, k, and n uniformly at random as candidate responses, and then checks the failure conditions $\mathsf{C}_{1\text{-}\mathsf{Encaps}}(\mathsf{k}||\mathsf{n})$ and:

- $\mathsf{C}_{2\text{-}\mathsf{Encaps}}(\mathsf{c}_\mathsf{k})$: The value c_k already appears in $T_\perp$ or T_{KEM}.

In response to a $\mathbf{KEM}.\mathsf{Decaps}$ query with input $(\mathsf{sk},\mathsf{c}_\mathsf{k})$, the simulator proceeds as follows. Let $\mathsf{pk} \leftarrow \mathbf{PKE}.\mathsf{KG}(\mathsf{sk})$. If c_k is obtained through $\mathbf{KEM}.\mathsf{Encaps}$ under pk, then the response $\mathsf{k}\|\mathsf{n}$ is already determined. Otherwise, if c_k is regarded as a valid ciphertext with respect to sk, the simulator samples k and n uniformly at random as the candidate response, and subsequently checks the following failure conditions:

- $\mathsf{C}_{1\text{-}\mathsf{Decaps}}(\mathsf{k}||\mathsf{n})$: The value $\mathsf{k}||\mathsf{n}$ already appears in T_{KEM} or $T_{\mathcal{AE}}$.
- $\mathsf{C}_{2\text{-}\mathsf{Decaps}}(\mathsf{sk})$: The value sk already appears in T^*_{sk}.

In response to a $\mathcal{AE}$ query, if the response is not already determined by the relationships among other interfaces, the simulator samples c_m uniformly at random as the candidate response, and then checks the following failure conditions:

- $\mathsf{C}_{1\text{-}\mathcal{AE}}(\mathsf{c}_\mathsf{m})$: The value c_m already exists in $T_{\mathcal{P}}$ or $T_{\mathcal{AE}}$.

In response to a $\mathcal{P}$ query with input $\mathsf{c_k}||\mathsf{c_m}$, the simulator first checks the validity of $\mathsf{c_k}$. If $\mathsf{c_k}$ is regarded as invalid (i.e., $\mathsf{c_k} \in T_\perp$), the simulator samples an invalid ciphertext c as the candidate response and then checks the following failure conditions:

- $\mathsf{C}_{1\text{-}\mathcal{P}}(\mathsf{c})$: The generated c already appears in $T_{\mathcal{P}}$.
- $\mathsf{C}_{2\text{-}\mathcal{P}}(\mathsf{c})$: The generated c is valid, i.e., $\mathbf{PKE}.\mathsf{Dec}(\mathsf{sk_{test}}, \mathsf{c}) \neq \perp_{\mathsf{c}}$.

If $\mathsf{c_k}$ and $\mathsf{c_m}$ are generated through Encaps and $\mathcal{AE}$ respectively, and correspond to the same pk and m, then the response is the ciphertext under $(\mathsf{pk}, \mathsf{m})$. If no corresponding randomness r exists, the simulator generates a fresh ciphertext c as the candidate response by sampling r uniformly at random, and then checks the following failure conditions:

- $\mathsf{C}_{3\text{-}\mathcal{P}}(\mathsf{r})$: The value r already appears in $T_{\mathcal{H}}$ or T^*_{r}.
- $\mathsf{C}_{4\text{-}\mathcal{P}}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$: The exists c in $T_{\mathcal{P}}$ such that $\mathsf{c} = \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$.

Otherwise, the simulator determines the validity of an unknown $\mathsf{c_k}$ by sampling from a Bernoulli distribution. If $\mathsf{c_k}$ is regarded as invalid, the simulator samples an invalid ciphertext c as the candidate response and then checks the failure conditions $\mathsf{C}_{1\text{-}\mathcal{P}}(\mathsf{c})$ and $\mathsf{C}_{2\text{-}\mathcal{P}}(\mathsf{c})$. Otherwise, for valid $\mathsf{c_k}$, the simulator generates a fresh ciphertext c as the candidate response by sampling sk, m, and r uniformly at random, and then checks the following failure conditions:

- $\mathsf{C}_{5\text{-}\mathcal{P}}(\mathsf{sk}, \mathsf{m}, \mathsf{r})$: The exists c in $T_{\mathcal{P}}$ such that $\mathsf{c} = \mathbf{PKE}.\mathsf{Enc}\left(\mathbf{PKE}.\mathsf{KG}(\mathsf{sk}), \mathsf{m}, \mathsf{r}\right)$.
- $\mathsf{C}_{6\text{-}\mathcal{P}}(\mathsf{sk})$: The value sk already appears in T_{KEM}.

In response to a $\mathcal{P}^{-1}$ query with input c, the simulator first checks the validity of c using $\mathbf{PKE}.\mathsf{Dec}$. If c is invalid, the simulator samples $\mathsf{c_k}$ and $\mathsf{c_m}$ uniformly at random as the candidate response, and then checks the following failure conditions:

- $\mathsf{C}_{1\text{-}\mathcal{P}^{-1}}(\mathsf{c_m})$: The value $\mathsf{c_m}$ already appears in $T_{\mathcal{P}}$ or $T_{\mathcal{AE}}$.
- $\mathsf{C}_{2\text{-}\mathcal{P}^{-1}}(\mathsf{c_k})$: The value $\mathsf{c_k}$ already appears in T_{KEM}.

If c is valid and there exists $(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}})$ such that $(\mathsf{pk}, \mathsf{m}, \mathsf{r})$ corresponds to c and $\tilde{\mathsf{r}}$ corresponds to $\mathsf{c_k}$, the simulator samples $\mathsf{c_m}$ uniformly at random and then checks the following failure conditions $\mathsf{C}_{1\text{-}\mathcal{P}^{-1}}(\mathsf{c_m})$ and:

- $\mathsf{C}_{3\text{-}\mathcal{P}^{-1}}(\mathsf{c_k})$: The value $\mathsf{c_k}$ already exists in $T_\perp$.

Otherwise, the simulator samples both $\mathsf{c_k}$ and $\mathsf{c_m}$ uniformly at random as the candidate response, and then checks the failure conditions $\mathsf{C}_{1\text{-}\mathcal{P}^{-1}}(\mathsf{c_m})$ and $\mathsf{C}_{3\text{-}\mathcal{P}^{-1}}(\mathsf{c_k})$.

Intuitively, each of the conditions above corresponds to an unwanted collision between a freshly sampled element and information already recorded in the simulator's tables; each such event can be bounded by a negligible probability. A formal probabilistic analysis is deferred to Appendix C.7.

Lemma C.1. $\Pr\left[\text{Failure Conditions Hold}\right] \leq O\left(q^2/N + q^2/2^\lambda\right)$.

The differentiator can distinguish between Game 1 and Game 2 only when the following two events occur:

- In Game 1, the simulator $\mathcal{S}$ does not fail, even when its response satisfies one of the failure conditions above;
- In Game 2, the simulator $\mathcal{S}_{\mathsf{fail}}$ does fail.

Let $\mathbf{G}_2$ denoted Game 2, which outputs 1 if and only if $\mathcal{D}$ outputs 1 after interacting with $\mathcal{F}$ and $\mathcal{S}_{\mathsf{fail}}$. Formally,

$$\Pr[\mathbf{G}_2(\mathcal{D})] = \Pr[\mathcal{D}^{\mathcal{F}^{\mathbf{PKE}},\mathcal{S}_{\mathsf{fail}}^{\mathbf{PKE}}}(1^\lambda) = 1].$$

Then

$$\begin{aligned}|\Pr[\mathbf{G}_2(\mathcal{D}) = 1] - \Pr[\mathbf{G}_1(\mathcal{D}) = 1]| &\leq 2 \cdot \Pr[\text{Failure Conditions Hold}] \\ &\leq O\left(\frac{q^2}{N} + \frac{q^2}{2^\lambda}\right).\end{aligned}$$

C.3 Properties under the absence of failure conditions

Before describing Game 3, we state several propositions that record basic observations and will be useful in the proofs of the subsequent lemmas.

Proposition C.1. *Suppose that none of the failure conditions holds. Then there do not exist two distinct pairs* $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ *and* $(\mathsf{k}, \mathsf{n}, \mathsf{m}', \mathsf{c_m})$ *in* $T_{\mathcal{AE}}$ *with* $\mathsf{m} \neq \mathsf{m}'$.

Proof. For any pair $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ inserted by the $\mathcal{AE}(\mathsf{k}, \mathsf{n}, \mathsf{m})$ query, rather than Case 1 where the corresponding response is already recorded in $T_{\mathcal{AE}}$, the simulator proceeds according to one of the remaining three cases, which correspond to those already specified in the simulator's description.

Case 2. In this case, k and n are generated via the KEM interfaces, with the corresponding $\tilde{\mathsf{r}}$ recorded; that is, there exists a tuple $(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$ in T_{KEM}. If there further exists $(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}}) \in T_{\mathcal{H}}$ and $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}) \in T_{\mathcal{P}}$ such that $\mathsf{c} = \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$, then the simulator returns $\mathsf{c_m}$.

Case 3. In this case, k and n are generated via the Decaps interface, i.e., there exists a tuple $(\mathsf{sk}, \mathsf{pk}, \triangle, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$ in T_{KEM}. If there further exists $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}) \in T_{\mathcal{P}}$ such that $\mathsf{m} = \mathbf{PKE}.\mathsf{Dec}(\mathsf{sk}, \mathsf{c})$, then the simulator returns $\mathsf{c_m}$.

Case 4. If none of the above cases applies, the simulator returns a uniformly random $\mathsf{c_m}$.

If the pair is inserted by the $\mathcal{AD}(\mathsf{k}, \mathsf{n}, \mathsf{c_m})$ query, then, as described for the simulator, rather than Case 1 where the corresponding response is already recorded in $T_{\mathcal{AE}}$, the situation falls into one of three possibilities. Case 2 and Case 3 for $\mathcal{AD}$ correspond directly to Case 2 and Case 3 of $\mathcal{AE}$, with identical conditions. Finally, in Case 4, if the response is not already determined by the above relationships, the simulator returns $\perp$.

Suppose there exist two distinct pairs $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ and $(\mathsf{k}, \mathsf{n}, \mathsf{m}', \mathsf{c_m})$ in $T_{\mathcal{AE}}$. Then $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ must have been inserted first, either by a $\mathcal{AE}$ or $\mathcal{AD}$ query, and only afterwards $(\mathsf{k}, \mathsf{n}, \mathsf{m}', \mathsf{c_m})$ must have inserted by a $\mathcal{AE}$ query. Any such collision necessarily implies that one of the failure conditions is triggered. Specifically,

- If both $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ and $(\mathsf{k}, \mathsf{n}, \mathsf{m}', \mathsf{c_m})$ are inserted under Cases 2 or 3, and no $\mathsf{k}||\mathsf{n}$ collision exists in T_{KEM} (i.e., conditions $\mathsf{C}_{\text{1-Encaps}}$ and $\mathsf{C}_{\text{1-Decaps}}$ are not triggered), then they correspond to the same tuple $(*, \mathsf{pk}, *, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$ in T_{KEM}. Moreover, the associated tuples $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ and $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}')$ must have been inserted via a $\mathcal{P}^{-1}$ query, with corresponding $(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}})$ and $(\mathsf{pk}||\mathsf{m}'||\mathsf{r}', \tilde{\mathsf{r}})$ in $T_{\mathcal{H}}$; otherwise, $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ or $(\mathsf{k}, \mathsf{n}, \mathsf{m}', \mathsf{c_m})$ would already exist. Consequently, a $\tilde{\mathsf{r}}$ collision occurs in $T_{\mathcal{H}}$, triggering condition $\mathsf{C}_{\text{2-}\mathcal{H}}$.
- If $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ is first inserted by Case 3 of $\mathcal{AE}$, where $\mathsf{c_m}$ is sampled uniformly, and later $(\mathsf{k}, \mathsf{n}, \mathsf{m}', \mathsf{c_m})$ is inserted by Case 2 or 3, then since $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}')$ corresponding to $(\mathsf{k}, \mathsf{n}, \mathsf{m}', \mathsf{c_m})$ is inserted via $\mathcal{P}^{-1}$ with $\mathsf{c_m}$ also uniformly sampled, a collision occurs between $\mathsf{c_m}$ in $T_{\mathcal{AE}}$ and $T_{\mathcal{P}}$, thereby triggering the failure conditions $\mathsf{C}_{\text{1-}\mathcal{AE}}$ or $\mathsf{C}_{\text{2-}\mathcal{P}^{-1}}$.

□
□

Proposition C.2. *Suppose that none of the failure conditions holds. Then there do not exist two distinct pairs* $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ *and* $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}')$ *in* $T_{\mathcal{AE}}$ *with* $\mathsf{c_m} \neq \mathsf{c_m}'$.

Proof. Suppose there exist two distinct pairs $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ and $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}')$ in $T_{\mathcal{AE}}$. Then $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ must have been inserted first, either by a $\mathcal{AE}$ or $\mathcal{AD}$ query, and only afterwards $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}')$ must have inserted by a $\mathcal{AD}$ query. Any such collision necessarily implies that one of the failure conditions is triggered. Specifically,

- If both $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ and $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}')$ are inserted under Cases 2 or 3, and no $\mathsf{k}||\mathsf{n}$ collision exists in T_{KEM} (i.e., conditions $\mathsf{C}_{\text{1-Encaps}}$ and $\mathsf{C}_{\text{1-Decaps}}$ are not triggered), then they correspond to the same tuple $(*, \mathsf{pk}, *, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$ in T_{KEM}. Moreover, the associated tuples $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ and $(\mathsf{c_k}||\mathsf{c_m}', \mathsf{c}')$ must have been inserted via a $\mathcal{P}^{-1}$ query, with corresponding $(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}})$ and $(\mathsf{pk}'||\mathsf{m}'||\mathsf{r}', \tilde{\mathsf{r}})$ in $T_{\mathcal{H}}$; otherwise, $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ or $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}')$ would already exist. Consequently, a $\tilde{\mathsf{r}}$ collision occurs in $T_{\mathcal{H}}$, triggering condition $\mathsf{C}_{\text{2-}\mathcal{H}}$.
- If $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ is first inserted by Case 3 of $\mathcal{AE}$, and then $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}')$ is inserted by Case 2 or 3, then, provided no failure conditions are triggered, the corresponding $(\mathsf{c_k}||\mathsf{c_m}', \mathsf{c})$ already exists in $T_{\mathcal{P}}$ and $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ exists in $T_{\mathcal{AE}}$. The simulator must therefore ensure that $\mathsf{c_m} = \mathsf{c_m}'$.

□
□

Proposition C.3. *Suppose that none of the failure conditions holds. Then there do not exist two distinct pairs* $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ *and* $(\mathsf{c_k}'||\mathsf{c_m}', \mathsf{c})$ *in* $T_{\mathcal{P}}$ *with* $\mathsf{c_k}||\mathsf{c_m} \neq \mathsf{c_k}'||\mathsf{c_m}'$.

Proof. For any pair $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ inserted by the $\mathcal{P}(\mathsf{c_k}||\mathsf{c_m})$ query, rather than Case 1 where the corresponding response is already recorded in $T_{\mathcal{P}}$, the simulator proceeds according to one of the remaining seven cases, which correspond to those already specified in the simulator's description.

Case 2. If $\mathsf{c_k}$ is regarded as invalid, the simulator repeatedly samples ciphertexts c uniformly at random until an invalid one is obtained, making at most λ attempts.

Case 3. If $\mathsf{c_k}$ and $\mathsf{c_m}$ are generated via Encaps and $\mathcal{AE}$ for the same pk and m, that is, there exist $(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \in T_{\mathsf{KEM}}$ and $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}) \in T_{\mathcal{AE}}$, and if the corresponding randomness r also exists, i.e., $(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}}) \in T_{\mathcal{H}}$, then the simulator returns the ciphertext c under $(\mathsf{pk}, \mathsf{m}, \mathsf{r})$.

Case 4. Under the same pk and m as in Case 3, if no corresponding r exists, the simulator samples r uniformly at random, generates c via $\mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$, and records r in T_{r}^*.

Case 5. If $\mathsf{c_k}$ and $\mathsf{c_m}$ are generated via Decaps and $\mathcal{AE}$ for the same pk and m, that is, there exist $(\mathsf{sk}, \mathsf{pk}, \triangle, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \in T_{\mathsf{KEM}}$ and $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}) \in T_{\mathcal{AE}}$. The simulator samples r uniformly at random, generates c via $\mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$, and records r in T_{r}^*.

Case 6. If the validity of $\mathsf{c_k}$ is unknown, the simulator decides its validity by sampling from a Bernoulli distribution. If $\mathsf{c_k}$ is deemed invalid, the simulator repeatedly samples ciphertexts c uniformly at random until an invalid one is obtained, performing at most λ attempts.

Case 7. If $\mathsf{c_k}$ is regarded as valid but none of the above cases applies, the simulator samples sk, m, and r uniformly at random to generate a ciphertext c. It then records r in T_{r}^* and sk in T_{sk}^*.

If the pair $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ inserted by the $\mathcal{P}^{-1}(\mathsf{c})$ query, rather than Case 8 where the corresponding response is already recorded in $T_{\mathcal{P}}$, the simulator proceeds according to one of the remaining four cases, which correspond to those already specified in the simulator's description.

Case 9. If c is invalid, the simulator returns uniformly sampled $\mathsf{c_k}$ and $\mathsf{c_m}$, and records $\mathsf{c_k}$ in $T_{\perp}$.

Case 10. If there exist $(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}}) \in T_{\mathcal{H}}$ such that $\mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r}) = \mathsf{c}$, and if a corresponding $\mathsf{c_k}$ exists for $\tilde{\mathsf{r}}$ with $\mathsf{c_m}$ being the encryption of m under $\mathsf{k}||\mathsf{n}$ associated with $\mathsf{c_k}$ and pk, then the response is fully determined.

Case 11. Given a determined c_k as in Case 10, if no corresponding c_m exists, the simulator samples c_m uniformly at random.

Case 12. If none of the above cases applies, the simulator samples both c_k and c_m uniformly at random and returns them.

Suppose there exist two distinct pairs $(c_k||c_m, c)$ and $(c_k'||c_m', c)$ in $T_{\mathcal{P}}$. Then $(c_k||c_m, c)$ must have been inserted first, either by a $\mathcal{P}$ or $\mathcal{P}^{-1}$ query, and only afterwards $(c_k'||c_m', c)$ must have inserted by a $\mathcal{P}$ query. Any such collision necessarily implies that one of the failure conditions is triggered. Specifically,

- If $(c_k||c_m, c)$ is first inserted by Cases 2, 6, or 9, and then $(c_k'||c_m', c)$ is inserted by Cases 2 or 6, the randomly sampled invalid c must collide with an existing ciphertext in $T_{\mathcal{P}}$ (i.e., condition $C_{1\text{-}\mathcal{P}}$ is triggered).
- If $(c_k||c_m, c)$ is first inserted by Case 2 or 6, and later $(c_k'||c_m', c)$ is inserted by Case 3, 4, 5, or 7, this indicates that in Case 2 or 6 the simulator still produced a valid ciphertext even after λ iterations (i.e., condition $C_{2\text{-}\mathcal{P}}$ is triggered).
- If $(c_k||c_m, c)$ is first inserted by Case 3, 4, 5, 7, 10, 11, or 12, and later $(c_k'||c_m', c)$ is inserted by Case 2 or 6, condition $C_{2\text{-}\mathcal{P}}$ is triggered as well.
- If $(c_k||c_m, c)$ is first inserted by Case 3, or 10, and later $(c_k'||c_m', c)$ is inserted by Case 3, there must exist both (k, n, m, c_m) and (k, n, m, c_m'), contradicting Proposition C.2.
- If $(c_k||c_m, c)$ is first inserted by Case 3, 10, or 11, and later $(c_k'||c_m', c)$ is inserted by Case 4, 5, or 7, there must exist one r that simultaneously appears in both $T_{\mathcal{H}}$ and T_r^* (i.e., conditions $C_{3\text{-}\mathcal{H}}$ or $C_{3\text{-}\mathcal{P}}$ are triggered).
- If $(c_k||c_m, c)$ is first inserted by Case 4, 5, or 7, and later $(c_k'||c_m', c)$ is inserted by Case 3, conditions $C_{3\text{-}\mathcal{H}}$ or $C_{3\text{-}\mathcal{P}}$ are triggered as well.
- If both $(c_k||c_m, c)$ and $(c_k'||c_m', c)$ are inserted by Case 4, 5, or 7, there must exist a r collision in T_r^* (i.e., condition $C_{4\text{-}\mathcal{P}}$ is triggered).
- If $(c_k||c_m, c)$ is first inserted by Case 11 or 12, and later $(c_k'||c_m', c)$ is inserted by Case 3, then—provided no failure conditions occur—the simulator guarantees that $c_k||c_m = c_k'||c_m'$.
- If $(c_k||c_m, c)$ is first inserted by Case 12, and later $(c_k'||c_m', c)$ is inserted by Case 4 or 5, the randomly generated valid ciphertext c must collide with an existing ciphertext in $T_{\mathcal{P}}$ (i.e., conditions $C_{5\text{-}\mathcal{P}}$ or $C_{6\text{-}\mathcal{P}}$ are triggered).

□
□

Proposition C.4. *Suppose that none of the failure conditions holds. Then there do not exist two distinct pairs* $(c_k||c_m, c)$ *and* $(c_k||c_m, c')$ *in* $T_{\mathcal{P}}$ *with* $c \neq c'$.

Proof. Suppose there exist two distinct pairs $(c_k||c_m, c)$ and $(c_k||c_m, c')$ in $T_{\mathcal{P}}$. Then $(c_k||c_m, c)$ must have been inserted first, either by a $\mathcal{P}$ or $\mathcal{P}^{-1}$ query, and only afterwards $(c_k'||c_m', c)$ must have inserted by a $\mathcal{P}^{-1}$ query. Any such collision necessarily implies that one of the failure conditions is triggered. Specifically,

- If $(c_k||c_m, c)$ is first inserted either $\mathcal{P}$ or $\mathcal{P}^{-1}$, and then $(c_k||c_m, c')$ is inserted by Case 9, 11 or 12, the randomly sampled $c_k||c_m$ must collide with an existing ciphertext in $T_{\mathcal{P}}$ (i.e., condition $C_{1\text{-}\mathcal{P}^{-1}}$ is triggered).
- If $(c_k||c_m, c)$ is first inserted by Case 2, 6, or 9, and then $(c_k||c_m, c')$ is inserted by Case 10, there must exist one c_k that simultaneously appears in both T_{KEM} and $T_{\perp}$ (i.e., conditions $C_{2\text{-}\mathsf{Encaps}}$, $C_{2\text{-}\mathcal{P}^{-1}}$, or $C_{4\text{-}\mathcal{P}^{-1}}$ are triggered).

- If $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ is first inserted by Case 3, 4, or 5, and then $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}')$ is inserted by Case 10, then—provided there is no $\tilde{\mathsf{r}}$ collision in $T_{\mathcal{H}}$ and no $\mathsf{k}||\mathsf{n}$ collision in T_{KEM}—there must exist $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ and $(\mathsf{k}', \mathsf{n}', \mathsf{m}', \mathsf{c_m})$ with $\mathsf{c_m}$ sampled uniformly at random, thereby triggering condition $\mathsf{C}_{2\text{-}\mathcal{AE}}$.
- If $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ is first inserted by Case 3, 4, 5, or 10, and then $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}')$ is inserted by Case 10, then—provided there is no $\tilde{\mathsf{r}}$ collision in $T_{\mathcal{H}}$ and no $\mathsf{k}||\mathsf{n}$ collision in T_{KEM}—there must exist $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ and $(\mathsf{k}', \mathsf{n}', \mathsf{m}', \mathsf{c_m})$ with $\mathsf{c_m}$ sampled uniformly at random, thereby triggering condition $\mathsf{C}_{2\text{-}\mathcal{AE}}$.
- If $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ is first inserted by Case 7, 11, or 12, and then $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}')$ is inserted by Case 10, then—provided there is no $\tilde{\mathsf{r}}$ collision in $T_{\mathcal{H}}$ and no $\mathsf{k}||\mathsf{n}$ collision in T_{KEM}—the uniformly sampled $\mathsf{c_m}$ by either $\mathcal{P}^{-1}$ or $\mathcal{AE}$ must collide with the recorded cipher (i.e., conditions $\mathsf{C}_{1\text{-}\mathcal{AE}}$, $\mathsf{C}_{2\text{-}\mathcal{AE}}$, or $\mathsf{C}_{1\text{-}\mathcal{P}^{-1}}$ are triggered).

□
□

To facilitate the subsequent analysis, we use the symbol "$\prec$" to indicate the order in which elements are recorded in different tables by the simulator. For instance, when Item_1 is recorded in T_1 prior to Item_2 being recorded in T_2, we write

$$\mathsf{Item}_1 \in \mathsf{T}_1 \prec \mathsf{Item}_2 \in \mathsf{T}_2.$$

If the context is sufficiently clear, this can be abbreviated as $\mathsf{Item}_1 \prec \mathsf{Item}_2$.

Proposition C.5. *Suppose that none of the failure conditions holds. Then for any tuple* $(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}})$ *in* $T_{\mathcal{H}}$, $(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$ *in* T_{KEM}, $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ *in* $T_{\mathcal{AE}}$, *and* $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ *in* $T_{\mathcal{P}}$, *it holds that*

$$\mathsf{c} = \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r}).$$

Proof. When inserting $(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}})$ into $T_{\mathcal{H}}$, where $\tilde{\mathsf{r}}$ is sampled uniformly at random, provided that $\tilde{\mathsf{r}}$ does not already apppears in T_{KEM} (i.e., condition $\mathsf{C}_{1\text{-}\mathcal{H}}(\tilde{\mathsf{r}})$ does not hold), we have

$$(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}}) \in T_{\mathcal{H}} \prec (*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \in T_{\mathsf{KEM}}$$

If $(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$ is inserted through the encapsulation interface in Case 3 or 4, then there already exists $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ in $T_{\mathcal{P}}$ such that $\mathsf{c} = \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$. If $(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$ is inserted through the encapsulation interface in Case 5, we obtain

$$(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}}) \prec (*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \prec (\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}), (\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}).$$

If $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ is inserted by Case 2 or 3 of $\mathcal{AE}$ or $\mathcal{AD}$, then the pair $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ already exists with $\mathsf{c} = \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$. Otherwise, we have $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}) \prec (\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$, and, as argued before, $\mathsf{c} = \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$, provided that none of the failure conditions are triggered.

Thus, by considering all cases above, we conclude that, as long as none of the failure conditions occur, the simulator assigns $\mathsf{c} = \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$ as stated in the proposition. □ □

Proposition C.6. *Suppose that none of the failure conditions holds. Then for any tuple* $(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}})$ *in* $T_{\mathcal{H}}$, $(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, *)$ *in* T_{KEM}, *and* $(\mathsf{c_k}'||\mathsf{c_m}, \mathsf{c})$ *in* $T_{\mathcal{P}}$ *such that* $\mathsf{c} = \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$, *it holds that* $\mathsf{c_k} = \mathsf{c_k}'$.

Proof. Provided that the uniformly sampled $\tilde{\mathsf{r}}$ from $\mathcal{H}$ does not already appear in T_{KEM}, we have

$$(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}}) \prec (*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, *).$$

If $(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, *)$ is inserted through the encapsulation interface in Case 3 or 4, then $(\mathsf{c_k}'||\mathsf{c_m}, \mathsf{c})$ must already exist with $\mathsf{c_k} = \mathsf{c_k}'$.

If $(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, *)$ is inserted through the encapsulation interface in Case 5, then

$$(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}}) \prec (*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, *) \prec (\mathsf{c_k}'||\mathsf{c_m}, \mathsf{c}).$$

Provided that no r appears simultaneously in both $T_{\mathcal{H}}$ and T^*_{r}, the pair $(\mathsf{c_k}'||\mathsf{c_m}, \mathsf{c})$ may be inserted by $\mathcal{P}$ in Case 3, or by $\mathcal{P}^{-1}$. In all such cases, the simulator ensures that $\mathsf{c_k} = \mathsf{c_k}'$.

□ □

Proposition C.7. *Suppose that none of the failure conditions holds. Then for any tuple* $(\mathsf{pk}||\mathsf{m}||\mathsf{r},\tilde{\mathsf{r}})$ *in* $T_{\mathcal{H}}$, $(*,\mathsf{pk},\tilde{\mathsf{r}},\mathsf{c_k},\mathsf{k}||\mathsf{n})$ *in* T_{KEM}, $(\mathsf{k},\mathsf{n},\mathsf{m},\mathsf{c_m})$ *in* $T_{\mathcal{AE}}$, *and* $(\mathsf{c_k}||\mathsf{c_m}',\mathsf{c})$ *in* $T_{\mathcal{P}}$ *such that* $\mathsf{c} = \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk},\mathsf{m},\mathsf{r})$, *it holds that* $\mathsf{c_m} = \mathsf{c_m}'$.

Proof. Provided that the uniformly sampled $\tilde{\mathsf{r}}$ from $\mathcal{H}$ does not already appear in T_{KEM}, we have

$$(\mathsf{pk}||\mathsf{m}||\mathsf{r},\tilde{\mathsf{r}}) \prec (*,\mathsf{pk},\tilde{\mathsf{r}},\mathsf{c_k},\mathsf{k}||\mathsf{n}).$$

If $(*,\mathsf{pk},\tilde{\mathsf{r}},\mathsf{c_k},\mathsf{k}||\mathsf{n})$ is inserted through the encapsulation interface in Case 3, then provided that no failure conditions occur, we have

$$(\mathsf{c_k}||\mathsf{c_m}',\mathsf{c}) \prec (\mathsf{sk},\mathsf{pk},\triangle,\mathsf{c_k},\mathsf{k}||\mathsf{n}) \prec (\mathsf{k},\mathsf{n},\mathsf{m},\mathsf{c_m}).$$

If the pair $(\mathsf{k},\mathsf{n},\mathsf{m},\mathsf{c_m})$ is inserted by $\mathcal{AE}$ in Case 2 or 3, the simulator ensures that $\mathsf{c_m} = \mathsf{c_m}'$. If $(\mathsf{k},\mathsf{n},\mathsf{m},\mathsf{c_m})$ is inserted by $\mathcal{AD}$ with $\mathsf{c_m} =\neq \mathsf{c_m}'$, this would amount to the existence of an $\mathsf{c_m}$-collision in $T_{\mathcal{AE}}$, which is ruled out by Proposition C.2.

If $(*,\mathsf{pk},\tilde{\mathsf{r}},\mathsf{c_k},\mathsf{k}||\mathsf{n})$ is inserted through the encapsulation interface in Case 4, then, provided that no failure conditions occur, we have

$$(\mathsf{pk}||\mathsf{m}||\mathsf{r},\tilde{\mathsf{r}}),(\mathsf{c_k}||\mathsf{c_m}',\mathsf{c}) \prec (*,\mathsf{pk},\tilde{\mathsf{r}},\mathsf{c_k},\mathsf{k}||\mathsf{n}) \prec (\mathsf{k},\mathsf{n},\mathsf{m},\mathsf{c_m}).$$

Here, $(\mathsf{k},\mathsf{n},\mathsf{m},\mathsf{c_m})$ must be inserted by $\mathcal{AE}$ or $\mathcal{AD}$ in Case 2, with $\mathsf{c_m} = \mathsf{c_m}'$.

If instead $(*,\mathsf{pk},\tilde{\mathsf{r}},\mathsf{c_k},*)$ is inserted through the encapsulation interface in Case 5, then, provided that no failure conditions occur, we have

$$(\mathsf{pk}||\mathsf{m}||\mathsf{r},\tilde{\mathsf{r}}) \prec (\triangle,\mathsf{pk},\tilde{\mathsf{r}},\mathsf{c_k},\mathsf{k}||\mathsf{n}) \prec (\mathsf{k},\mathsf{n},\mathsf{m},\mathsf{c_m}),$$

and

$$(\mathsf{pk}||\mathsf{m}||\mathsf{r},\tilde{\mathsf{r}}) \prec (\triangle,\mathsf{pk},\tilde{\mathsf{r}},\mathsf{c_k},\mathsf{k}||\mathsf{n}) \prec (\mathsf{c_k}||\mathsf{c_m}',\mathsf{c}).$$

In this case, the pair $(\mathsf{k},\mathsf{n},\mathsf{m},\mathsf{c_m})$ must be inserted by $\mathcal{AE}$ in Case 2, 3, or 4, and the simulator again enforces $\mathsf{c_m} = \mathsf{c_m}'$.

Thus, by considering all possibilities, we conclude that as long as no failure conditions occur, the simulator always ensures $\mathsf{c_m} = \mathsf{c_m}'$, as stated in the proposition. □

□

Proposition C.8. *Suppose that none of the failure conditions holds. Then for any tuple* $(\mathsf{sk},\mathsf{pk},*,\mathsf{c_k},\mathsf{k}||\mathsf{n})$ *in* T_{KEM}, $(\mathsf{k},\mathsf{n},\mathsf{m},\mathsf{c_m})$ *in* $T_{\mathcal{AE}}$, *and* $(\mathsf{c_k}||\mathsf{c_m}',\mathsf{c})$ *in* $T_{\mathcal{P}}$ *such that* $\mathsf{m} = \mathbf{PKE}.\mathsf{Dec}(\mathsf{sk},\mathsf{c})$, *it holds that* $\mathsf{c_m} = \mathsf{c_m}'$.

Proof. Since $\mathbf{PKE}.\mathsf{Enc}$ is injective from the public key, message, and randomness to the ciphertext, each c can be decrypted by exactly one sk. Therefore, if $(\mathsf{c_k}||\mathsf{c_m}',\mathsf{c})$ is inserted by $\mathcal{P}$ in Case 3, 4, or 5, or by $\mathcal{P}^{-1}$ in Case 10, it must already hold that $(\mathsf{k},\mathsf{n},\mathsf{m},\mathsf{c_m}')$ exists. By Proposition C.2, it follows that $\mathsf{c_m} = \mathsf{c_m}'$.

If $(\mathsf{c_k}||\mathsf{c_m}',\mathsf{c})$ is inserted by $\mathcal{P}$ in Case 7, the ciphertext c is generated by sampling sk, m, and r uniformly at random, with sk recorded in T^*_{sk}. In this situation, sk appears simultaneously in both T_{KEM} and T^*_{sk}, thereby triggering condition $\mathsf{C}_{\text{2-Decaps}}$ or $\mathsf{C}_{7\text{-}\mathcal{P}}$.

If $(\mathsf{c_k}||\mathsf{c_m}',\mathsf{c})$ is inserted by $\mathcal{P}^{-1}$ in Case 11, then $(\mathsf{pk}||\mathsf{m}||\mathsf{r},\tilde{\mathsf{r}})$ must already exist in $T_{\mathcal{H}}$ with $\mathsf{c} = \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk},\mathsf{m},\mathsf{r})$, and $\mathsf{c_k}$ in T_{KEM} corresponding to $\tilde{\mathsf{r}}$. Provided that no failure conditions occur, we have

$$(\mathsf{pk}||\mathsf{m}||\mathsf{r},\tilde{\mathsf{r}}) \prec (*,\mathsf{pk},\tilde{\mathsf{r}},\mathsf{c_k},\mathsf{k}||\mathsf{n}) \prec (\mathsf{c_k}||\mathsf{c_m}',\mathsf{c}).$$

The tuple $(*,\mathsf{pk},\tilde{\mathsf{r}},\mathsf{c_k},\mathsf{k}||\mathsf{n})$ must be inserted through the encapsulation interface in Case 5. By Proposition C.2, the pair $(\mathsf{k},\mathsf{n},\mathsf{m},\mathsf{c_m})$ must then be inserted by $\mathcal{AE}$ or $\mathcal{AD}$ in Case 2 or 3, and the simulator enforces $\mathsf{c_m} = \mathsf{c_m}'$.

Thus, by considering all cases above, we conclude that, as long as none of the failure conditions occur, the simulator indeed assigns $\mathsf{c_m} = \mathsf{c_m}'$ as stated in the proposition. □

□

Proposition C.9. *Suppose that none of the failure conditions holds. Then for any tuple* $(\mathsf{sk}, \mathsf{pk}, *, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$ *in* T_{KEM}*,* $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ *in* $T_{\mathcal{AE}}$*, and* $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ *in* $T_{\mathcal{P}}$*, it holds that*

$$\mathsf{m} = \mathbf{PKE}.\mathsf{Dec}(\mathsf{sk}, \mathsf{c}).$$

Proof. Provided that no failure conditions occur, if $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ is inserted by $\mathcal{AE}$ or $\mathcal{AD}$ in Case 2 or 3, then clearly $\mathsf{m} = \mathbf{PKE}.\mathsf{Dec}(\mathsf{sk}, \mathsf{c})$.

If $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ is inserted by $\mathcal{AE}$ in Case 4, where $\mathsf{c_m}$ is sampled uniformly at random, we have

$$(*, \mathsf{pk}, *, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \prec (\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}) \prec (\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}).$$

If $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ is inserted by $\mathcal{P}$, the simulator sets c as the encryption of m under pk, implying $\mathsf{m} = \mathbf{PKE}.\mathsf{Dec}(\mathsf{sk}, \mathsf{c})$. If $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ is inserted by $\mathcal{P}^{-1}$ with $\mathsf{m} \neq \mathbf{PKE}.\mathsf{Dec}(\mathsf{sk}, \mathsf{c})$, this would amount to the existence of an c-collision in $T_{\mathcal{P}}$, which is ruled out by Proposition C.4.

If $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ is inserted by $\mathcal{AD}$ in Case 4 with $\mathsf{m} \neq \mathbf{PKE}.\mathsf{Dec}(\mathsf{sk}, \mathsf{c})$, then $\mathsf{m} = \perp$ even though c is a valid ciphertext under pk. In this case, $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ must have been inserted by $\mathcal{P}$ in Case 8 or by $\mathcal{P}^{-1}$ in Case 12 or 13, leading to either a collision of sk between T_{KEM} and T^*_{sk}, or a collision on m.

Thus, by considering all cases above, we conclude that, as long as none of the failure conditions occur, the simulator enforces

$$\mathsf{m} = \mathbf{PKE}.\mathsf{Dec}(\mathsf{sk}, \mathsf{c})$$

as stated in the proposition. □

□

Proposition C.10. *Suppose that none of the failure conditions holds. Then for any tuple* $(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$ *in* T_{KEM}*,* $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ *in* $T_{\mathcal{AE}}$*, and* $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ *in* $T_{\mathcal{P}}$*, it holds that*

$$\mathsf{m} = \mathbf{PKE}.\mathsf{Dec}(\mathsf{sk}, \mathsf{c}),$$

where $\mathsf{pk} = \mathbf{PKE}.\mathsf{KG}(\mathsf{sk})$.

The proof is identical to that of Proposition C.9.

C.4 Transition from $\mathbf{G}_2$ to $\mathbf{G}_3$

Game 3. In this game, the simulator $\mathcal{S}_{\mathsf{fail}}$ remains unchanged. Instead, the relay algorithm $\mathcal{F} = (\mathcal{F}.\mathsf{KG}, \mathcal{F}.\mathsf{Enc}, \mathcal{F}.\mathsf{Dec})$ is replaced with the construction $\Pi_{\mathsf{PKE}} = (\Pi_{\mathsf{PKE}}.\mathsf{KG}, \Pi_{\mathsf{PKE}}.\mathsf{Enc}, \Pi_{\mathsf{PKE}}.\mathsf{Dec})$. Notably, Π_{PKE} does not query the ideal PKE directly; rather, it interacts with $\mathcal{S}_{\mathsf{fail}}$, thereby ensuring that the honest interfaces are consistent with the adversarial interfaces. Consequently, the differentiator can distinguish between Game 2 and Game 3 only if it detects an inconsistency between $\mathcal{F}$ and $\mathcal{S}_{\mathsf{fail}}$ in Game 2.

Let $\mathbf{G}_3$ denote Game 3, which outputs 1 if and only if $\mathcal{D}$ outputs 1 after interacting with Π_{NIKE} and $\mathcal{S}_{\mathsf{fail}}$. Formally,

$$\Pr[\mathbf{G}_3(\mathcal{D}) = 1] = \Pr[\mathcal{D}^{\Pi_{\mathsf{PKE}}^{\mathcal{S}_{\mathsf{fail}}}, \mathcal{S}_{\mathsf{fail}}^{\mathbf{PKE}}}(1^\lambda) = 1].$$

Lemma C.2. *Let* q *denote the number of queries made by the differentiator to both the honest and adversarial interfaces. Then,*

$$|\Pr[\mathbf{G}_3(\mathcal{D}) = 1] - \Pr[\mathbf{G}_2(\mathcal{D}) = 1]| \leq O\left(\frac{(\lambda \cdot q)^2}{N} + \frac{(\lambda \cdot q)^2}{2^\lambda}\right).$$

Proof. To prove that the views in Game 2 and Game 3 are statistically close, we argue that the differentiator can only exploit an inconsistency if the simulator $\mathcal{S}_{\mathsf{fail}}$ fails in at least one of the two games.

For Game 3, recall that the honest interfaces produce their responses by querying the adversarial interfaces, namely the simulator $\mathcal{S}_{\mathsf{fail}}$. Hence, as long as the simulator provides consistent answers to identical queries across all invocations, the responses of the honest interface necessarily coincide with those of the

adversarial interface. For random oracle and KEM interfaces, this consistency holds trivially. Proposition C.1 and C.2 guarantee that a AE/AD query cannot produce more than one distinct response, which Proposition C.3 and C.4 ensure the consistency of permutation.

For Game 2, any inconsistency between the honest interface $\mathcal{F}$ and the adversarial interface $\mathcal{S}_{\mathsf{fail}}$ can occur only when the response generated by $\mathcal{F}$ deviates from that of the simulator implementing the Π_{PKE} construction. We now establish that, provided $\mathcal{S}_{\mathsf{fail}}$ does not fail, such an inconsistency cannot arise.

Inconsistency with respect to $\mathcal{F}$.KG. For any public-key generation query on input sk, both the relay algorithm $\mathcal{F}$ and the simulator $\mathcal{S}_{\mathsf{fail}}.\mathbf{KEM}.\mathsf{KG}$ return $\mathsf{pk} \leftarrow \mathbf{PKE}.\mathsf{KG}(\mathsf{sk})$. Therefore, consistency holds directly.

Inconsistency with respect to $\mathcal{F}$.Enc. For any encryption query on input pk, m and r, the relay algorithm $\mathcal{F}$ always returns $\mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$. Consistency therefore requires that the following two hold:

1. If pk is invalid with respect to the ideal PKE, then $\mathcal{S}_{\mathsf{fail}}.\mathbf{KEM}.\mathsf{Encaps}$ on input pk must return $\perp_{\mathsf{pk}}$.
2. Otherwise, let $\mathsf{c} \leftarrow \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$, the simulator $\mathcal{S}_{\mathsf{fail}}$ must return the same ciphertext c.

Based on the description of the simulator, when processing an encapsulation query with input pk, the simulator determines its validity by querying $\mathbf{PKE}.\mathsf{Enc}$. Therefore, for any public key pk that is invalid with respect to the ideal PKE, we must have

$$\mathcal{S}_{\mathsf{fail}}.\mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk}, *) = \perp_{\mathsf{pk}}.$$

Now consider any private key $\mathsf{sk} \in \mathbb{Z}_N$, with $\mathsf{pk} \leftarrow \mathbf{PKE}.\mathsf{KG}(\mathsf{sk})$, and any message m and randomness r. When the differentiator obtains the ciphertext c corresponding to $(\mathsf{pk}, \mathsf{m}, \mathsf{r})$ through $\mathcal{S}_{\mathsf{fail}}$, it must query $\mathcal{P}$ on input $\mathsf{c_k}||\mathsf{c_m}$, where $\mathsf{c_m}$ is the authenticated-encryption ciphertext of m under the key $\mathsf{k}||\mathsf{n}$ and $\mathsf{c_k}$ associated with $(\mathsf{pk}, \mathsf{m}, \mathsf{r})$. In particular, the relationship between $(\mathsf{pk}, \mathsf{m}, \mathsf{r})$ and $\mathsf{c_k}||\mathsf{c_m}$ must fall into the following case:

- There exist tuple $(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}})$ in $T_{\mathcal{H}}$, tuple $(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$ in T_{KEM}, and tuple $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ in $T_{\mathcal{AE}}$.

In this case, Proposition C.5 implies that

$$\mathcal{P}(\mathsf{c_k}||\mathsf{c_m}) = \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r}).$$

Furthermore, the differentiator may also issue a query $\mathsf{c_k}||\mathsf{c_m} \leftarrow \mathcal{P}^{-1}(\mathsf{c})$ with $\mathsf{c} = \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$. In this case, the simulator $\mathcal{S}_{\mathsf{fail}}$ must likewise return $\mathsf{c_k}$ and $\mathsf{c_m}$ whenever the relevant associations hold.

For the consistency of $\mathsf{c_k}$, the simulator queries $\mathcal{H}$ on input $(\mathsf{pk}, \mathsf{m}, \mathsf{r})$ to obtain $\tilde{\mathsf{r}}$, and $\mathsf{c_k}$ must correspond to $\tilde{\mathsf{r}}$ in T_{KEM}. Thus, there exist $(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}}) \in T_{\mathcal{H}}$, $(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, *) \in T_{\mathsf{KEM}}$, and $(\mathsf{c_k}'||\mathsf{c_m}, \mathsf{c}) \in T_{\mathcal{P}}$, and Proposition C.6 ensures that $\mathsf{c_k} = \mathsf{c_k}'$.

For the consistency of $\mathsf{c_m}$, the simulator must query $\mathcal{AE}$ on $(\mathsf{k}, \mathsf{n}, \mathsf{m})$ to obtain $\mathsf{c_m}'$, where $\mathsf{k}||\mathsf{n}$ either directly corresponds to $(\mathsf{pk}, \mathsf{m}, \mathsf{r})$ or arises from the decapsulation of $\mathsf{c_k}$ under sk. Specifically, the relationship between $(\mathsf{pk}, \mathsf{m}, \mathsf{r})$ and $\mathsf{k}||\mathsf{n}$ must fall into one of the following cases:

- Form 1: There exists tuple $(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}})$ in $T_{\mathcal{H}}$ and tuple $(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$ in T_{KEM}.
- Form 2: There exists tuple $(\mathsf{sk}, \mathsf{pk}, *, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$ in T_{KEM}.

Proposition C.7 and Proposition C.8 respectively guarantee that in both forms, $\mathsf{c_m} = \mathsf{c_m}'$.

Inconsistency with respect to $\mathcal{F}$.Dec. For any decryption query on input $(\mathsf{sk}, \mathsf{c})$, the relay algorithm always returns $\mathbf{PKE}.\mathsf{Dec}(\mathsf{sk}, \mathsf{c})$. Consistency therefore requires that the following conditions hold:

1. If c is invalid with respect to the ideal PKE, and there exists $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ in $T_{\mathcal{P}}$, then $\mathcal{S}_{\mathsf{fail}}.\mathbf{KEM}.\mathsf{Decaps}$ on input $\mathsf{c_k}$ must return $\perp_{\mathsf{c}}$.
2. Otherwise, let $\mathsf{m} \leftarrow \mathbf{PKE}.\mathsf{Dec}(\mathsf{sk}, \mathsf{c})$, the simulator $\mathcal{S}_{\mathsf{fail}}$ must return the same ciphertext c.

By the construction of the simulator, whenever $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ is inserted into $T_{\mathcal{P}}$ with c invalid, the corresponding $\mathsf{c_k}$ must be recorded in $T_{\perp}$. If $\mathcal{S}_{\mathsf{fail}}$ does not return $\perp_{\mathsf{c}}$, then $\mathsf{c_k}$ simultaneously appears in both T_{KEM} and $T_{\perp}$, thereby triggering conditions $\mathsf{C}_{\text{2-Encaps}}$, $\mathsf{C}_{\text{2-}\mathcal{P}^{-1}}$, or $\mathsf{C}_{\text{4-}\mathcal{P}^{-1}}$. Thus we have

$$\mathcal{S}_{\mathsf{fail}}.\mathbf{KEM}.\mathsf{Decaps}(*, \mathsf{c_k}) = \perp_{\mathsf{c}}.$$

Now consider any private key sk and ciphertext c with $\mathsf{m} \leftarrow \mathbf{PKE}.\mathsf{Dec}(\mathsf{sk}, \mathsf{c})$, where $\mathsf{c} = \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$ and $\mathsf{pk} = \mathbf{PKE}.\mathsf{KG}(\mathsf{sk})$. When the differentiator decrypts c under sk through $\mathcal{S}_{\mathsf{fail}}$, it must first query $\mathsf{c_k}||\mathsf{c_m} \leftarrow \mathcal{P}^{-1}(\mathsf{c})$ and then obtain $\mathsf{m}' \leftarrow \mathcal{AD}(\mathsf{k}, \mathsf{n}, \mathsf{c_k})$, where $\mathsf{k}||\mathsf{n}$ corresponds to c. The relationship between $(\mathsf{sk}, \mathsf{c})$ and $\mathsf{k}||\mathsf{n}$ must fall into one of the following forms:

- Form 1: There exists tuple $(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$ in T_{KEM}.
- Form 2: There exists tuple $(\mathsf{sk}, \mathsf{pk}, *, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$ in T_{KEM}.

Proposition C.9 and Proposition C.10 together guarantee that in both forms, $\mathsf{m} = \mathbf{PKE}.\mathsf{Dec}(\mathsf{sk}, \mathsf{c})$.

Hence, as long as the simulator $\mathcal{S}_{\mathsf{fail}}$ does not fail in either Game 2 or Game 3, the differentiator's view is statistically indistinguishable between the two games. Let q_{hon} and q_{adv} denote the number of queries made to the honest and adversarial interfaces, with $q_{\text{hon}} + q_{\text{adv}} \leq q$. Furthermore, each query to the honest interface in Game 3 requires at most λ queries to the simulator. Then,

$$\begin{aligned}|\Pr[\mathbf{G}_3(\mathcal{D}) = 1] - \Pr[\mathbf{G}_2(\mathcal{D}) = 1]| &\leq 2 \cdot \Pr[\mathcal{S}_{\mathsf{fail}} \text{ fails}] \\ &\leq O\left(\frac{(\lambda \cdot q)^2}{N} + \frac{(\lambda \cdot q)^2}{2^\lambda}\right).\end{aligned}$$

□
□

C.5 Transition from $\mathbf{G}_3$ to $\mathbf{G}_4$

Game 4. In this game, we modify the simulator such that its responses no longer depend on the ideal PKE. Specifically, the new simulator $\mathcal{S}_{\mathsf{samp}}$ implements each interface using lazy sampling, and all failure conditions are removed from $\mathcal{S}_{\mathsf{samp}}$.

We use the notation $\mathsf{Coin} \xleftarrow{\$} \mathrm{Ber}(p)$ to denote that a single bit Coin is sampled from a Bernoulli distribution with success probability p (i.e., $\Pr[\mathsf{Coin} = 1] = p$). We write $\mathsf{Ele\text{-}pk}(T)$ for the set of public keys $\mathsf{pk} \in \mathcal{PK}$ that appear in T, and $\mathsf{Ele\text{-}c_k}(T)$ for the set of ciphertexts $\mathsf{c_k} \in \mathcal{C}_{\mathsf{k}}$ that appear in T.

Simulator $\mathcal{S}_{\mathsf{samp}}$

$\mathcal{S}_{\mathsf{samp}}.\mathcal{H}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$:

if $\exists(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}}) \in T_{\mathcal{H}}$, then **return** $\tilde{\mathsf{r}}$;

$\tilde{\mathsf{r}} \xleftarrow{\$} \mathcal{R}$; $T_{\mathcal{H}} \leftarrow T_{\mathcal{H}} \cup \{(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}})\}$;

return $\tilde{\mathsf{r}}$;

$\mathcal{S}_{\mathsf{samp}}.\mathbf{KEM}.\mathsf{KG}(\mathsf{sk})$:

if $\exists(\mathsf{sk}, \mathsf{pk}) \in T_{\mathsf{KG}}$, then **return** pk;

$\mathsf{pk} \xleftarrow{\$} \mathcal{PK}$; $T_{\mathsf{KG}} \leftarrow T_{\mathsf{KG}} \cup \{(\mathsf{sk}, \mathsf{pk})\}$;

return pk;

$\mathcal{S}_{\mathsf{samp}}.\mathbf{KEM}.\mathsf{Encaps}(\mathsf{pk}, \tilde{\mathsf{r}})$:

if $\exists(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \in T_{\mathsf{KEM}}$, then **return** $(\mathsf{c_k}, \mathsf{k}||\mathsf{c})$;

if $\mathsf{pk} \in T_{\perp_{\mathsf{pk}}}$: **return** $\perp_{\mathsf{pk}}$;

if pk does not appear in T_{KG} and T_{KEM}:

$b \xleftarrow{\$} \mathrm{Ber}\left(\frac{|\mathcal{SK}|-|\mathsf{Ele\text{-}pk}(T_{\mathsf{KG}} \cup T_{\mathsf{KEM}})|}{|\mathcal{PK}|-|\mathsf{Ele\text{-}pk}(T_{\mathsf{KG}} \cup T_{\mathsf{KEM}} \cup T_{\perp_{\mathsf{pk}}})|}\right)$;

if $b = 0$: $T_{\perp_{\mathsf{pk}}} \leftarrow T_{\perp_{\mathsf{pk}}} \cup \{\mathsf{pk}\}$; **return** $\perp_{\mathsf{pk}}$;

$(\mathsf{c_k}, \mathsf{k}||\mathsf{n}) \xleftarrow{\$} \mathcal{C}_{\mathsf{k}} \times \mathcal{K} \times \mathcal{N}$; $T_{\mathsf{KEM}} \leftarrow T_{\mathsf{KEM}} \cup \{(\triangle, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n})\}$;

return $(\mathsf{c_k}, \mathsf{k}||\mathsf{n})$.

$\mathcal{S}_{\mathsf{samp}}.\mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c_k})$:

if $\exists(\mathsf{sk}, *, *, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$, then **return** $\mathsf{k}||\mathsf{n}$;

if $\exists(\mathsf{sk}, \mathsf{pk}) \in T_{\mathsf{KG}}$ with $(\triangle, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \in T_{\mathsf{KEM}}$:

replace $\triangle$ by sk; **return** $\mathsf{k}||\mathsf{n}$;

if $\mathsf{c_k} \in T_{\perp_{\mathsf{c}}}$, then **return** $\perp_{\mathsf{c}}$;

if $\mathsf{c_k}$ does not appear in T_{KEM} and $T_{\mathcal{P}}$:

$b \xleftarrow{\$} \mathrm{Ber}(\frac{|\mathcal{R}|-|\mathsf{Ele\text{-}c_k}(T_{\mathsf{KEM}})|}{|\mathcal{C}_{\mathsf{k}}|-|\mathsf{Ele\text{-}c_k}(T_{\mathsf{KEM}} \cup T_{\perp_{\mathsf{c}}})|})$;

if $b = 0$: $T_{\perp_{\mathsf{c}}} \leftarrow T_{\perp_{\mathsf{c}}} \cup \{\mathsf{c_k}\}$; **return** $\perp_{\mathsf{c}}$;

$T_{\mathsf{KEM}} \leftarrow T_{\mathsf{KEM}} \cup \{(\mathsf{sk}, \triangle, \triangle, \mathsf{c_k}, \perp)\}$;

return $\perp$;

$\mathcal{S}_{\mathsf{samp}}.\mathcal{AE}(\mathsf{k}, \mathsf{n}, \mathsf{m})$:

if $\exists(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}) \in T_{\mathcal{AE}}$, **then return** $\mathsf{c_m}$;

$\mathsf{c_m} \xleftarrow{\$} \mathcal{C}_{\mathsf{m}}$; $T_{\mathcal{AE}} \leftarrow T_{\mathcal{AE}} \cup \{(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})\}$;

return $\mathsf{c_m}$;

$\mathcal{S}_{\mathsf{samp}}.\mathcal{AD}(\mathsf{k}, \mathsf{n}, \mathsf{c_m})$:

if $\exists(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}) \in T_{\mathcal{AE}}$, then **return** m;

$T_{\mathcal{AE}} \leftarrow T_{\mathcal{AE}} \cup \{(\mathsf{k}, \mathsf{n}, \perp, \mathsf{c_m})\}$;

return $\perp$;

$\mathcal{S}_{\mathsf{samp}}.\mathcal{P}(\mathsf{c_k}||\mathsf{c_m})$:

if $\exists(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}) \in T_{\mathcal{P}}$, then **return** c;

$\mathsf{c} \xleftarrow{\$} \mathcal{C}$; $T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})\}$;

return c;

$\underline{\mathcal{S}_{\mathsf{samp}}.\mathcal{P}^{-1}(\mathsf{c})}$:

if $\exists(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}) \in T_{\mathcal{P}}$, then **return** $\mathsf{c_k}||\mathsf{c_m}$;

$(\mathsf{c_k}, \mathsf{c_m}) \xleftarrow{\$} \mathcal{C}_{\mathsf{k}} \times \mathcal{C}_{\mathsf{m}}$; $T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})\}$;

return $\mathsf{c_k}||\mathsf{c_m}$.

We now show that the views of the differentiator in Game 3 and Game 4 are statistically close. Recall that the only difference between the two games lies in how the simulator answers queries. Thus, the proof reduces to analyzing, for each interface, whether the responses generated by $\mathcal{S}_{\mathsf{samp}}$ faithfully reproduce the distribution enforced by $\mathcal{S}_{\mathsf{fail}}$, and whether any statistical deviation can only occur through rare collision events.

Response from the $\mathcal{H}$ interface. For the $\mathcal{H}$ query, the responses generated by $\mathcal{S}_{\mathsf{fail}}$ and $\mathcal{S}_{\mathsf{samp}}$ are all sampled uniformly from $\mathbb{Z}_N$. Hence, their output distributions are identical.

Response from the $\mathbf{KEM}.\mathsf{KG}$ *interface.* For a query to $\mathbf{KEM}.\mathsf{KG}$ with input sk, the simulator $\mathcal{S}_{\mathsf{samp}}$ generates the response pk by sampling uniformly from $\mathcal{PK}$, rather than invoking $\mathbf{PKE}.\mathsf{KG}$. Since the public-key generation interface of the ideal PKE is a random injection from $\mathcal{SK}$ to $\mathcal{PK}$, the only bad event is that the sampled pk collides with an entry in either T_{KG} or $T_{\perp_{\mathsf{pk}}}$ (the set of currently marked invalid public keys). The probability of such a collision is at most $q/|\mathcal{PK}|$.

Response from the $\mathbf{KEM}.\mathsf{Encaps}$ *interface.* For a query to Encaps of $\mathbf{KEM}$ with input $(\mathsf{pk}, \tilde{\mathsf{r}})$, the simulator $\mathcal{S}_{\mathsf{samp}}$ decides whether an unknown public key is valid by sampling from a Bernoulli distribution with probability

$$\frac{|\mathcal{SK}| - |\mathsf{Ele\text{-}pk}(T_{\mathsf{KG}} \cup T_{\mathsf{KEM}})|}{|\mathcal{PK}| - |\mathsf{Ele\text{-}pk}(T_{\mathsf{KG}} \cup T_{\mathsf{KEM}} \cup T_{\perp})|}$$

of being valid, rather than querying $\mathbf{PKE}.\mathsf{Enc}$. These two approaches yield identical distributions. For valid pk, if there exists no $\mathsf{c_k}$ corresponding to $\tilde{\mathsf{r}}$, the simulator $\mathcal{S}_{\mathsf{samp}}$ samples a $\mathsf{c_k}$ uniformly at random. Assuming no failure condition is triggered, the only bad event is that the sampled $\mathsf{c_k}$ collides with the existing entry in T_{KEM} or $T_{\perp_{\mathsf{c}}}$ (the set of currently marked invalid ciphertexts), which occurs with probability at most $q/|\mathcal{C}_{\mathsf{k}}|$.

Response from the $\mathbf{KEM}.\mathsf{Decaps}$ *interface.* For a query to Decaps of $\mathbf{KEM}$ with input $(\mathsf{sk}, \mathsf{c_k})$, both $\mathcal{S}_{\mathsf{fail}}$ and $\mathcal{S}_{\mathsf{samp}}$ decide the validity of $\mathsf{c_k}$ by sampling from a Bernoulli distribution with the same probability. If no failure condition is triggered, then for a valid $\mathsf{c_k}$ with no previously associated $\mathsf{k}||\mathsf{n}$, the simulators sample $\mathsf{k}||\mathsf{n}$ uniformly at random. Hence, their output distributions are identical.

Response from the $\mathcal{AE}$ interface. For a query to $\mathcal{AE}$ with input $(\mathsf{k}, \mathsf{n}, \mathsf{m})$, if there exists a corresponding ciphertext c with $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ recorded in $T_{\mathcal{P}}$ in Game 3, then $\mathcal{S}_{\mathsf{fail}}$ returns c. Provided that no failure condition is triggered, $\mathsf{c_m}$ is sampled uniformly at random in both Game 3 and Game 4. Therefore, the only bad event is that the sampled $\mathsf{c_m}$ collides with an existing ciphertext in $T_{\mathcal{AE}}$, which occurs with probability at most $q/|\mathcal{C}_{\mathsf{m}}|$.

Response from the $\mathcal{AD}$ interface. For a query to $\mathcal{AD}$ with input $(\mathsf{k}, \mathsf{n}, \mathsf{c_m})$, if no corresponding response recorded in $T_{\mathcal{AE}}$, the simulator return $\perp$ directly. Provided that no failure condition is triggered, their output distributions are identical.

Response from the $\mathcal{P}$ interface. For a $\mathcal{P}$ query with input $\mathsf{c_k}||\mathsf{c_m}$, assuming no failure condition is triggered, the only difference is that $\mathcal{S}_{\mathsf{samp}}$ generates the response c by sampling uniformly from $\mathcal{C}$ rather than invoking $\mathbf{PKE}.\mathsf{Enc}$. Since the $\mathbf{PKE}.\mathsf{Enc}$ is itself a random injection from $\mathcal{PK} \times \mathcal{M} \times \mathcal{R}$ to $\mathcal{C}$, the only bad event here is that the sample c collides with a recorded ciphertext in $T_{\mathcal{P}}$, which occurs with probability at most $q/|\mathcal{C}|$.

Response from the $\mathcal{P}^{-1}$ interface. For a $\mathcal{P}^{-1}$ query with input c, assuming no failure condition is triggered, the only bad event is that the sampled $\mathsf{c_k}||\mathsf{c_m}$ collides with an existing entry in $T_{\mathcal{P}}$, which occurs with probability at most $q/|\mathcal{C}|$.

Therefore, if the simulator $\mathcal{S}_{\mathsf{fail}}$ does not fail in Game 3 and no response generated by $\mathcal{S}_{\mathsf{samp}}$ in Game 4 triggers any of the bad events above, the distinguisher's view remains statistically close between the two games. Let $\mathbf{G}_4$ denote Game 4, which outputs 1 if and only if the differentiator outputs 1 after interacting with Π_{PKE} and $\mathcal{S}_{\mathsf{samp}}$. Formally,

$$\Pr[\mathbf{G}_4(\mathcal{D})=1]=\Pr[\mathcal{D}^{\Pi_{\mathsf{PKE}}^{\mathcal{S}_{\mathsf{samp}}},\mathcal{S}_{\mathsf{samp}}}(1^\lambda)=1].$$

Then,

$$|\Pr[\mathbf{G}_4(\mathcal{D})=1]-\Pr[\mathbf{G}_3(\mathcal{D})=1]|\leq O\left(\frac{(\lambda\cdot q)^2}{N}+\frac{(\lambda\cdot q)^2}{2^\lambda}\right).$$

C.6 Transition from $\mathbf{G}_4$ to Real

Real Game. By replacing the simulator $\mathcal{S}_{\mathsf{samp}}$ in Game 4 with the real idealized oracles $\mathcal{O}=(\mathcal{H},\mathbf{KEM},\mathcal{AE},\mathcal{AD},\mathcal{P},\mathcal{P}^{-1})$, we arrive at the final game of our proof, which we refer to as the real game. In this game, the differentiator $\mathcal{D}$ interacts with $\Pi_{\mathsf{PKE}}^{\mathcal{O}}$ and $\mathcal{O}$. The real game outputs 1 if and only if $\mathcal{D}$ outputs 1. Formally,

$$\Pr[\mathbf{Real}^{\Pi_{\mathsf{PKE}}^{\mathcal{O}}}(\mathcal{D})=1]=\Pr[\mathcal{D}^{\Pi_{\mathsf{PKE}}^{\mathcal{O}},\mathcal{O}}(1^\lambda)=1].$$

We now demonstrate that the views of the differentiator in Game 4 and the Real Game are statistically close.

For the random oracle query $\mathcal{H}$, the responses in Game 4 and Real Game are uniformly distributed.

For the public-key generation query $\mathbf{KEM}.\mathsf{KG}$, the differentiator may observe a difference if the uniformly sampled pk collides with the existing one in T_{KG} or $T_{\perp_{\mathsf{pk}}}$, each with probability at most $q/|\mathcal{PK}|$. For the encapsulation query $\mathbf{KEM}.\mathsf{Encaps}$, the simulator in Game 4 determines the validity of public key via the Bernoulli distribution and $T_{\perp_{\mathsf{pk}}}$. Specifically, an unknown public key is deemed valid with probability

$$\frac{|\mathcal{SK}|-|\mathsf{Ele\text{-}pk}(T_{\mathsf{KG}}\cup T_{\mathsf{KEM}})|}{|\mathcal{PK}|-|\mathsf{Ele\text{-}pk}(T_{\mathsf{KG}}\cup T_{\mathsf{KEM}}\cup T_{\perp_{\mathsf{pk}}})|}.$$

Since T_{KG}, T_{KEM} and $T_{\perp_{\mathsf{pk}}}$ contain all valid and invalid public keys known to the differentiator, the views in the two games are identical. For valid input with no corresponding response determined, the simulator in Game 4 samples $\mathsf{c_k}$ and $\mathsf{k}||\mathsf{n}$ uniformly, and the differentiator may observe a difference if such $\mathsf{c_k}$ collides with the existing one in T_{KEM} or $T_{\perp_{\mathsf{c}}}$, each with probability at most $q/|\mathcal{C}_{\mathsf{k}}|$. For the decapsulation query $\mathbf{KEM}.\mathsf{Decaps}$, apparently the views in the two games are identical.

For the $\mathcal{AE}$ query, the differentiator may observe a difference if the uniformly sample $\mathsf{c_m}$ by $\mathcal{S}_{\mathsf{samp}}$ collides with the existing one in $T_{\mathcal{AE}}$, each with probability at most $q/|\mathcal{C}_{\mathsf{m}}|$. For the $\mathcal{AD}$ query, note that the simulator in Game 4 always returns $\perp$ if no response recorded. The differentiator may observe a difference if it generate a pair $(\mathsf{k},\mathsf{n},\mathsf{c_m})$ in the ideal game that can be decrypted, the probability of which is at most $|\mathcal{M}|/|\mathcal{C}_{\mathsf{m}}|\leq 1/2^\lambda$.

For the random permutation queries $\mathcal{P}$ and $\mathcal{P}^{-1}$, the differentiator may observe a difference if the uniformly sampled response by $\mathcal{S}_{\mathsf{samp}}$ collides with the existing one in $T_{\mathcal{P}}$, each with probability at most $q/|\mathcal{C}|$.

Therefore, we deduce that

$$\left|\Pr[\mathbf{Real}^{\Pi_{\mathsf{PKE}}^{\mathcal{O}}}(\mathcal{D})=1]-\Pr[\mathbf{G}_4(\mathcal{D})=1]\right|\leq O\left(\frac{q^2}{2^\lambda}\right).$$

Combining all the games above, we conclude that:

$$\left|\Pr[\mathbf{Real}^{\Pi_{\mathsf{PKE}}^{\mathcal{O}}}(\mathcal{D})=1]-\Pr[\mathbf{Ideal}_{\mathcal{S}}^{\mathbf{PKE}}(\mathcal{D})=1]\right|\leq O\left(\frac{(\lambda\cdot q)^2}{N}+\frac{(\lambda\cdot q)^2}{2^\lambda}\right).$$

□

C.7 Bounding the failure conditions

Note that the simulator $\mathcal{S}_{\mathsf{fail}}$ in Games 2 and 3 may fail under various conditions. In this section, we analyze the probability of each such failure occurring.

Lemma C.1. $\Pr[\text{Failure Conditions Hold}] \leq O\left(q^2/N + q^2/2^\lambda\right)$.

Proof. As described in Appendix C.2, the simulator evaluates potential failure conditions whenever it handles a $\mathcal{H}$, $\mathbf{KEM}$.Encaps, $\mathbf{KEM}$.Decaps, $\mathcal{AE}$, $\mathcal{P}$ or $\mathcal{P}^{-1}$ query.

When processing a $\mathcal{H}$ query on input $(\mathsf{pk}, \mathsf{m}, \mathsf{r})$, the simulator samples $\tilde{\mathsf{r}} \in \mathcal{R}$ uniformly at random as the candidate response. Two distinct failure conditions arise:

Condition $\mathsf{C}_{1\text{-}\mathcal{H}}$. The simulator fails if the sampled $\tilde{\mathsf{r}}$ already exists in T_{KEM} or $T_{\mathcal{H}}$. Since the simulator makes at most q queries to $\mathcal{S}_{\mathsf{fail}}$, each table maintained by $\mathcal{S}_{\mathsf{fail}}$ contains at most q entries, meaning there are at most q randomness recorded. Therefore, the probability that a uniformly sampled $\tilde{\mathsf{r}}$ collides with one of these randomness is bounded by

$$\Pr[\mathsf{C}_{1\text{-}\mathcal{H}}] \leq \frac{q}{|\mathcal{R}|} = \frac{q}{2^\lambda}.$$

Condition $\mathsf{C}_{2\text{-}\mathcal{H}}$. The simulator fails if the differentiator queries with some r such that r already appears in T^*_{r}. Since each randomness in T^*_{r} is sampled uniformly at random and remains hidden from the differentiator, the probability of this event is bounded by

$$\Pr[\mathsf{C}_{2\text{-}\mathcal{H}_x}] \leq \frac{q}{|\mathcal{R}|} = \frac{q}{2^\lambda}.$$

When processing a $\mathbf{KEM}$.Encaps query on input $(\mathsf{pk}, \tilde{\mathsf{r}})$, in some cases the simulator samples $\mathsf{k}||\mathsf{n} \in \mathcal{K} \times \mathcal{N}$ and/or $\mathsf{c}_\mathsf{k} \in \mathcal{C}_\mathsf{k}$ uniformly at random as the candidate response. Two distinct failure conditions arise:

Condition $\mathsf{C}_{1\text{-}\mathsf{Encaps}}$. The simulator fails if the sampled $\mathsf{k}||\mathsf{n}$ already appears in T_{KEM} or $T_{\mathcal{AE}}$. As noted previously, each table maintained by $\mathcal{S}_{\mathsf{fail}}$ contains at most q entries. Therefore, the probability is bounded by

$$\Pr[\mathsf{C}_{1\text{-}\mathsf{Encaps}}] \leq \frac{q}{|\mathcal{K} \times \mathcal{N}|} = \frac{q}{2^{2\lambda}}.$$

Condition $\mathsf{C}_{2\text{-}\mathsf{Encaps}}$. The simulator fails if the sampled c_k already appears in T_{KEM} or $T_\perp$. The probability is bounded by

$$\Pr[\mathsf{C}_{2\text{-}\mathsf{Encaps}}] \leq \frac{q}{|\mathcal{C}_\mathsf{k}|} = \frac{q}{2^{\ell+\lambda}}.$$

When processing a $\mathbf{KEM}$.Decaps query on input $(\mathsf{sk}, \mathsf{c}_\mathsf{k})$, in some cases the simulator samples $\mathsf{k}||\mathsf{n} \in \mathcal{K} \times \mathcal{N}$ uniformly at random as the candidate response. Two distinct failure conditions arise:

Condition $\mathsf{C}_{1\text{-}\mathsf{Decaps}}$. The simulator fails if the sampled $\mathsf{k}||\mathsf{n}$ already appears in T_{KEM} or $T_{\mathcal{AE}}$. The probability is bounded by

$$\Pr[\mathsf{C}_{1\text{-}\mathsf{Decaps}}] \leq \frac{q}{|\mathcal{K} \times \mathcal{N}|} = \frac{q}{2^{2\lambda}}.$$

Condition $\mathsf{C}_{2\text{-}\mathsf{Decaps}}$. The simulator fails if the differentiator queries with some sk such that sk already appears in T^*_{sk}. Since each private key in T^*_{sk} is sampled uniformly at random and remains hidden from the differentiator, the probability of this event is bounded by

$$\Pr[\mathsf{C}_{2\text{-}\mathsf{Decaps}}] \leq \frac{q}{|\mathcal{SK}|} = \frac{q}{N}.$$

When processing a $\mathcal{AE}$ query on input $(\mathsf{k}, \mathsf{n}, \mathsf{m})$, in some cases the simulator samples $\mathsf{c}_\mathsf{m} \in \mathcal{C}_\mathsf{m}$ uniformly at random as the candidate response. One failure condition arises:

Condition $\mathsf{C}_{1\text{-}\mathcal{AE}}$. The simulator fails if the sampled $\mathsf{c_m}$ already appears in $T_{\mathcal{P}}$ or $T_{\mathcal{AE}}$. The probability is bounded by

$$\Pr[\mathsf{C}_{1\text{-Decaps}}] \leq \frac{q}{|\mathcal{C}_{\mathsf{m}}|} = \frac{q}{2^{2\lambda}}.$$

When processing a $\mathcal{P}$ query on input $\mathsf{c_k}||\mathsf{c_m}$, the simulator first checks the validity of $\mathsf{c_k}$. If $\mathsf{c_k}$ is regarded as invalid, the simulator samples an invalid ciphertext c as the candidate response. Two failure conditions arises:

Condition $\mathsf{C}_{1\text{-}\mathcal{P}}$. The simulator fails if the sampled c already appears in $T_{\mathcal{P}}$. The probability is bounded by

$$\Pr[\mathsf{C}_{1\text{-}\mathcal{P}}] \leq \frac{q}{|\mathcal{C}|} \leq \frac{q}{2^{\ell+3\lambda}}.$$

Condition $\mathsf{C}_{2\text{-}\mathcal{P}}$. The simulator fails if, even after λ iterations, the uniformly sampled ciphertext c remains valid. The probability is bounded by

$$\Pr\left[\mathsf{C}_{2\text{-}\mathcal{P}}\right] \leq \left(\frac{|\mathcal{R}| \cdot |\mathcal{C}_{\mathsf{m}}|}{|\mathcal{C}|}\right)^{\lambda} \leq \frac{1}{2^{\lambda}}.$$

If $\mathsf{c_k}$ and $\mathsf{c_m}$ are generated through Encaps and $\mathcal{AE}$ respectively, and correspond to the same pk and m, then the response is the ciphertext under $(\mathsf{pk}, \mathsf{m})$. If no corresponding randomness r exists, the simulator generates a fresh ciphertext c as the candidate response by sampling r uniformly at random, and two failure conditions arises:

Condition $\mathsf{C}_{3\text{-}\mathcal{P}}$. The simulator fails if the sampled r already appears in $T_{\mathcal{H}}$ or T^*_{r}. The probability is bounded by

$$\Pr[\mathsf{C}_{3\text{-}\mathcal{P}}] \leq \frac{q}{|\mathcal{R}|} = \frac{q}{2^{\lambda}}.$$

Condition $\mathsf{C}_{4\text{-}\mathcal{P}}$. The simulator fails if the generated c already appears in $T_{\mathcal{P}}$. The probability is bounded by

$$\Pr[\mathsf{C}_{4\text{-}\mathcal{P}}] \leq \frac{q}{|\mathcal{R}|} = \frac{q}{2^{\lambda}}.$$

Otherwise, the simulator determines the validity of an unknown $\mathsf{c_k}$ by sampling from a Bernoulli distribution. If $\mathsf{c_k}$ is regarded as invalid, the simulator samples an invalid ciphertext c as the candidate response and then checks the failure conditions $\mathsf{C}_{1\text{-}\mathcal{P}}(\mathsf{c})$ and $\mathsf{C}_{2\text{-}\mathcal{P}}(\mathsf{c})$. Otherwise, for valid $\mathsf{c_k}$, the simulator generates a fresh ciphertext c as the candidate response by sampling sk, m, and r uniformly at random, and two failure conditions arises:

Condition $\mathsf{C}_{5\text{-}\mathcal{P}}$. The simulator fails if the sampled r already appears in $T_{\mathcal{H}}$ or T^*_{r}. The probability is bounded by

$$\Pr[\mathsf{C}_{5\text{-}\mathcal{P}}] \leq \frac{q}{|\mathcal{R}|} = \frac{q}{2^{\lambda}}.$$

Condition $\mathsf{C}_{6\text{-}\mathcal{P}}$. The simulator fails if the sampled sk already appears in T_{KEM}. The probability is bounded by

$$\Pr[\mathsf{C}_{6\text{-}\mathcal{P}}] \leq \frac{q}{|\mathcal{SK}|} = \frac{q}{2^{\lambda}}.$$

When processing a $\mathcal{P}^{-1}$ query with input c, the simulator first checks the validity of c using $\mathbf{PKE}.\mathsf{Dec}$. If c is invalid, the simulator samples $\mathsf{c_k}$ and $\mathsf{c_m}$ uniformly at random as the candidate response, and two distinct failure conditions arises:

Condition $\mathsf{C}_{1\text{-}\mathcal{P}^{-1}}$. The simulator fails if the sampled $\mathsf{c_m}$ already appears in $T_{\mathcal{P}}$ or $T_{\mathcal{AE}}$. The probability is bounded by

$$\Pr[\mathsf{C}_{1\text{-}\mathcal{P}^{-1}}] \leq \frac{q}{|\mathcal{C}_{\mathsf{m}}|} = \frac{q}{2^{2\lambda}}.$$

Condition $\mathsf{C}_{2\text{-}\mathcal{P}^{-1}}$. The simulator fails if the sampled c_k already appears in T_{KEM}. The probability is bounded by

$$\Pr[\mathsf{C}_{2\text{-}\mathcal{P}^{-1}}] \leq \frac{q}{|\mathcal{C}_\mathsf{k}|} = \frac{q}{2^{\ell+\lambda}}.$$

If c is valid and there exists $(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}})$ such that $(\mathsf{pk}, \mathsf{m}, \mathsf{r})$ corresponds to c and $\tilde{\mathsf{r}}$ corresponds to c_k, the simulator samples c_m uniformly at random and one failure condition arises:

Condition $\mathsf{C}_{3\text{-}\mathcal{P}^{-1}}$. The simulator fails if the sampled c_m already appears in $T_\perp$. The probability is bounded by

$$\Pr[\mathsf{C}_{3\text{-}\mathcal{P}^{-1}}] \leq \frac{q}{|\mathcal{C}_\mathsf{m}|} = \frac{q}{2^{2\lambda}}.$$

Combine together, we have

$$\Pr\left[\text{Failure Conditions Hold}\right] \leq O\left(\frac{q^2}{N} + \frac{q^2}{2^\lambda}\right).$$

□
□

D Proof of Theorem 6.1

Theorem 6.1. $\Pi^{\mathcal{O}}_{\mathsf{IES}}$ is indifferentiable from the ECIES-compatible ideal KEM **IES-KEM**. More precisely, there exists a simulator $\mathcal{S}$ such that for all q-query differentiator $\mathcal{D}$, we have

$$\mathrm{Adv}^{\mathsf{indiff}}_{\Pi^{\mathcal{O}}_{\mathsf{IES}},\mathbf{IES\text{-}KEM},\mathcal{S}}(\mathcal{D}) \leq O\left(\frac{(\lambda\cdot q)^4}{N} + \frac{(\lambda\cdot q)^2}{2^\lambda}\right).$$

Here the simulator makes at most $O(\lambda \cdot q)$ number of queries to its oracles.

Proof. Correctness of our scheme follows immediately. For indifferentiability, note that in the real world the differentiator interacts with three honest interfaces $(\Pi_{\mathsf{IES}}.\mathsf{KG}, \Pi_{\mathsf{IES}}.\mathsf{Encaps}, \Pi_{\mathsf{IES}}.\mathsf{Decaps})$, along with the adversarial interfaces consisting of the random oracles $\mathcal{H}_x$, $\mathcal{H}_y$, and $\mathcal{H}_{\mathsf{key}}$, the random permutation $(\mathcal{P}, \mathcal{P}^{-1})$, and the generic group $\mathcal{G} = (\mathcal{G}^{\mathsf{label}}, \mathcal{G}^{\mathsf{add}})$. To establish the proof, we construct an efficient simulator $\mathcal{S}$ in the ideal world such that:

1. $\mathcal{S}$ has access to the ECIES-compatible ideal KEM **IES-KEM**;
2. $\mathcal{S}$ simulates those seven adversarial interfaces

$$\left(\mathcal{S}.\mathcal{H}_x, \mathcal{S}.\mathcal{H}_y, \mathcal{S}.\mathcal{H}_{\mathsf{key}}, \mathcal{S}.\mathcal{P}, \mathcal{S}.\mathcal{P}^{-1}, \mathcal{S}.\mathcal{G}^{\mathsf{label}}, \mathcal{S}.\mathcal{G}^{\mathsf{add}}\right).$$

To describe the simulator precisely, we first introduce some notation. Let $\mathsf{sk}_{\mathsf{test}} \in \mathcal{SK}$ be a fixed but otherwise arbitrary private key, and let $\mathsf{pk}_{\mathsf{test}} = \mathbf{IES\text{-}KEM}.\mathsf{KG}(\mathsf{sk}_{\mathsf{test}})$. In the generic group model, a labeling query $\mathsf{str} \leftarrow \mathcal{G}^{\mathsf{label}}(x)$ corresponds to a linear equation $(\mathsf{str} - x \cdot \diamondsuit = 0)$, where $\diamondsuit$ represents the generator, i.e., $\diamondsuit = \mathcal{G}^{\mathsf{label}}(1)$. Similarly, an addition query $\mathsf{str} \leftarrow \mathcal{G}^{\mathsf{add}}(\mathsf{str}_1, \mathsf{str}_2)$ corresponds to a linear equation $(\mathsf{str} - \mathsf{str}_1 - \mathsf{str}_2 = 0)$. Let T_{eq} denote the set of such linear equations. We write $\mathsf{Eq} \in \mathsf{Span}(T_{\mathsf{eq}})$ to indicate that an equation Eq lies in the linear span of the equations in T_{eq}.

In the following, we illustrate the full description of our simulator.

Simulator $\mathcal{S}$

$\underline{\mathcal{S}.\mathcal{H}_x(\mathsf{sk})}$:

if $\exists(\mathsf{sk}, x) \in T_{\mathcal{H}_x}$, then **return** x; ..... Case 1

else:

$x \xleftarrow{\$} \mathbb{Z}_N$; $T_{\mathcal{H}_x} \leftarrow T_{\mathcal{H}_x} \cup \{(\mathsf{sk}, x)\}$;

$\mathsf{pk} \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{KG}(\mathsf{sk})$;

$T_{\mathsf{eq}} \leftarrow T_{\mathsf{eq}} \cup \{\mathsf{pk} - x \cdot \diamondsuit = 0\}$;

return x; ..... Case 2

$\underline{\mathcal{S}.\mathcal{H}_y(\mathsf{r})}$:

if $\exists(\mathsf{r}, y) \in T_{\mathcal{H}_y}$, then **return** y; ..... Case 1

else:

$y \xleftarrow{\$} \mathbb{Z}_N$; $T_{\mathcal{H}_y} \leftarrow T_{\mathcal{H}_y} \cup \{(\mathsf{r}, y)\}$;

$(\mathsf{c}, \mathsf{k}) \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{Encaps}(\mathsf{pk}_{\mathsf{test}}, \mathsf{r})$;

if $\exists(\tilde{\mathsf{c}}, \mathsf{c}) \in T_{\mathcal{P}}$: $T_{\mathsf{eq}} \leftarrow T_{\mathsf{eq}} \cup \{\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0\}$;

return y; ..... Case 2

$\underline{\mathcal{S}.\mathcal{H}_{\mathsf{key}}(\mathsf{pk}, \tilde{\mathsf{c}}, \mathsf{str})}$:

if $\exists(\mathsf{pk}\|\tilde{\mathsf{c}}\|\mathsf{str}, \mathsf{k}) \in T_{\mathcal{H}_{\mathsf{key}}}$, then **return** k; ..... Case 1

if $\exists y \in \mathbb{Z}_N$ s.t. $(\mathsf{str} - y \cdot \mathsf{pk} = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$:

if $(\tilde{\mathsf{c}} - y = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$:

if $\exists(\mathsf{r}, y) \in T_{\mathcal{H}_y}$: $(\mathsf{c}, \mathsf{k}) \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$;
$T_{\mathcal{H}_{\mathsf{key}}} \leftarrow T_{\mathcal{H}_{\mathsf{key}}} \cup \{(\mathsf{pk}\|\tilde{\mathsf{c}}\|\mathsf{str}, \mathsf{k})\}$;
return k; ..... Case 2

if $\exists(\mathsf{r}, \tilde{\mathsf{c}}) \in T^*_{\mathsf{r}}$: $(\mathsf{c}, \mathsf{k}) \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$;
$T_{\mathcal{H}_{\mathsf{key}}} \leftarrow T_{\mathcal{H}_{\mathsf{key}}} \cup \{(\mathsf{pk}\|\tilde{\mathsf{c}}\|\mathsf{str}, \mathsf{k})\}$;
return k; ..... Case 3

else: $\mathsf{r} \xleftarrow{\$} \mathbb{Z}_N$; $T^*_{\mathsf{r}} \leftarrow T^*_{\mathsf{r}} \cup \{(\mathsf{r}, \tilde{\mathsf{c}})\}$;
$(\mathsf{c}, \mathsf{k}) \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$;
$T_{\mathcal{H}_{\mathsf{key}}} \leftarrow T_{\mathcal{H}_{\mathsf{key}}} \cup \{(\mathsf{pk}\|\tilde{\mathsf{c}}\|\mathsf{str}, \mathsf{k})\}$;
return k; ..... Case 4

if $\exists x \in \mathbb{Z}_N$ s.t. $(\mathsf{str} - x \cdot \tilde{\mathsf{c}} = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$:

if $\exists(\mathsf{sk}, x) \in T_{\mathcal{H}_x}$ and $(\mathsf{pk} - x \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$:

if $\exists(\tilde{\mathsf{c}}, \mathsf{c}) \in T_{\mathcal{P}}$: $\mathsf{k} \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c})$;
$T_{\mathcal{H}_{\mathsf{key}}} \leftarrow T_{\mathcal{H}_{\mathsf{key}}} \cup \{(\mathsf{pk}\|\tilde{\mathsf{c}}\|\mathsf{str}, \mathsf{k})\}$;
return k; ..... Case 5

if $\exists(\mathsf{r}, \tilde{\mathsf{c}}) \in T^*_{\mathsf{r}}$: $(\mathsf{c}, \mathsf{k}) \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$;
$T_{\mathcal{H}_{\mathsf{key}}} \leftarrow T_{\mathcal{H}_{\mathsf{key}}} \cup \{(\mathsf{pk}\|\tilde{\mathsf{c}}\|\mathsf{str}, \mathsf{k})\}$;
return k; ..... Case 6

else: $\mathsf{r} \xleftarrow{\$} \mathbb{Z}_N$; $T^*_{\mathsf{r}} \leftarrow T^*_{\mathsf{r}} \cup \{(\mathsf{r}, \tilde{\mathsf{c}})\}$;
$(\mathsf{c}, \mathsf{k}) \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$;
$T_{\mathcal{H}_{\mathsf{key}}} \leftarrow T_{\mathcal{H}_{\mathsf{key}}} \cup \{(\mathsf{pk} \| \tilde{\mathsf{c}} \| \mathsf{str}, \mathsf{k})\}$;
return k; ..... Case 7

if $\exists (\mathsf{sk}, \mathsf{pk}) \in T^*_{\mathsf{sk}}$ and $(\mathsf{pk} - x \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$:
if $\exists (\tilde{\mathsf{c}}, \mathsf{c}) \in T_{\mathcal{P}}$: $\mathsf{k} \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c})$;
$T_{\mathcal{H}_{\mathsf{key}}} \leftarrow T_{\mathcal{H}_{\mathsf{key}}} \cup \{(\mathsf{pk} \| \tilde{\mathsf{c}} \| \mathsf{str}, \mathsf{k})\}$;
return k; ..... Case 8

$\mathsf{k} \xleftarrow{\$} \mathcal{K}$; $T_{\mathcal{H}_{\mathsf{key}}} \leftarrow T_{\mathcal{H}_{\mathsf{key}}} \cup \{(\mathsf{str}, \mathsf{k})\}$;

return k; ..... Case 9

$\underline{\mathcal{S}.\mathcal{P}(\tilde{\mathsf{c}})}$:

if $\exists (\tilde{\mathsf{c}}, \mathsf{c}) \in T_{\mathcal{P}}$, then **return** c; ..... Case 1

if $\mathbf{IES\text{-}KEM}.\mathsf{Encaps}(\tilde{\mathsf{c}}, \mathsf{r}_{\mathsf{test}}) = \perp_{\mathsf{pk}}$:
for attempt $= 1$ to λ do:
$\mathsf{c} \xleftarrow{\$} \{0,1\}^{\ell}$;
if $\mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}_{\mathsf{test}}, \mathsf{c}) = \perp_{\mathsf{c}}$ **then break**;
$T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\tilde{\mathsf{c}}, \mathsf{c})\}$;
return c; ..... Case 2

if $\exists y \in \mathbb{Z}_N$ s.t. $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$:
if $\exists (\mathsf{r}, y) \in T_{\mathcal{H}_y}$: $(\mathsf{c}, \mathsf{k}) \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{Encaps}(\mathsf{pk}_{\mathsf{test}}, \mathsf{r})$;
$T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\tilde{\mathsf{c}}, \mathsf{c})\}$;
return c; ..... Case 3

if $\exists (\mathsf{r}, \tilde{\mathsf{c}}) \in T^*_{\mathsf{r}}$: $(\mathsf{c}, \mathsf{k}) \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{Encaps}(\mathsf{pk}_{\mathsf{test}}, \mathsf{r})$;
$T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\tilde{\mathsf{c}}, \mathsf{c})\}$;
return c; ..... Case 4

$\mathsf{r} \xleftarrow{\$} \mathbb{Z}_N$; $T^*_{\mathsf{r}} \leftarrow T^*_{\mathsf{r}} \cup \{(\mathsf{r}, \tilde{\mathsf{c}})\}$;
$\mathsf{c} \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{Encaps}(\mathsf{pk}_{\mathsf{test}}, \mathsf{r})$; $T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\tilde{\mathsf{c}}, \mathsf{c})\}$;
return c; ..... Case 5

$\underline{\mathcal{S}.\mathcal{P}^{-1}(\mathsf{c})}$:

if $\exists (\tilde{\mathsf{c}}, \mathsf{c}) \in T_{\mathcal{P}}$, then **return** $\tilde{\mathsf{c}}$; ..... Case 6

if $\mathbf{KEM}.\mathsf{Decaps}(\mathsf{sk}_{\mathsf{test}}, \mathsf{c}) = \perp_{\mathsf{c}}$:

for attempt $= 1$ to λ do:

$\tilde{\mathsf{c}} \xleftarrow{\$} \{0,1\}^{\ell}$;

if $\mathbf{IES\text{-}KEM}.\mathsf{Encaps}(\tilde{\mathsf{c}}, \mathsf{r}_{\mathsf{test}}) = \perp_{\mathsf{pk}}$ **then break**;

$T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\tilde{\mathsf{c}}, \mathsf{c})\}$;

return $\tilde{\mathsf{c}}$; ..... Case 7

if $\exists(\mathsf{r}, y) \in T_{\mathcal{H}_y}$ s.t. $\mathbf{IES\text{-}KEM}.\mathsf{Encaps}(\mathsf{pk}_{\mathsf{test}}, \mathsf{r}) = (\mathsf{c}, *)$:

if $\exists\tilde{\mathsf{c}} \in \{0,1\}^{\ell}$ s.t. $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$:

$T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\tilde{\mathsf{c}}, \mathsf{c})\}$;

return $\tilde{\mathsf{c}}$; ..... Case 8

else: $\mathsf{sk} \xleftarrow{\$} \mathcal{SK}$; $\tilde{\mathsf{c}} \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{KG}(\mathsf{sk})$; $T^*_{\mathsf{sk}} \leftarrow T^*_{\mathsf{sk}} \cup \{(\mathsf{sk}, \tilde{\mathsf{c}})\}$;

$T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\tilde{\mathsf{c}}, \mathsf{c})\}$; $T_{\mathsf{eq}} \leftarrow T_{\mathsf{eq}} \cup \{\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0\}$;

return $\tilde{\mathsf{c}}$; ..... Case 9

$\mathsf{sk} \xleftarrow{\$} \mathcal{SK}$; $\tilde{\mathsf{c}} \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{KG}(\mathsf{sk})$; $T^*_{\mathsf{sk}} \leftarrow T^*_{\mathsf{sk}} \cup \{(\mathsf{sk}, \tilde{\mathsf{c}})\}$;

$T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup (\tilde{\mathsf{c}}, \mathsf{c})$;

return $\tilde{\mathsf{c}}$; ..... Case 10

$\underline{\mathcal{S}.\mathcal{G}^{\mathsf{label}}(x)}$:

if $\exists(\mathsf{str} - x \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$, then **return** str; ..... Case 1

else:

$\mathsf{sk} \xleftarrow{\$} \mathcal{SK}$; $\mathsf{str} \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{KG}(\mathsf{sk})$; $T^*_{\mathsf{sk}} \leftarrow T^*_{\mathsf{sk}} \cup \{(\mathsf{sk}, \mathsf{str})\}$;

$T_{\mathsf{eq}} \leftarrow T_{\mathsf{eq}} \cup \{\mathsf{str} - x \cdot \diamondsuit = 0\}$;

return str; ..... Case 2

$\underline{\mathcal{S}.\mathcal{G}^{\mathsf{add}}(\mathsf{str}_1, \mathsf{str}_2)}$:

$\phi_1 \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{Encaps}(\mathsf{str}_1, \mathsf{r}_{\mathsf{test}})$;

$\phi_2 \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{Encaps}(\mathsf{str}_2, \mathsf{r}_{\mathsf{test}})$;

if $\phi_1 = \perp_{\mathsf{pk}} \vee \phi_2 = \perp_{\mathsf{pk}}$: **return** $\perp$; ..... Case 1

if $\exists(\mathsf{str} - \mathsf{str}_1 - \mathsf{str}_2 = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$, then **return** str; ..... Case 2

$\mathsf{sk} \xleftarrow{\$} \mathcal{SK}$; $\mathsf{str} \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{KG}(\mathsf{sk})$; $T^*_{\mathsf{sk}} \leftarrow T^*_{\mathsf{sk}} \cup \{(\mathsf{sk}, \mathsf{str})\}$;

$T_{\mathsf{eq}} \leftarrow T_{\mathsf{eq}} \cup \{\mathsf{str} - \mathsf{str}_1 - \mathsf{str}_2 = 0\}$;

return str. ..... Case 3

We first observe that if the differentiator $\mathcal{D}$ issues q queries to the honest and adversarial interfaces, then the simulator $\mathcal{S}$ makes at most $O(\lambda \cdot q)$ queries to the ideal KEM, ensuring its efficiency.

Next, we show that the view of $\mathcal{D}$ in the real world is indistinguishable from its view in the ideal

world. To this end, we define a sequence of hybrid games that gradually transition from the ideal game $\mathbf{Ideal}_{\mathcal{S}}^{\mathbf{IES\text{-}KEM}}$ to the real game $\mathbf{Real}^{\Pi_{\mathsf{IES}}^{\mathcal{O}}}$. Specifically,

1. **Ideal Game**. We begin with the ideal game $\mathbf{Ideal}_{\mathcal{S}}^{\mathbf{IES\text{-}KEM}}$, in which $\mathcal{D}$ interacts with the honest interface $\mathbf{IES\text{-}KEM}$ and the adversarial interface $\mathcal{S}^{\mathbf{IES\text{-}KEM}}$.

2. **Game $\mathbf{G}_1$**. A relay algorithm $\mathcal{F}$ is introduced as an intermediary between $\mathcal{D}$ and ideal KEM. Here, $\mathcal{D}$ interacts with $\mathcal{F}^{\mathbf{IES\text{-}KEM}}$ and $\mathcal{S}^{\mathbf{IES\text{-}KEM}}$, where $\mathcal{F}$ simply forwards queries to $\mathbf{IES\text{-}KEM}$ and returns the corresponding outputs.

3. **Game $\mathbf{G}_2$**. We refine the simulator by introducing several failure conditions on its query–response pairs. Whenever a failure condition is triggered, the modified simulator $\mathcal{S}_{\mathsf{fail}}$ aborts explicitly. In this game, $\mathcal{D}$ interacts with $\mathcal{F}^{\mathbf{IES\text{-}KEM}}$ and $\mathcal{S}_{\mathsf{fail}}^{\mathbf{IES\text{-}KEM}}$.

4. **Game $\mathbf{G}_3$**. The relay algorithm $\mathcal{F}$ is replaced with the construction Π_{IES}, as defined earlier. The construction Π_{IES} employs $\mathcal{S}_{\mathsf{fail}}$ in place of the ideal model. The central goal is to show that, except with negligible probability of triggering a failure condition (from $\mathbf{G}_2$), the responses generated by Π_{IES} remain consistent with those of $\mathbf{IES\text{-}KEM}$. In this game, $\mathcal{D}$ interacts with $\Pi_{\mathsf{IES}}^{\mathcal{S}_{\mathsf{fail}}}$ and $\mathcal{S}_{\mathsf{fail}}^{\mathbf{IES\text{-}KEM}}$.

5. **Game $\mathbf{G}_4$**. The simulator is further modified to sample its responses independently of the ideal KEM. Moreover, the new simulator $\mathcal{S}_{\mathsf{samp}}$ no longer checks for failure conditions. In this game, $\mathcal{D}$ interacts with $\Pi_{\mathsf{IES}}^{\mathcal{S}_{\mathsf{samp}}}$ and $\mathcal{S}_{\mathsf{samp}}$.

6. **Real Game**. Finally, we arrive at the real game $\mathbf{Real}^{\Pi_{\mathsf{IES}}^{\mathcal{O}}}$, in which the simulator $\mathcal{S}_{\mathsf{samp}}$ from $\mathbf{G}_4$ is replaced with the real oracle $\mathcal{O}$. Thus, $\mathcal{D}$ interacts with $\Pi_{\mathsf{IES}}^{\mathcal{O}}$ and $\mathcal{O}$.

In what follows, we give a detailed description of each hybrid game and demonstrate that the statistical distance between any pair of consecutive games is negligible.

Ideal Game. In this game, the differentiator $\mathcal{D}$ interacts with the ideal KEM and the simulator $\mathcal{S}$, as defined previously. The game outputs 1 if and only if $\mathcal{D}$ outputs 1 after this interaction. Formally,

$$\Pr[\mathbf{Ideal}_{\mathcal{S}}^{\mathbf{IES\text{-}KEM}}(\mathcal{D}) = 1] = \Pr[\mathcal{D}^{\mathbf{IES\text{-}KEM},\mathcal{S}^{\mathbf{IES\text{-}KEM}}}(1^\lambda) = 1].$$

D.1 Transition from Ideal to $\mathbf{G}_1$

Game 1. This game is identical to the ideal game, except that the differentiator $\mathcal{D}$ interacts with a relay algorithm $\mathcal{F}$ rather than directly with the ideal KEM. The relay algorithm $\mathcal{F}$ forwards all queries to the ideal KEM and provides the following three interfaces:

- $\mathcal{F}.\mathsf{KG}(\mathsf{sk})$: query $\mathsf{pk} \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{KG}(\mathsf{sk})$ and return pk.
- $\mathcal{F}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$: query $(\mathsf{c}, \mathsf{k}) \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$ and return (c, k).
- $\mathcal{F}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c})$: query $\mathsf{k} \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c})$ and return k.

Since all queries issued by $\mathcal{D}$ are answered exactly as in the ideal game, its view in Game 1 is identical to that in Ideal Game. Defining $\mathbf{G}_1$ as Game 1, which outputs 1 if and only if $\mathcal{D}$ outputs 1 after interacting with $\mathcal{F}$ and $\mathcal{S}$, we obtain

$$\begin{aligned}\Pr[\mathbf{G}_1(\mathcal{D}) = 1] &= \Pr[\mathcal{D}^{\mathcal{F}^{\mathbf{IES\text{-}KEM}},\mathcal{S}^{\mathbf{IES\text{-}KEM}}}(1^\lambda) = 1] \\ &= \Pr[\mathbf{Ideal}_{\mathcal{S}}^{\mathbf{IES\text{-}KEM}}(\mathcal{D}) = 1].\end{aligned}$$

D.2 Transition from $\mathbf{G}_1$ to $\mathbf{G}_2$

Game 2. In this game, the simulator is modified to include conditional checks before returning partial responses. These checks are specifically designed to capture correlations among the simulator's outputs that could potentially be exploited by the differentiator $\mathcal{D}$. If the modified simulator $\mathcal{S}_{\mathsf{fail}}$ generates a response that triggers any of the failure conditions listed below, it explicitly aborts instead of returning that response.

In response to a $\mathcal{H}_x$ query with input sk, the simulator samples a value $x \in \mathbb{Z}_N$ uniformly as the candidate response. Let $\mathsf{pk} \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{KG}(\mathsf{sk})$. The simulator then checks the following failure conditions:

- $\mathsf{C}_{1\text{-}\mathcal{H}_x}(\mathsf{sk}, \mathsf{pk})$: The pair $(\mathsf{sk}, \mathsf{pk})$ already appears in T^*_{sk}.
- $\mathsf{C}_{2\text{-}\mathcal{H}_x}(\mathsf{pk}, x)$: There exists $(\mathsf{str} - x \cdot \diamondsuit = 0)$ in the span of T_{eq} with $\mathsf{str} \neq \mathsf{pk}$.
- $\mathsf{C}_{3\text{-}\mathcal{H}_x}(x)$: There exists $(\mathsf{str}_1 - x \cdot \mathsf{str}_2 = 0)$ in the span of T_{eq}.
- $\mathsf{C}_{4\text{-}\mathcal{H}_x}(\mathsf{pk}, x)$: There exists $(\mathsf{str}_1 - \alpha \cdot \mathsf{str}_2 - \beta \cdot \mathsf{pk} - \gamma \cdot \diamondsuit = 0)$ in the span of T_{eq} such that $\beta \cdot x + \gamma = 0$.
- $\mathsf{C}_{5\text{-}\mathcal{H}_x}(\mathsf{pk}, x)$: There exists $(\mathsf{str} - \alpha \cdot \mathsf{pk} - \beta \cdot \diamondsuit = 0)$ in the span of T_{eq} such that condition $\mathsf{C}_{2\text{-}\mathcal{H}_x}(\mathsf{str}, \alpha \cdot x + \beta)$ holds.
- $\mathsf{C}_{6\text{-}\mathcal{H}_x}(\mathsf{pk}, x)$: There exists $(\mathsf{str} - \alpha \cdot \mathsf{pk} - \beta \cdot \diamondsuit = 0)$ in the span of T_{eq} such that condition $\mathsf{C}_{3\text{-}\mathcal{H}_x}(\alpha \cdot x + \beta)$ holds.
- $\mathsf{C}_{7\text{-}\mathcal{H}_x}(\mathsf{pk}, x)$: There exists $(\mathsf{str} - \alpha \cdot \mathsf{pk} - \beta \cdot \diamondsuit = 0)$ in the span of T_{eq} such that the value $(\alpha \cdot x + \beta)$ already appears in $T_{\mathcal{H}_y}$.
- $\mathsf{C}_{8\text{-}\mathcal{H}_x}(\mathsf{pk}, x)$: There exists $(\mathsf{str} - \alpha \cdot \mathsf{pk} - \beta \cdot \diamondsuit = 0)$ in the span of T_{eq} such that condition $\mathsf{C}_{4\text{-}\mathcal{H}_x}(\mathsf{str}, \alpha \cdot x + \beta)$ holds.

In response to a $\mathcal{H}_y$ query with input r, the simulator samples a value $y \in \mathbb{Z}_N$ uniformly as the candidate response. If there exists $(\tilde{\mathsf{c}}, \mathsf{c}) \in T_{\mathcal{P}}$ associated with r, the simulator additionally inserts the equation $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$ into T_{eq}. It then checks the following failure conditions:

- $\mathsf{C}_{1\text{-}\mathcal{H}_y}(\mathsf{r})$: The value r already appears in T^*_{r}.
- $\mathsf{C}_{2\text{-}\mathcal{H}_y}(y)$: The value y already appears in $T_{\mathcal{H}_y}$.
- $\mathsf{C}_{3\text{-}\mathcal{H}_y}(\tilde{\mathsf{c}}, y)$: There exists $(\mathsf{str} - y \cdot \diamondsuit = 0)$ in the span of T_{eq} with $\mathsf{str} \neq \tilde{\mathsf{c}}$.
- $\mathsf{C}_{4\text{-}\mathcal{H}_y}(y)$: There exists $(\mathsf{str}_1 - y \cdot \mathsf{str}_2 = 0)$ in the span of T_{eq}.
- $\mathsf{C}_{5\text{-}\mathcal{H}_y}(\tilde{\mathsf{c}}, y)$: There exists $(\mathsf{str}_1 - \alpha \cdot \mathsf{str}_2 - \beta \cdot \tilde{\mathsf{c}} - \gamma \cdot \diamondsuit = 0)$ in the span of T_{eq} such that $\beta \cdot y + \gamma = 0$.
- $\mathsf{C}_{6\text{-}\mathcal{H}_y}(\tilde{\mathsf{c}}, y)$: There exists $(\mathsf{str} - \alpha \cdot \tilde{\mathsf{c}} - \beta \cdot \diamondsuit = 0)$ in the span of T_{eq} such that condition $\mathsf{C}_{2\text{-}\mathcal{H}_y}(\alpha \cdot y + \beta)$ holds.
- $\mathsf{C}_{7\text{-}\mathcal{H}_y}(\tilde{\mathsf{c}}, y)$: There exists $(\mathsf{str} - \alpha \cdot \tilde{\mathsf{c}} - \beta \cdot \diamondsuit = 0)$ in the span of T_{eq} such that condition $\mathsf{C}_{3\text{-}\mathcal{H}_y}(\mathsf{str}, \alpha \cdot y + \beta)$ holds.

- $\mathsf{C}_{8\text{-}\mathcal{H}_y}(\tilde{\mathsf{c}}, y)$: There exists $(\mathsf{str} - \alpha \cdot \tilde{\mathsf{c}} - \beta \cdot \diamondsuit = 0)$ in the span of T_{eq} such that condition $\mathsf{C}_{4\text{-}\mathcal{H}_y}(\alpha \cdot y + \beta)$ holds.

- $\mathsf{C}_{9\text{-}\mathcal{H}_y}(\tilde{\mathsf{c}}, y)$: There exists $(\mathsf{str} - \alpha \cdot \tilde{\mathsf{c}} - \beta \cdot \diamondsuit = 0)$ in the span of T_{eq} such that condition $\mathsf{C}_{5\text{-}\mathcal{H}_y}(\mathsf{str}, \alpha \cdot y + \beta)$ holds.

In response to a $\mathcal{H}_{\mathsf{key}}$ query with input $(\mathsf{pk}, \tilde{\mathsf{c}}, \mathsf{str})$, if there exists a value y such that both $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$ and $(\mathsf{str} - y \cdot \mathsf{pk} = 0)$ lie in the span of T_{eq}, and no corresponding r exists, the simulator samples r uniformly at random, generates k as the candidate response by invoking $(\mathsf{c}, \mathsf{k}) \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$, and then checks the following failure condition:

- $\mathsf{C}_{1\text{-}\mathcal{H}_{\mathsf{key}}}(\mathsf{r})$: The value r already appears in $T_{\mathcal{H}_y}$ or T^*_{r}.

- $\mathsf{C}_{2\text{-}\mathcal{H}_{\mathsf{key}}}(\mathsf{c})$: The generated c already appears in $T_{\mathcal{P}}$.

Otherwise, if there exists $(\mathsf{sk}, x) \in T_{\mathcal{H}_x}$ such that both $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ and $(\mathsf{str} - x \cdot \tilde{\mathsf{c}} = 0)$ lie in the span of T_{eq}, but no corresponding r or c exists, the simulator proceeds similarly: it samples r uniformly at random, generates k by invoking $(\mathsf{c}, \mathsf{k}) \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$, and then checks the same failure conditions $\mathsf{C}_{1\text{-}\mathcal{H}_{\mathsf{key}}}(\mathsf{r})$ and $\mathsf{C}_{2\text{-}\mathcal{H}_{\mathsf{key}}}(\mathsf{c})$.

In response to a $\mathcal{P}$ query with input $\tilde{\mathsf{c}}$, the simulator first checks the validity of $\tilde{\mathsf{c}}$ using $\mathbf{IES\text{-}KEM}.\mathsf{Encaps}$. If $\tilde{\mathsf{c}}$ is invalid, the simulator samples an invalid ciphertext c as the candidate response and then checks the following failure conditions:

- $\mathsf{C}_{1\text{-}\mathcal{P}}(\mathsf{c})$: The generated c already appears in $T_{\mathcal{P}}$.

- $\mathsf{C}_{2\text{-}\mathcal{P}}(\mathsf{c})$: The generated c is valid, i.e., $\mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}_{\mathsf{test}}, \mathsf{c}) \neq \perp_{\mathsf{c}}$.

For a valid $\tilde{\mathsf{c}}$, if a corresponding r exists in $T_{\mathcal{H}_y}$ or T^*_{r}, the response is determined by r. Otherwise, the simulator generates a fresh ciphertext c as the candidate response by sampling r uniformly at random and then checks the following failure conditions:

- $\mathsf{C}_{3\text{-}\mathcal{P}}(\mathsf{r})$: The value r already appears in $T_{\mathcal{H}_y}$ or T^*_{r}.

- $\mathsf{C}_{4\text{-}\mathcal{P}}(\mathsf{c})$: The generated c already appears in $T_{\mathcal{P}}$.

In response to a $\mathcal{P}^{-1}$ query with input c, the simulator first checks the validity of c using $\mathbf{IES\text{-}KEM}.\mathsf{Decaps}$. If c is invalid, the simulator samples an invalid public key $\tilde{\mathsf{c}}$ as the candidate response and then checks the following failure conditions:

- $\mathsf{C}_{1\text{-}\mathcal{P}^{-1}}(\tilde{\mathsf{c}})$: The generated $\tilde{\mathsf{c}}$ already appears in $T_{\mathcal{P}}$.

- $\mathsf{C}_{2\text{-}\mathcal{P}^{-1}}(\tilde{\mathsf{c}})$: The generated $\tilde{\mathsf{c}}$ is valid, i.e., $\mathbf{IES\text{-}KEM}.\mathsf{Encaps}(\tilde{\mathsf{c}}, \mathsf{r}_{\mathsf{test}}) \neq \perp_{\mathsf{pk}}$.

For a valid c, the simulator examines whether there exists a pair $(\mathsf{r}, y) \in T_{\mathcal{H}_y}$ such that r corresponds to c. If such a pair exists, the response is set to a group element with y as its discrete logarithm. If no corresponding element exists in T_{eq}, the simulator generates a fresh public key by sampling sk uniformly and setting $\tilde{\mathsf{c}} \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{KG}(\mathsf{sk})$, then checks the following failure condition:

- $\mathsf{C}_{3\text{-}\mathcal{P}^{-1}}(\tilde{\mathsf{c}})$: The generated $\tilde{\mathsf{c}}$ already appears in T_{eq}, T^*_{r}, $T_{\mathcal{P}}$, or $T_{\mathcal{H}_{\mathsf{key}}}$.

- $\mathsf{C}_{4\text{-}\mathcal{P}^{-1}}(\mathsf{sk})$: The value sk already appears in $T_{\mathcal{H}_x}$.

If no corresponding (r, y) exists in $T_{\mathcal{H}_y}$, the simulator similarly generates a fresh public key $\tilde{\mathsf{c}}$ by sampling sk uniformly and checks the failure conditions $\mathsf{C}_{3\text{-}\mathcal{P}^{-1}}(\tilde{\mathsf{c}})$ and $\mathsf{C}_{4\text{-}\mathcal{P}^{-1}}(\mathsf{sk})$.

In response to a $\mathcal{G}$ query for which no corresponding entry exists in $\mathsf{Span}(T_{\mathsf{eq}})$, the simulator generates a fresh public key by uniformly sampling sk as the candidate response. Let $\mathsf{str} \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{KG}(\mathsf{sk})$. The simulator then checks the following failure conditions:

- $\mathsf{C}_{1\text{-}\mathcal{G}}(\mathsf{str})$: The generated str already appears in T_{eq}, T^*_{r}, $T_{\mathcal{P}}$, or $T_{\mathcal{H}_{\mathsf{key}}}$.
- $\mathsf{C}_{2\text{-}\mathcal{G}}(\mathsf{sk})$: The value sk already appears in $T_{\mathcal{H}_x}$.

Intuitively, each of the above conditions corresponds to an unwanted collision or algebraic relation between a freshly sampled element and those already recorded in the simulator's tables. Each such event occurs with only negligible probability. A detailed probabilistic analysis is provided in Appendix D.7.

Lemma D.1. $\Pr[\text{Failure Conditions Hold}] \leq O\left(q^4/N + q^2/2^\lambda\right)$.

The differentiator can distinguish between Game 1 and Game 2 only when the following two events occur:

- In Game 1, the simulator $\mathcal{S}$ does not fail, even when its response satisfies one of the failure conditions above;
- In Game 2, the simulator $\mathcal{S}_{\mathsf{fail}}$ does fail.

Let $\mathbf{G}_2$ denoted Game 2, which outputs 1 if and only if $\mathcal{D}$ outputs 1 after interacting with $\mathcal{F}$ and $\mathcal{S}_{\mathsf{fail}}$. Formally,

$$\Pr[\mathbf{G}_2(\mathcal{D})] = \Pr[\mathcal{D}^{\mathcal{F}^{\mathbf{IES\text{-}KEM}}, \mathcal{S}_{\mathsf{fail}}^{\mathbf{IES\text{-}KEM}}}(1^\lambda) = 1].$$

Then

$$\begin{aligned} |\Pr[\mathbf{G}_2(\mathcal{D}) = 1] - \Pr[\mathbf{G}_1(\mathcal{D}) = 1]| &\leq 2 \cdot \Pr[\text{Failure Conditions Hold}] \\ &\leq O\left(\frac{q^4}{N} + \frac{q^2}{2^\lambda}\right). \end{aligned}$$

D.3 Properties under the absence of failure conditions

Before describing Game 3, we state several propositions that record basic observations and will be useful in the proofs of the subsequent lemma.

Proposition D.1. *Suppose that the conditions* $\mathsf{C}_{1\text{-}\mathcal{H}_x}$, $\mathsf{C}_{1\text{-}\mathcal{H}_y}$, $\mathsf{C}_{1\text{-}\mathcal{H}_{\mathsf{key}}}$, $\mathsf{C}_{3\text{-}\mathcal{P}}$, $\mathsf{C}_{3,4\text{-}\mathcal{P}^{-1}}$ *and* $\mathsf{C}_{1,2\text{-}\mathcal{G}}$ *do not hold. Then for every equation* $(\mathsf{str} - x \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$*, there must exist either a pair* (sk, x) *in* $T_{\mathcal{H}_x}$ *or a pair* $(\mathsf{sk}, \mathsf{str})$ *in* T^*_{sk} *such that* $\mathsf{str} = \mathbf{IES\text{-}KEM}.\mathsf{KG}(\mathsf{sk})$.

Proof. As described for the simulator, the equations recorded in T_{eq} can be classified into five types:

- **Type 1:** An equation of the form $(\mathsf{str} - x \cdot \diamondsuit = 0)$, introduced when processing a $\mathcal{G}^{\mathsf{label}}$ query, where $\mathsf{str} = \mathbf{IES\text{-}KEM}.\mathsf{KG}(\mathsf{sk})$ with sk sampled uniformly at random from $\mathbb{Z}_N$.
- **Type 2:** An equation of the form $(\mathsf{str} - \mathsf{str}_1 - \mathsf{str}_2 = 0)$, introduced when processing a $\mathcal{G}^{\mathsf{add}}$ query, where $\mathsf{str} = \mathbf{IES\text{-}KEM}.\mathsf{KG}(\mathsf{sk})$ with sk sampled uniformly at random from $\mathbb{Z}_N$.
- **Type 3:** An equation of the form $(\mathsf{pk} - x \cdot \diamondsuit = 0)$, introduced when processing a $\mathcal{H}_x$ query, where x is sampled uniformly at random from $\mathbb{Z}_N$.
- **Type 4:** An equation of the form $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$, introduced when processing a $\mathcal{H}_y$ query, where y is sampled uniformly at random from $\mathbb{Z}_N$.

- **Type 5:** An equation of the form $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$, introduced when processing a $\mathcal{P}^{-1}$ query, where $\tilde{\mathsf{c}} = \mathbf{IES\text{-}KEM}.\mathsf{KG}(\mathsf{sk})$ with sk sampled uniformly at random from $\mathbb{Z}_N$.

For an equation Eq of the form $(\mathsf{str} - x \cdot \diamondsuit = 0)$, corresponding to Type 1 (with Type 5 being analogous), the pair $(\mathsf{sk}, \mathsf{str})$ with $\mathsf{str} = \mathbf{IES\text{-}KEM}.\mathsf{KG}(\mathsf{sk})$ is recorded in T^*_{sk}. Provided that str does not already appear in T_{eq} (i.e., conditions $\mathsf{C}_{1\text{-}\mathcal{G}}$ and $\mathsf{C}_{3\text{-}\mathcal{P}^{-1}}$ do not hold), this insertion does not endow any other group elements with discrete logarithms.

For an equation Eq of the form $(\mathsf{str} - \mathsf{str}_1 - \mathsf{str}_2 = 0)$, corresponding to Type 2, if both str_1 and str_2 already have discrete logarithms, the situation is analogous to Type 1. Otherwise, provided that condition $\mathsf{C}_{1\text{-}\mathcal{G}}$ does not hold, inserting Eq does not assign discrete logarithms to any additional group elements in T_{eq}.

For an equation Eq of the form $(\mathsf{pk} - x \cdot \diamondsuit = 0)$, corresponding to Type 3, the pair (sk, x) is recorded in $T_{\mathcal{H}_x}$ with $\mathsf{pk} = \mathbf{IES\text{-}KEM}.\mathsf{KG}(\mathsf{sk})$. Inserting Eq may also endow other group elements with discrete logarithms. If this occurs, and provided that no sk appears simultaneously in both $T_{\mathcal{H}_x}$ and T^*_{sk} (i.e., conditions $\mathsf{C}_{1\text{-}\mathcal{H}_x}$, $\mathsf{C}_{4\text{-}\mathcal{P}^{-1}}$ and $\mathsf{C}_{2\text{-}\mathcal{G}}$ do not hold), then there must exist an equation

$$\left(\widetilde{\mathsf{str}} - \widetilde{\mathsf{str}}_1 - \widetilde{\mathsf{str}}_2 = 0\right) \in T_{\mathsf{eq}}$$

such that $\widetilde{\mathsf{str}}_1 = \mathsf{pk}$ and $(\widetilde{\mathsf{str}}_2 - \tilde{x}_2 \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$. In particular, the insertion of Eq yields

$$\left(\widetilde{\mathsf{str}} - (x + \tilde{x}_2) \cdot \diamondsuit = 0\right) \in \mathsf{Span}(T_{\mathsf{eq}}),$$

so that $\widetilde{\mathsf{str}}$ acquires a discrete logarithm and $(\widetilde{\mathsf{sk}}, \widetilde{\mathsf{str}})$ is added to T^*_{sk}. Moreover, $\widetilde{\mathsf{str}}$ may in turn enable additional group elements to acquire discrete logarithms in the same manner, wherein each group element is likewise associated with a private key recorded in T^*_{sk}.

For an equation Eq of the form $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$, corresponding to Type 4, there must exist a pair $(\tilde{\mathsf{c}}, \mathsf{c})$ in $T_{\mathcal{P}}$, where c corresponds to the input r of the $\mathcal{H}_y$ query. Such a pair cannot originate from a $\mathcal{P}$ query; otherwise, r would necessarily appear in both $T_{\mathcal{H}_y}$ and T^*_{r}, thereby triggering condition $\mathsf{C}_{1\text{-}\mathcal{H}_y}$, $\mathsf{C}_{1\text{-}\mathcal{H}_{\mathsf{key}}}$, or $\mathsf{C}_{3\text{-}\mathcal{P}}$. If the pair was inserted via a $\mathcal{P}^{-1}$ query, then $\tilde{\mathsf{c}}$ must have been generated by sampling sk uniformly, in which case the simulator simultaneously records $(\mathsf{sk}, \tilde{\mathsf{c}})$ in T^*_{sk}. Finally, if the insertion of Eq also endows other group elements with discrete logarithms, the situation is analogous to that described for Type 3. □ □

Proposition D.2. *Suppose that the conditions* $\mathsf{C}_{1,2,5\text{-}\mathcal{H}_x}$, $\mathsf{C}_{1,3,7\text{-}\mathcal{H}_y}$, $\mathsf{C}_{1\text{-}\mathcal{H}_{\mathsf{key}}}$, $\mathsf{C}_{3\text{-}\mathcal{P}}$, $\mathsf{C}_{3,4\text{-}\mathcal{P}^{-1}}$ *and* $\mathsf{C}_{1,2\text{-}\mathcal{G}}$ *do not hold. Then* T_{eq} *simulates a proper view of the generic group.*

Proof. A proper simulation of the generic group with respect to T_{eq} requires that the following two bad events do not occur:

- DLog-Coll: There exist $(\mathsf{str}_1 - x \cdot \diamondsuit = 0)$ and $(\mathsf{str}_2 - x \cdot \diamondsuit = 0)$ in $\mathsf{Span}(T_{\mathsf{eq}})$ with $\mathsf{str}_1 \neq \mathsf{str}_2$.
- Multi-DLog: There exist $(\mathsf{str} - x_1 \cdot \diamondsuit = 0)$ and $(\mathsf{str} - x_2 \cdot \diamondsuit = 0)$ in $\mathsf{Span}(T_{\mathsf{eq}})$ with $x_1 \neq x_2$.

Write $\mathsf{Cons}(T_{\mathsf{eq}})$ to denote that neither bad event occurs for T_{eq}. Clearly $\mathsf{Cons}(T_{\mathsf{eq}})$ holds when T_{eq} is empty or contains a single equation. Assume $\mathsf{Cons}(T_{\mathsf{eq}})$ holds prior to inserting a new equation Eq. We show that, provided no failure condition is triggered, $\mathsf{Cons}(T_{\mathsf{eq}} \cup \mathsf{Eq})$ still holds.

If Eq is generated by $\mathcal{P}^{-1}$ or $\mathcal{G}^{\mathsf{label}}$ in the form $(\mathsf{str} - x \cdot \diamondsuit = 0)$, or by $\mathcal{G}^{\mathsf{add}}$ in the form $(\mathsf{str} - \mathsf{str}_1 - \mathsf{str}_2 = 0)$, then, as long as the newly generated str does not already appear in T_{eq} (i.e., conditions $\mathsf{C}_{1\text{-}\mathcal{G}}$ and $\mathsf{C}_{3\text{-}\mathcal{P}^{-1}}$ are not met), the insertion of Eq introduces no new algebraic relations among the existing group elements recorded in T_{eq}. Hence neither DLog-Coll nor Multi-DLog can arise in this case.

By contrast, if Eq is inserted by $\mathcal{H}_x(\mathsf{sk})$ as $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ with x sampled uniformly at random, then the insertion may directly cause either DLog-Coll or Multi-DLog. In the former event the failure condition $\mathsf{C}_{2\text{-}\mathcal{H}_x}$ is triggered. In the latter event Proposition D.1 implies the existence of $(\mathsf{sk}, \mathsf{pk})$ in T^*_{sk}; because sk is recorded in $T_{\mathcal{H}_x}$ as well, condition $\mathsf{C}_{1\text{-}\mathcal{H}_x}$ is forced to hold.

Furthermore, the insertion of Eq may also endow additional group elements with random discrete logarithms. As noted in Proposition D.1, there must exist an equation

$$(\mathsf{str} - \alpha \cdot \mathsf{pk} - \beta \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}}),$$

so that the insertion of Eq induces

$$(\mathsf{str} - (\alpha \cdot x + \beta) \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}}).$$

The random discrete logarithm assigned to str may, however, collide with existing discrete logarithms, in which case the failure condition $\mathsf{C}_{5\text{-}\mathcal{H}_x}$ is triggered.

Similarly, if Eq is inserted by $\mathcal{H}_y(\mathsf{r})$ in the form $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$, as noted in Proposition D.1, there must exist a pair $(\tilde{\mathsf{c}}, \mathsf{c})$ in $T_{\mathcal{P}}$ that was inserted via a $\mathcal{P}^{-1}$ query by generating $\tilde{\mathsf{c}}$ randomly. Provided that conditions $\mathsf{C}_{1\text{-}\mathcal{H}_x}$, $\mathsf{C}_{3,4\text{-}\mathcal{P}^{-1}}$ and $\mathsf{C}_{1,2\text{-}\mathcal{G}}$ do not hold, the group element $\tilde{\mathsf{c}}$ has no specific discrete logarithm in T_{eq} prior to the insertion of Eq. The insertion of Eq may directly trigger DLog-Coll, satisfying condition $\mathsf{C}_{3\text{-}\mathcal{H}_y}$. Analogously, Eq may assign random discrete logarithms to additional group elements, which may collide with existing discrete logarithms, in which case the failure condition $\mathsf{C}_{7\text{-}\mathcal{H}_y}$ is triggered. □ □

Proposition D.3. *Suppose that none of the failure conditions holds. Then there do not exist two distinct pairs* $(\tilde{\mathsf{c}}_1, \mathsf{c})$ *and* $(\tilde{\mathsf{c}}_2, \mathsf{c})$ *in* $T_{\mathcal{P}}$ *with* $\tilde{\mathsf{c}}_1 \neq \tilde{\mathsf{c}}_2$.

Proof. For any pair $(\tilde{\mathsf{c}}, \mathsf{c})$ inserted by the $\mathcal{P}(\tilde{\mathsf{c}})$ query, rather than Case 1 where the corresponding response is already recorded in $T_{\mathcal{P}}$, the simulator proceeds according to one of the remaining three cases, which correspond to those already specified in the simulator's description.

Case 2. If $\tilde{\mathsf{c}}$ is invalid, the simulator repeatedly samples ciphertexts c uniformly at random until an invalid one is obtained, making at most λ attempts.

Case 3. If $\tilde{\mathsf{c}}$ is valid and there exists $y \in \mathbb{Z}_N$ such that $(\mathsf{r}, y) \in T_{\mathcal{H}_y}$ and $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$, then the simulator generates the response c by invoking $\mathbf{IES\text{-}KEM}.\mathsf{Encaps}$ with r.

Case 4. If $\tilde{\mathsf{c}}$ is valid and there exists a pair $(\mathsf{r}, \tilde{\mathsf{c}})$ in T^*_{r}, then the simulator generates the response c by invoking $\mathbf{IES\text{-}KEM}.\mathsf{Encaps}$ with r.

Case 5. If $\tilde{\mathsf{c}}$ is valid but neither of the above applies, the simulator samples r uniformly at random, generates c by invoking $\mathbf{IES\text{-}KEM}.\mathsf{Encaps}$ with r, and records $(\mathsf{r}, \tilde{\mathsf{c}})$ in T^*_{r}.

If the pair $(\tilde{\mathsf{c}}, \mathsf{c})$ is inserted by the $\mathcal{P}^{-1}(\mathsf{c})$ query, rather than Case 6 where the corresponding response is already recorded in $T_{\mathcal{P}}$, the simulator proceeds according to one of the remaining four cases, which correspond to those already specified in the simulator's description.

Case 7. If c is invalid, the simulator repeatedly samples public key $\tilde{\mathsf{c}}$ uniformly at random until an invalid one is obtained, making at most λ attempts.

Case 8. If c is valid and there exists r corresponding to c, such that $(\mathsf{r}, y) \in T_{\mathcal{H}_y}$ and $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$, then the simulator responds with $\tilde{\mathsf{c}}$.

Case 9. If c is valid and there exists only a pair $(\mathsf{r}, y) \in T_{\mathcal{H}_y}$, the simulator samples sk uniformly at random, generates $\tilde{\mathsf{c}}$ by invoking $\mathbf{IES\text{-}KEM}.\mathsf{KG}$ with sk, and records $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$ in T_{eq}.

Case 10. If c is valid but neither of the above applies, the simulator samples sk uniformly at random, and generates $\tilde{\mathsf{c}}$ by invoking $\mathbf{IES\text{-}KEM}.\mathsf{KG}$ with sk.

Suppose there exist two distinct pairs $(\tilde{\mathsf{c}}_1, \mathsf{c})$ and $(\tilde{\mathsf{c}}_2, \mathsf{c})$ in $T_{\mathcal{P}}$. Then $(\tilde{\mathsf{c}}_1, \mathsf{c})$ must have been inserted first, either by a $\mathcal{P}$ or $\mathcal{P}^{-1}$ query, and only afterwards $(\tilde{\mathsf{c}}_2, \mathsf{c})$ must have been inserted by a $\mathcal{P}$ query. Any such collision necessarily implies that one of the failure conditions is triggered. Specifically:

- If $(\tilde{\mathsf{c}}_1, \mathsf{c})$ is first inserted by Cases 2 or 7 and then $(\tilde{\mathsf{c}}_2, \mathsf{c})$ is inserted by Case 2, the randomly sampled invalid c in Case 2 must collide with an existing ciphertext in $T_{\mathcal{P}}$ (i.e., condition $\mathsf{C}_{1\text{-}\mathcal{P}}$ is triggered).

- If $(\tilde{c}_1, c)$ is first inserted by Case 2 and then $(\tilde{c}_2, c)$ is inserted by Case 3, 4, or 5, this implies that in Case 2, even after λ iterations, the sampled ciphertext remains valid (i.e., condition $\mathsf{C}_{2\text{-}\mathcal{P}}$ is triggered).
- If $(\tilde{c}_1, c)$ is first inserted by Case 3, 4, 5, 8, 9, or 10 and then $(\tilde{c}_2, c)$ is inserted by Case 2, condition $\mathsf{C}_{2\text{-}\mathcal{P}}$ is triggered as well.
- If $(\tilde{c}_1, c)$ is first inserted by Cases 3, 8, or 9 and then $(\tilde{c}_2, c)$ is inserted by Case 3, both $\tilde{c}_1$ and $\tilde{c}_2$ would yield the same discrete logarithm in T_{eq}, contradicting Proposition D.2.
- If $(\tilde{c}_1, c)$ is first inserted by Cases 3, 8, or 9 and then $(\tilde{c}_2, c)$ is inserted by Cases 4 or 5, there must exist one r that simultaneously appears in both $T_{\mathcal{H}_y}$ and T^*_r (i.e., conditions $\mathsf{C}_{1\text{-}\mathcal{H}_y}$, $\mathsf{C}_{1\text{-}\mathcal{H}_{\mathsf{key}}}$, or $\mathsf{C}_{3\text{-}\mathcal{P}}$ are triggered).
- If $(\tilde{c}_1, c)$ is first inserted by Cases 4 or 5 and then $(\tilde{c}_2, c)$ is inserted by Case 3, conditions $\mathsf{C}_{1\text{-}\mathcal{H}_y}$, $\mathsf{C}_{1\text{-}\mathcal{H}_{\mathsf{key}}}$, or $\mathsf{C}_{3\text{-}\mathcal{P}}$ are triggered as well.
- If both $(\tilde{c}_1, c)$ and $(\tilde{c}_2, c)$ are inserted by Cases 4 or 5, there must exist both $(r, \tilde{c}_1)$ and $(r, \tilde{c}_2)$ in T^*_r, thus triggering $\mathsf{C}_{1\text{-}\mathcal{H}_{\mathsf{key}}}$ or $\mathsf{C}_{3\text{-}\mathcal{P}}$.
- If $(\tilde{c}_1, c)$ is first inserted by Case 10 and then $(\tilde{c}_2, c)$ is inserted by Case 3, both $\tilde{c}_1$ and $\tilde{c}_2$ would yield the same discrete logarithm in T_{eq}, contradicting Proposition D.2.
- If $(\tilde{c}_1, c)$ is first inserted by Case 10 and then $(\tilde{c}_2, c)$ is inserted by Case 4, either the randomly generated $\tilde{c}$ in Case 10 collides with the one in T^*_r, or the randomly sampled r corresponds to a ciphertext in $T_{\mathcal{P}}$ (i.e., $\mathsf{C}_{3\text{-}\mathcal{P}^{-1}}$ or $\mathsf{C}_{2\text{-}\mathcal{H}_{\mathsf{key}}}$ are triggered).
- If $(\tilde{c}_1, c)$ is first inserted by Case 10 and then $(\tilde{c}_2, c)$ is inserted by Case 5, the randomly sampled c in Case 5 collides with an existing ciphertext in $T_{\mathcal{P}}$ (i.e., condition $\mathsf{C}_{4\text{-}\mathcal{P}}$ is triggered).

□
□

Proposition D.4. *Suppose that none of the failure conditions holds. Then there do not exist two distinct pairs* $(\tilde{c}, c_1)$ *and* $(\tilde{c}, c_2)$ *in* $T_{\mathcal{P}}$ *with* $c_1 \neq c_2$.

Proof. Suppose there exist two distinct pairs $(\tilde{c}, c_1)$ and $(\tilde{c}, c_2)$ in $T_{\mathcal{P}}$. Then $(\tilde{c}, c_1)$ must have been inserted first, either by a $\mathcal{P}$ or $\mathcal{P}^{-1}$ query, and only afterwards $(\tilde{c}, c_2)$ must have been inserted by a $\mathcal{P}^{-1}$ query. Any such collision necessarily implies that one of the failure conditions is triggered. Specifically:

- If $(\tilde{c}, c_1)$ is first inserted by a $\mathcal{P}$ or $\mathcal{P}^{-1}$ query, and later $(\tilde{c}, c_2)$ is inserted by Case 7, then the randomly sampled $\tilde{c}$ in Case 7 must collide with an existing string in $T_{\mathcal{P}}$ (i.e., condition $\mathsf{C}_{1\text{-}\mathcal{P}^{-1}}$ is triggered).
- If $(\tilde{c}, c_1)$ is first inserted by Case 7, and later $(\tilde{c}, c_2)$ is inserted by Case 8, 9, or 10, this implies that in Case 7, even after λ iterations, the randomly sampled $\tilde{c}$ remains valid (i.e., condition $\mathsf{C}_{2\text{-}\mathcal{P}^{-1}}$ is triggered).
- If $(\tilde{c}, c_1)$ is first inserted by Case 3, 8, 9, or 10, and later $(\tilde{c}, c_2)$ is inserted by Case 8, then either the randomly sampled y by $\mathcal{H}_y$ collides with one already in $T_{\mathcal{H}_y}$ (i.e., condition $\mathsf{C}_{2\text{-}\mathcal{H}_y}$ is triggered), or some group element in T_{eq} obtains two discrete logarithms, which contradicts Proposition D.2.
- If $(\tilde{c}, c_1)$ is first inserted by Case 3, 4, 5, 8, 9, or 10, and later $(\tilde{c}, c_2)$ is inserted by Case 9 or 10, then the randomly sampled $\tilde{c}$ in Case 9 or 10 must collide with an existing string in $T_{\mathcal{P}}$ (i.e., condition $\mathsf{C}_{3\text{-}\mathcal{P}^{-1}}$ is triggered).
- If $(\tilde{c}, c_1)$ is first inserted by Case 4, and later $(\tilde{c}, c_2)$ is inserted by Case 9, then the randomly sampled y by $\mathcal{H}_y$ must collide with an existing discrete logarithm in T_{eq} (i.e., condition $\mathsf{C}_{3\text{-}\mathcal{H}_y}$ is triggered).
- If $(\tilde{c}, c_1)$ is first inserted by Case 5, and later $(\tilde{c}, c_2)$ is inserted by Case 9, then the randomly generated str by $\mathcal{G}$ must already appear in $T_{\mathcal{P}}$ (i.e., condition $\mathsf{C}_{1\text{-}\mathcal{G}}$ is triggered).

□
□

Proposition D.5. *Suppose that none of the failure conditions hold. Then for any* $(\mathsf{sk}, x) \in T_{\mathcal{H}_x}$ *and* $(\mathsf{str} - x \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$*, we have* $\mathsf{str} = \mathbf{IES\text{-}KEM}.\mathsf{KG}(\mathsf{sk})$.

Proof. When the pair (sk, x) is recorded in $T_{\mathcal{H}_x}$, the simulator simultaneously inserts $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ into T_{eq} with $\mathsf{pk} = \mathbf{IES\text{-}KEM}.\mathsf{KG}(\mathsf{sk})$. Suppose, for the sake of contradiction, that $\mathsf{pk} \neq \mathsf{str}$. Then both $(\mathsf{str} - x \cdot \diamondsuit = 0)$ and $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ lie in the span of T_{eq}, which implies that T_{eq} fails to simulate a proper view of the generic group. Since no failure conditions are assumed to hold, this contradicts Proposition D.2. Therefore, it must be the case that $\mathsf{str} = \mathbf{IES\text{-}KEM}.\mathsf{KG}(\mathsf{sk})$, as claimed. □ □

To facilitate the subsequent analysis, we use the symbol "$\prec$" to indicate the order in which elements are recorded in different tables by the simulator. For instance, when Item_1 is recorded in T_1 prior to Item_2 being recorded in T_2, we write

$$\mathsf{Item}_1 \in \mathsf{T}_1 \prec \mathsf{Item}_2 \in \mathsf{T}_2.$$

If the context is sufficiently clear, this can be abbreviated as $\mathsf{Item}_1 \prec \mathsf{Item}_2$.

Proposition D.6. *Suppose that none of the failure conditions hold. Then for any tuple* (r, y) *in* $T_{\mathcal{H}_y}$*, equation* $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$ *in the span of* T_{eq}*, and tuple* $(\tilde{\mathsf{c}}, \mathsf{c})$ *in* $T_{\mathcal{P}}$*, it holds that*

$$(\mathsf{c}, *) = \mathbf{IES\text{-}KEM}.\mathsf{Encaps}(*, \mathsf{r}).$$

Proof. When the simulator inserts (r, y) into $T_{\mathcal{H}_y}$, provided that the uniformly sampled y does not collide with any existing discrete logarithm in T_{eq} (i.e., the failure condition $\mathsf{C}_{3\text{-}\mathcal{H}_y}$ is not triggered), we obtain

$$(\mathsf{r}, y) \prec (\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0),$$

where the equation $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$ may result from the insertion of an equation generated by $\mathcal{P}^{-1}$, $\mathcal{G}$, $\mathcal{H}_x$, or $\mathcal{H}_y$.

Type 1: $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$ *induced by an equation* Eq *from* $\mathcal{P}^{-1}$. Provided that the group element generated by $\mathcal{P}^{-1}$ does not already appear in T_{eq} (i.e., condition $\mathsf{C}_{3\text{-}\mathcal{P}^{-1}}$ is not triggered), the insertion does not assign discrete logarithms to any additional group elements. Hence, we must have $\mathsf{Eq} = (\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$. Therefore, as long as no collision occurs in $T_{\mathcal{H}_y}$ or $T_{\mathcal{P}}$, it follows that $(\tilde{\mathsf{c}}, \mathsf{c}) \in T_{\mathcal{P}}$ with $(\mathsf{c}, *) = \mathbf{IES\text{-}KEM}.\mathsf{Encaps}(*, \mathsf{r})$.

Type 2: $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$ *induced by an equation* Eq *from* $\mathcal{G}$. Analogous to Type 1, provided that condition $\mathsf{C}_{1\text{-}\mathcal{G}}$ is not triggered, we have $\mathsf{Eq} = (\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$ with $\tilde{\mathsf{c}}$ randomly generated by $\mathcal{G}$. If $\tilde{\mathsf{c}}$ does not already appear in $T_{\mathcal{P}}$ (i.e., condition $\mathsf{C}_{1\text{-}\mathcal{G}}$ is not triggered), then

$$(\mathsf{r}, y) \prec (\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0) \prec (\tilde{\mathsf{c}}, \mathsf{c}).$$

When $(\tilde{\mathsf{c}}, \mathsf{c})$ is inserted by $\mathcal{P}$, the simulator ensures that c corresponds to r. Otherwise, if $(\tilde{\mathsf{c}}, \mathsf{c})$ is inserted by $\mathcal{P}^{-1}$ and c does not correspond to r, this is equivalent to the existence of a c-collision in $T_{\mathcal{P}}$, which is shown to be impossible in Proposition D.4.

Type 3: $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$ *induced by an equation* Eq *from* $\mathcal{H}_x$. Let $\mathsf{Eq} := (\mathsf{pk} - x \cdot \diamondsuit = 0)$, where x is sampled uniformly at random by $\mathcal{H}_x$. As noted earlier, there must exist an equation

$$(\tilde{\mathsf{c}} - \alpha \cdot \mathsf{pk} - \beta \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}}),$$

so that the insertion of Eq induces

$$(\tilde{\mathsf{c}} - (\alpha \cdot x + \beta) \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}}),$$

where $y = a \cdot x + b$. Consequently, the failure condition $\mathsf{C}_{7\text{-}\mathcal{H}_x}$ is triggered.

Type 4: $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$ *induced by an equation* Eq *from* $\mathcal{H}_y$. If the equation is induced during the insertion of (r, y) into $T_{\mathcal{H}_y}$, i.e., $\mathsf{Eq} = (\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$, and no collision exists in $T_{\mathcal{P}}$, then there exists $(\tilde{\mathsf{c}}, \mathsf{c}) \in T_{\mathcal{P}}$ with $(\mathsf{c}, *) = \mathbf{IES\text{-}KEM}.\mathsf{Encaps}(*, \mathsf{r})$. Otherwise, as in Type 3, $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$ cannot be induced without triggering the failure condition $\mathsf{C}_{6\text{-}\mathcal{H}_y}$.

Thus, by considering all cases above, we conclude that, as long as none of the failure conditions occur, the simulator assigns $(\mathsf{c}, *) = \mathbf{IES\text{-}KEM}.\mathsf{Encaps}(*, \mathsf{r})$ as stated in the proposition. □ □

Proposition D.7. *Suppose that none of the failure conditions hold. Then for any tuple* (r, y) *in* $T_{\mathcal{H}_y}$*, equation* $(\tilde{\mathsf{c}}_1 - y \cdot \diamondsuit = 0)$ *in the span of* T_{eq}*, and tuple* $(\tilde{\mathsf{c}}_2, \mathsf{c})$ *in* $T_{\mathcal{P}}$ *with* $(\mathsf{c}, *) = \mathbf{IES\text{-}KEM}.\mathsf{Encaps}(*, \mathsf{r})$*, it holds that* $\tilde{\mathsf{c}}_1 = \tilde{\mathsf{c}}_2$.

Proof. We prove that $\tilde{\mathsf{c}}_1 = \tilde{\mathsf{c}}_2$ must hold regardless of whether $(\tilde{\mathsf{c}}_2, \mathsf{c})$ is inserted by $\mathcal{P}$ or by $\mathcal{P}^{-1}$, unless some failure condition is triggered.

Type 1. If $(\tilde{\mathsf{c}}_2, \mathsf{c})$ is inserted by $\mathcal{P}$, provided that no r appears simultaneously in both $T_{\mathcal{H}_y}$ and T^*_{r}, the pair $(\tilde{\mathsf{c}}_2, \mathsf{c})$ can only be inserted through Case 3 of $\mathcal{P}$, in which $(\tilde{\mathsf{c}}_2 - y \cdot \diamondsuit = 0)$ lies in the span of T_{eq}. If $\tilde{\mathsf{c}}_1 \neq \tilde{\mathsf{c}}_2$, then two distinct group elements would correspond to the same discrete logarithm, contradicting Proposition D.2.

Type 2. If $(\tilde{\mathsf{c}}_2, \mathsf{c})$ is inserted by $\mathcal{P}^{-1}$, by the simulator's design, regardless of whether (r, y) is added to $T_{\mathcal{H}_y}$ before $(\tilde{\mathsf{c}}_2, \mathsf{c})$ is added to $T_{\mathcal{P}}$, or vice versa, the equation $(\tilde{\mathsf{c}}_2 - y \cdot \diamondsuit = 0)$ necessarily lies in the span of T_{eq}. Thus, if $\tilde{\mathsf{c}}_1 \neq \tilde{\mathsf{c}}_2$, this again contradicts Proposition D.2. □ □

Proposition D.8. *Suppose that none of the failure conditions hold. Then for any tuple* (r, y) *in* $T_{\mathcal{H}_y}$*, equations* $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$ *and* $(\mathsf{str} - y \cdot \mathsf{pk} = 0)$ *in the span of* T_{eq}*, and tuple* $(\mathsf{pk} \| \tilde{\mathsf{c}} \| \mathsf{str}, \mathsf{k})$ *in* $T_{\mathcal{H}_{\mathsf{key}}}$*, it holds that*

$$(*, \mathsf{k}) = \mathbf{IES\text{-}KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r}).$$

Proof. As in Proposition D.6, we have $(\mathsf{r}, y) \prec (\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$, where the equation $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$ may arise from the insertion of an equation generated by one of four interfaces. In what follows, we analyze each possibility in turn.

Type 1. If $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$ is induced by the insertion of $\mathsf{Eq} := (\mathsf{pk} - x \cdot \diamondsuit = 0)$ from $\mathcal{H}_x$, then, as shown in Type 3 of Proposition D.6, the failure condition $\mathsf{C}_{7\text{-}\mathcal{H}_x}$ is necessarily triggered.

Type 2. If $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$ is induced by the insertion of Eq from $\mathcal{H}_y$, provided that failure condition $\mathsf{C}_{6\text{-}\mathcal{H}_y}$ does not occur, there must be $\mathsf{Eq} := (\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$. Thus we have

$$(\tilde{\mathsf{c}}, \mathsf{c}) \prec (\mathsf{r}, y) \prec (\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0) \prec (\mathsf{str} - y \cdot \mathsf{pk} = 0),$$

where r corresponds to c. Apparently, if

$$(\mathsf{str} - y \cdot \mathsf{pk} = 0) \prec (\mathsf{pk} \| \tilde{\mathsf{c}} \| \mathsf{str}, \mathsf{k}) \in T_{\mathcal{H}_{\mathsf{key}}},$$

the simulator guarantees that $(*, \mathsf{k}) = \mathbf{IES\text{-}KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$. Conversely, if

$$(\mathsf{pk} \| \tilde{\mathsf{c}} \| \mathsf{str}, \mathsf{k}) \prec \mathsf{Eq} := (\mathsf{str} - y \cdot \mathsf{pk} = 0).$$

There are two possibilities for how Eq is induced:

(i) Eq *is induced by an independent equation.* Consider the case where both $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$ and $(\mathsf{pk} \| \tilde{\mathsf{c}} \| \mathsf{str}, \mathsf{k})$ are already present. In other words, Eq is a specific equation. If Eq is induced by an equation from a $\mathcal{P}^{-1}$ or $\mathcal{G}$ query, as discussed earlier, then the randomly sampled str must collide with a group element that already appears in T_{eq} or $T_{\mathcal{H}_{\mathsf{key}}}$, thereby triggering conditions $\mathsf{C}_{3\text{-}\mathcal{P}^{-1}}$ or $\mathsf{C}_{1\text{-}\mathcal{G}}$. If, instead, Eq is induced by an equation $(\widetilde{\mathsf{str}} - \tilde{x} \cdot \diamondsuit = 0)$ from a $\mathcal{H}_x$ or $\mathcal{H}_y$ query, there must exist an equation

$$(\mathsf{str} - y \cdot \mathsf{pk} - \alpha \cdot \widetilde{\mathsf{str}} - \beta \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$$

such that the uniformly sampled $\tilde{x}$ satisfies $\alpha \cdot \tilde{x} + \beta = 0$, thereby triggering conditions $\mathsf{C}_{4\text{-}\mathcal{H}_x}$ or $\mathsf{C}_{5\text{-}\mathcal{H}_y}$.

(ii) Eq *is induced by the insertion of* $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$. In this case, there must exist an equation

$$(\mathsf{str} - y \cdot \mathsf{pk} - \alpha \cdot \tilde{\mathsf{c}} - \beta \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}}).$$

Suppose that, at the time of inserting $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$, no discrete logarithm of pk has yet been fixed; that is, there is no equation of the form $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ in $\mathsf{Span}(T_{\mathsf{eq}})$. In this case, the uniformly sampled y must satisfy $\alpha \cdot y + \beta = 0$, thereby triggering condition $\mathsf{C}_{5\text{-}\mathcal{H}_y}$. Conversely, if a discrete logarithm x of pk has already been determined, then $(\mathsf{str} - x \cdot \tilde{\mathsf{c}} = 0)$ must also appear. By Proposition D.1, this implies the existence of either (sk, x) in $T_{\mathcal{H}_x}$ or $(\mathsf{sk}, \mathsf{pk})$ in T^*_{sk}. Furthermore, Propositions D.9 and D.10 together guarantee that in this case $(*, \mathsf{k}) = \mathbf{IES\text{-}KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$ must hold.

Type 3. If $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$ is induced by the insertion of Eq from $\mathcal{P}^{-1}$ or $\mathcal{G}$, then—provided that the uniformly sampled group element does not already appear in T_{eq} or $T_{\mathcal{H}_{\mathsf{key}}}$—we must have $\mathsf{Eq} = (\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$. In this case,

$$(\mathsf{r}, y) \prec (\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0) \prec (\mathsf{pk} \| \tilde{\mathsf{c}} \| \mathsf{str}, \mathsf{k}).$$

Apparently, if $(\mathsf{str} - y \cdot \mathsf{pk} = 0) \prec (\mathsf{pk} \| \tilde{\mathsf{c}} \| \mathsf{str}, \mathsf{k})$, the simulator ensures that $(*, \mathsf{k}) = \mathbf{IES\text{-}KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$. Conversely, if

$$(\mathsf{pk} \| \tilde{\mathsf{c}} \| \mathsf{str}, \mathsf{k}) \prec \mathsf{Eq} := (\mathsf{str} - y \cdot \mathsf{pk} = 0),$$

then Eq must be a specific equation induced by an independent relation, in which case one of the failure conditions $\mathsf{C}_{4\text{-}\mathcal{H}_x}$, $\mathsf{C}_{5\text{-}\mathcal{H}_y}$, $\mathsf{C}_{3\text{-}\mathcal{P}^{-1}}$, or $\mathsf{C}_{1\text{-}\mathcal{G}}$ is necessarily triggered.

Thus, by considering all cases above, we conclude that, as long as none of the failure conditions occur, the simulator assigns $(*, \mathsf{k}) = \mathbf{IES\text{-}KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$ as stated in the proposition. □ □

Proposition D.9. *Suppose that none of the failure conditions hold. Then for any tuple* (sk, x) *in* $T_{\mathcal{H}_x}$*, equations* $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ *and* $(\mathsf{str} - x \cdot \tilde{\mathsf{c}} = 0)$ *in the span of* T_{eq}*, tuple* $(\tilde{\mathsf{c}}, \mathsf{c})$ *in* $T_{\mathcal{P}}$*, and tuple* $(\mathsf{pk} \| \tilde{\mathsf{c}} \| \mathsf{str}, \mathsf{k})$ *in* $T_{\mathcal{H}_{\mathsf{key}}}$*, provided that no pair* $(\mathsf{r}, \tilde{\mathsf{c}})$ *exists in* T^*_{r}*, it holds that*

$$\mathsf{k} = \mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c}).$$

Proof. By Proposition D.5, the simulator records (sk, x) in $T_{\mathcal{H}_x}$ along with $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ in T_{eq}, where $\mathsf{pk} = \mathbf{IES\text{-}KEM}.\mathsf{KG}(\mathsf{sk})$ and x is sampled uniformly at random. If no equation of the form $(\mathsf{str}_1 - x \cdot \mathsf{str}_2 = 0)$ lies in $\mathsf{Span}(T_{\mathsf{eq}})$ (i.e., condition $\mathsf{C}_{3\text{-}\mathcal{H}_x}$ is not triggered), the order is

$$(\mathsf{sk}, x) \prec (\mathsf{pk} - x \cdot \diamondsuit = 0) \prec (\mathsf{str} - x \cdot \tilde{\mathsf{c}} = 0).$$

In particular, if $(\mathsf{str} - x \cdot \tilde{\mathsf{c}} = 0), (\tilde{\mathsf{c}}, \mathsf{c}) \prec (\mathsf{pk} \| \tilde{\mathsf{c}} \| \mathsf{str}, \mathsf{k})$, then the simulator ensures that $\mathsf{k} = \mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c})$. Otherwise, if

$$(\mathsf{str} - x \cdot \tilde{\mathsf{c}} = 0) \prec (\mathsf{pk} \| \tilde{\mathsf{c}} \| \mathsf{str}, \mathsf{k}) \prec (\tilde{\mathsf{c}}, \mathsf{c}),$$

the simulator inserts $(\mathsf{r}, \tilde{\mathsf{c}})$ into T^*_{r} while processing the $\mathcal{H}_{\mathsf{key}}$ query, and the subsequent insertion of $(\tilde{\mathsf{c}}, \mathsf{c})$ ensures that

$$(\mathsf{c}, \mathsf{k}) = \mathbf{IES\text{-}KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r}).$$

Conversely, if $(\mathsf{pk} \| \tilde{\mathsf{c}} \| \mathsf{str}, \mathsf{k}) \prec (\mathsf{str} - x \cdot \tilde{\mathsf{c}} = 0)$, there are two types to consider:

- Type 1: $(\mathsf{str} - x \cdot \tilde{\mathsf{c}} = 0)$ is induced by the insertion of independent equation.
- Type 2: $(\mathsf{str} - x \cdot \tilde{\mathsf{c}} = 0)$ is induced by the insertion of $(\mathsf{pk} - x \cdot \diamondsuit = 0)$.

Type 1. In this case, $(\mathsf{str} - x \cdot \tilde{\mathsf{c}} = 0)$ is a specific equation induced by an independent query, causing one of the conditions $\mathsf{C}_{4\text{-}\mathcal{H}_x}$, $\mathsf{C}_{5\text{-}\mathcal{H}_y}$, $\mathsf{C}_{3\text{-}\mathcal{P}^{-1}}$, or $\mathsf{C}_{1\text{-}\mathcal{G}}$ to hold.

Type 2. There must exist an equation

$$(\mathsf{str} - x \cdot \tilde{\mathsf{c}} - \alpha \cdot \mathsf{pk} - \beta \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}}).$$

Suppose that, when inserting $(\mathsf{pk} - x \cdot \diamondsuit = 0)$, there is no $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$ in $\mathsf{Span}(T_{\mathsf{eq}})$. In this case, the random discrete logarithm x must satisfy $\alpha \cdot x + \beta = 0$, thereby triggering condition $\mathsf{C}_{4\text{-}\mathcal{H}_x}$.

On the other hand, if a discrete logarithm y of $\tilde{\mathsf{c}}$ already exists, then $(\mathsf{str} - y \cdot \mathsf{pk} = 0)$ must also appear in $\mathsf{Span}(T_{\mathsf{eq}})$. Provided that no $(\mathsf{r}, \tilde{\mathsf{c}})$ exists in T^*_{r}, there must exist (r, y) in $T_{\mathcal{H}_y}$ when inserting $(\tilde{\mathsf{c}}, \mathsf{c})$ into $T_{\mathcal{P}}$. Thus, there is a point at which (r, y), $(\tilde{\mathsf{c}} - t \cdot \diamondsuit = 0)$, $(\mathsf{str} - y \cdot \mathsf{pk} = 0)$, and $(\mathsf{pk} \| \tilde{\mathsf{c}} \| \mathsf{str}, \mathsf{k})$ are present but $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ is absent. By Proposition D.8, it follows that

$$\mathsf{k} = \mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c}).$$

Thus, by considering all cases above, we conclude that, as long as none of the failure conditions occur, the simulator assigns $\mathsf{k} = \mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c})$ as stated in the proposition. □

□

Proposition D.10. *Suppose that none of the failure conditions hold. Then for any tuple* $(\mathsf{sk}, \mathsf{pk})$ *in* T^*_{sk}*, equations* $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ *and* $(\mathsf{str} - x \cdot \tilde{\mathsf{c}} = 0)$ *in the span of* T_{eq}*, tuple* $(\tilde{\mathsf{c}}, \mathsf{c})$ *in* $T_{\mathcal{P}}$*, and tuple* $(\mathsf{pk} \| \tilde{\mathsf{c}} \| \mathsf{str}, \mathsf{k})$ *in* $T_{\mathcal{H}_{\mathsf{key}}}$*, provided that no pair* $(\mathsf{r}, \tilde{\mathsf{c}})$ *exists in* T^*_{r}*, it holds that*

$$\mathsf{k} = \mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c}).$$

Proof. By Proposition D.1, when $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ lies in $\mathsf{Span}(T_{\mathsf{eq}})$, there already exists $(\mathsf{sk}, \mathsf{pk})$ in T^*_{sk}. There are two types to consider:

- Type 1: $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ is induced by equation from $\mathcal{H}_x$ or $\mathcal{H}_y$.
- Type 2: $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ is inserted by $\mathcal{P}^{-1}$ or $\mathcal{G}$.

Type 1. If $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ is induced by an equation Eq from $\mathcal{H}_x$, and no sk appears simultaneously in both $T_{\mathcal{H}_x}$ and T^*_{sk}, then Eq must be of the form $(\widetilde{\mathsf{pk}} - \tilde{x} \cdot \diamondsuit = 0)$. In this case, there exists

$$(\mathsf{pk} - \alpha \cdot \widetilde{\mathsf{pk}} - \beta \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$$

such that $x = \alpha \cdot \tilde{x} + \beta$. Here, x can be regarded as uniformly sampled from $\mathbb{Z}_N$.

If no equation of the form $(\mathsf{str}_1 - x \cdot \mathsf{str}_2 = 0)$ exists in $\mathsf{Span}(T_{\mathsf{eq}})$ (i.e., condition $\mathsf{C}_{6\text{-}\mathcal{H}_x}$ is not triggered), then $(\mathsf{str} - x \cdot \tilde{\mathsf{c}} = 0)$ must be inserted after $(\mathsf{pk} - x \cdot \diamondsuit = 0)$. The same argument applies when $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ itself is induced by an equation from $\mathcal{H}_y$. Hence, the recording order is

$$(\mathsf{sk}, \mathsf{pk}) \prec (\mathsf{pk} - x \cdot \diamondsuit = 0) \prec (\mathsf{str} - x \cdot \tilde{\mathsf{c}} = 0).$$

Suppose $(\mathsf{str} - x \cdot \tilde{\mathsf{c}} = 0)$ is recorded prior to $(\mathsf{pk} \| \tilde{\mathsf{c}} \| \mathsf{str}, \mathsf{k})$. If, at the time of inserting $(\mathsf{pk} \| \tilde{\mathsf{c}} \| \mathsf{str}, \mathsf{k})$, the equations $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$ and $(\mathsf{str} - y \cdot \mathsf{pk} = 0)$ are already present, then—since $(\mathsf{r}, \tilde{\mathsf{c}})$ does not exist in T^*_{r}—there must also exist (r, α) and Proposition D.8 guarantees that $\mathsf{k} = \mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c})$. Otherwise, provided that no sk appears simultaneously in both $T_{\mathcal{H}_x}$ and T^*_{sk}, the tuple $(\tilde{\mathsf{c}}, \mathsf{c})$ must already exist when querying for k, and the simulator ensures that $\mathsf{k} = \mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c})$.

We now turn to the case where $(\mathsf{pk} \| \tilde{\mathsf{c}} \| \mathsf{str}, \mathsf{k}) \prec \mathsf{Eq} := (\mathsf{str} - x \cdot \tilde{\mathsf{c}} = 0)$. There are two possibilities for how Eq is induced:

(i) Eq *is induced by an independent equation.* Apparently, $(\mathsf{str} - x \cdot \tilde{\mathsf{c}} = 0)$ is a specific equation that induced by the independent query, leading to conditions $\mathsf{C}_{4\text{-}\mathcal{H}_x}$, $\mathsf{C}_{5\text{-}\mathcal{H}_y}$, $\mathsf{C}_{3\text{-}\mathcal{P}^{-1}}$, or $\mathsf{C}_{1\text{-}\mathcal{G}}$ hold.

(ii) Eq *is induced by the insertion of* $(\mathsf{pk} - x \cdot \diamondsuit = 0)$. There must exist an equation

$$(\mathsf{str} - x \cdot \tilde{\mathsf{c}} - \alpha \cdot \mathsf{pk} - \beta \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}}).$$

Suppose that, when inserting $(\mathsf{pk} - x \cdot \diamondsuit = 0)$, there is no $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$ in $\mathsf{Span}(T_{\mathsf{eq}})$. In this case, the random discrete logarithm x must satisfy $\alpha \cdot x + \beta = 0$, thereby triggering condition $\mathsf{C}_{8\text{-}\mathcal{H}_x}$ or $\mathsf{C}_{9\text{-}\mathcal{H}_y}$. On the other hand, if a discrete logarithm y of $\tilde{\mathsf{c}}$ already exists, then $(\mathsf{str} - y \cdot \mathsf{pk} = 0)$ must also appear in $\mathsf{Span}(T_{\mathsf{eq}})$. As noted previously, by Proposition D.8, it follows that $\mathsf{k} = \mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c})$.

Type 2. If $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ is induced by an equation Eq from $\mathcal{P}^{-1}$ or $\mathcal{G}$, then—provided that the uniformly sampled group element does not already appear in T_{eq} or $T_{\mathcal{H}_{\mathsf{key}}}$—we must have $\mathsf{Eq} = (\mathsf{pk} - x \cdot \diamondsuit = 0)$. In this case,

$$(\mathsf{sk}, \mathsf{pk}) \prec (\mathsf{pk} - x \cdot \diamondsuit = 0) \prec (\mathsf{pk} \| \tilde{\mathsf{c}} \| \mathsf{str}, \mathsf{k}).$$

Note that if the equation $(\mathsf{str} - x \cdot \tilde{\mathsf{c}} = 0)$ is induced after $(\mathsf{pk} \| \tilde{\mathsf{c}} \| \mathsf{str}, \mathsf{k})$, then it must be a specific equation derived from an independent query, thereby triggering one of the conditions $\mathsf{C}_{4\text{-}\mathcal{H}_x}$, $\mathsf{C}_{5\text{-}\mathcal{H}_y}$, $\mathsf{C}_{3\text{-}\mathcal{P}^{-1}}$, or $\mathsf{C}_{1\text{-}\mathcal{G}}$.

Suppose instead that $(\mathsf{str} - x \cdot \tilde{\mathsf{c}} = 0) \prec (\mathsf{pk} \| \tilde{\mathsf{c}} \| \mathsf{str}, \mathsf{k})$. If, at the time of inserting $(\mathsf{pk} \| \tilde{\mathsf{c}} \| \mathsf{str}, \mathsf{k})$, the equations $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$ and $(\mathsf{str} - y \cdot \mathsf{pk} = 0)$ are already present, then—since $(\mathsf{r}, \tilde{\mathsf{c}})$ does not exist in T^*_{r}—there must also exist (r, α), and Proposition D.8 guarantees that $\mathsf{k} = \mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c})$. Otherwise, provided that no sk appears simultaneously in both $T_{\mathcal{H}_x}$ and T^*_{sk}, the tuple $(\tilde{\mathsf{c}}, \mathsf{c})$ must already exist when querying for k, and the simulator ensures that $\mathsf{k} = \mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c})$.

Thus, by considering all cases above, we conclude that, as long as none of the failure conditions occur, the simulator assigns $\mathsf{k} = \mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c})$ as stated in the proposition. □ □

Proposition D.11. *Suppose that none of the failure conditions hold. Then for any equations* $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$ *and* $(\mathsf{str} - y \cdot \mathsf{pk} = 0)$ *in the span of* T_{eq}*, tuple* $(\tilde{\mathsf{c}}, \mathsf{c})$ *in* $T_{\mathcal{P}}$*, and tuple* $(\mathsf{pk} \| \tilde{\mathsf{c}} \| \mathsf{str}, \mathsf{k})$ *in* $T_{\mathcal{H}_{\mathsf{key}}}$*, provided that no pair* $(\mathsf{sk}, \mathsf{pk})$ *exists in* T^*_{sk}*, it holds that*

$$\mathsf{k} = \mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c}).$$

Proof. The equation $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$ may result from the insertion of an equation generated by $\mathcal{H}_x$, $\mathcal{H}_y$, $\mathcal{P}^{-1}$, or $\mathcal{G}$. In the following, we examine each of these possibilities separately.

Type 1. If $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$ is induced by inserting $\mathsf{Eq} := (\widetilde{\mathsf{pk}} - \tilde{x} \cdot \diamondsuit = 0)$ from $\mathcal{H}_x$, then there must exist

$$(\tilde{\mathsf{c}} - \alpha \cdot \widetilde{\mathsf{pk}} - \beta \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$$

such that $y = \alpha \cdot \tilde{x} + \beta$. If no equation of the form $(\mathsf{str}_1 - y \cdot \mathsf{str}_2 = 0)$ in $\mathsf{Span}(T_{\mathsf{eq}})$, then necessarily

$$(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0) \prec (\mathsf{str} - y \cdot \mathsf{pk} = 0).$$

Suppose the equation $(\mathsf{str} - y \cdot \mathsf{pk} = 0)$ is induced after $(\mathsf{pk} \| \tilde{\mathsf{c}} \| \mathsf{str}, \mathsf{k})$. Provided that no failure condition is triggered, there must already exist $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ and $(\mathsf{str} - x \cdot \tilde{\mathsf{c}} = 0)$. By Proposition D.1, there exists a pair (sk, x), and Proposition D.12 then ensures that $\mathsf{k} = \mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c})$.

Alternatively, suppose that

$$(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0) \prec (\mathsf{str} - y \cdot \mathsf{pk} = 0) \prec (\mathsf{pk} \| \tilde{\mathsf{c}} \| \mathsf{str}, \mathsf{k}).$$

If no failure condition is triggered, then there must exist either (r, y) in $T_{\mathcal{H}_y}$ or $(\mathsf{r}, \tilde{\mathsf{c}})$ in T^*_{r}, such that both c from $(\tilde{\mathsf{c}}, \mathsf{c})$ and k from $(\mathsf{pk} \| \tilde{\mathsf{c}} \| \mathsf{str}, \mathsf{k})$ correspond to the same r. Therefore, it follows that $\mathsf{k} = \mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c})$.

Type 2. If $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$ is induced by inserting Eq from $\mathcal{H}_y$, then the analysis proceeds in the same way as Type 1.

Type 3. If $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$ is induced by inserting Eq from $\mathcal{P}^{-1}$ or $\mathcal{G}$, provided that the uniformly sampled group element does not already appear in T_{eq} or $T_{\mathcal{H}_{\mathsf{key}}}$, we have $\mathsf{Eq} = (\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$. In this case,

$$(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0) \prec (\mathsf{pk} \| \tilde{\mathsf{c}} \| \mathsf{str}, \mathsf{k}).$$

If the equation $(\mathsf{str} - y \cdot \mathsf{pk} = 0)$ is induced after $(\mathsf{pk} \| \tilde{\mathsf{c}} \| \mathsf{str}, \mathsf{k})$, then it corresponds to a specific equation already analyzed earlier. Otherwise, if

$$(\mathsf{str} - y \cdot \mathsf{pk} = 0) \prec (\mathsf{pk} \| \tilde{\mathsf{c}} \| \mathsf{str}, \mathsf{k}),$$

then, by the same reasoning as in Type 1, there exists r corresponding to both c and k, which implies that $\mathsf{k} = \mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c})$.

Thus, by considering all cases above, we conclude that, as long as none of the failure conditions occur, the simulator assigns $\mathsf{k} = \mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c})$ as stated in the proposition. □ □

Proposition D.12. *Suppose that none of the failure conditions hold. Then for any tuple* (sk, x) *in* $T_{\mathcal{H}_x}$*, equations* $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ *and* $(\mathsf{str} - x \cdot \tilde{\mathsf{c}} = 0)$ *in the span of* T_{eq}*, tuple* $(\tilde{\mathsf{c}}, \mathsf{c})$ *in* $T_{\mathcal{P}}$*, and tuple* $(\mathsf{pk} \| \tilde{\mathsf{c}} \| \mathsf{str}, \mathsf{k})$ *in* $T_{\mathcal{H}_{\mathsf{key}}}$*, it holds that*

$$\mathsf{k} = \mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c}).$$

Proof. By Proposition D.5, the simulator records (sk, x) in $T_{\mathcal{H}_x}$ together with $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ in T_{eq}, where $\mathsf{pk} = \mathbf{IES\text{-}KEM}.\mathsf{KG}(\mathsf{sk})$ and x is sampled uniformly at random. If no equation of the form $(\mathsf{str}_1 - x \cdot \mathsf{str}_2 = 0)$ exists in $\mathsf{Span}(T_{\mathsf{eq}})$, the order is

$$(\mathsf{sk}, x) \prec (\mathsf{pk} - x \cdot \diamondsuit = 0) \prec (\mathsf{str} - x \cdot \tilde{\mathsf{c}} = 0).$$

Note that if $(\mathsf{str} - x \cdot \tilde{\mathsf{c}} = 0) \prec (\mathsf{pk} | \tilde{\mathsf{c}} | \mathsf{str}, \mathsf{k})$, then there exists some r corresponding to both c and k, which implies $\mathsf{k} = \mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c})$. Conversely, if $(\mathsf{pk} \| \tilde{\mathsf{c}} \| \mathsf{str}, \mathsf{k}) \prec (\mathsf{str} - x \cdot \tilde{\mathsf{c}} = 0)$, there are two types to consider:

- Type 1: $(\mathsf{str} - x \cdot \tilde{\mathsf{c}} = 0)$ is induced by the insertion of independent equation.
- Type 2: $(\mathsf{str} - x \cdot \tilde{\mathsf{c}} = 0)$ is induced by the insertion of $(\mathsf{pk} - x \cdot \diamondsuit = 0)$.

Type 1. Here, $(\mathsf{str} - x \cdot \tilde{\mathsf{c}} = 0)$ is a fixed equation induced by an independent query, thereby triggering one of the conditions $\mathsf{C}_{4\text{-}\mathcal{H}_x}$, $\mathsf{C}_{5\text{-}\mathcal{H}_y}$, $\mathsf{C}_{3\text{-}\mathcal{P}^{-1}}$, or $\mathsf{C}_{1\text{-}\mathcal{G}}$.

Type 2. In this case, there must exist an equation

$$(\mathsf{str} - x \cdot \tilde{\mathsf{c}} - \alpha \cdot \mathsf{pk} - \beta \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}}).$$

If, at the time of inserting $(\mathsf{pk} - x \cdot \diamondsuit = 0)$, no equation of the form $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$ exists in $\mathsf{Span}(T_{\mathsf{eq}})$, then the random discrete logarithm x must satisfy $\alpha \cdot x + \beta = 0$, which triggers condition $\mathsf{C}_{4\text{-}\mathcal{H}_x}$.

Otherwise, if a discrete logarithm y of $\tilde{\mathsf{c}}$ already exists, then $(\mathsf{str} - y \cdot \mathsf{pk} = 0)$ must also appear in $\mathsf{Span}(T_{\mathsf{eq}})$. Provided that no sk occurs simultaneously in both $T_{\mathcal{H}_x}$ and T^*_{sk}, Proposition D.11 ensures that $\mathsf{k} = \mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c})$.

Thus, by considering all cases above, we conclude that, as long as none of the failure conditions occur, the simulator assigns $\mathsf{k} = \mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c})$ as stated in the proposition. □ □

D.4 Transition from $\mathbf{G}_2$ to $\mathbf{G}_3$

Game 3. In this game, the simulator $\mathcal{S}_{\mathsf{fail}}$ remains unchanged. Instead, the relay algorithm $\mathcal{F} = (\mathcal{F}.\mathsf{KG}, \mathcal{F}.\mathsf{Encaps}, \mathcal{F}.\mathsf{Decaps})$ is replaced with the construction $\Pi_{\mathsf{IES}} = (\Pi_{\mathsf{IES}}.\mathsf{KG}, \Pi_{\mathsf{IES}}.\mathsf{Encaps}, \Pi_{\mathsf{IES}}.\mathsf{Decaps})$. Notably, Π_{IES} does not query the ideal KEM directly; rather, it interacts with $\mathcal{S}_{\mathsf{fail}}$, thereby ensuring that the honest interfaces are consistent with the adversarial interfaces. Consequently, the differentiator can distinguish between Game 2 and Game 3 only if it detects an inconsistency between $\mathcal{F}$ and $\mathcal{S}_{\mathsf{fail}}$ in Game 2.

Let $\mathbf{G}_3$ denote Game 3, which outputs 1 if and only if $\mathcal{D}$ outputs 1 after interacting with Π_{IES} and $\mathcal{S}_{\mathsf{fail}}$. Formally,

$$\Pr[\mathbf{G}_3(\mathcal{D}) = 1] = \Pr[\mathcal{D}^{\Pi_{\mathsf{IES}}^{\mathcal{S}_{\mathsf{fail}}}, \mathcal{S}_{\mathsf{fail}}^{\mathbf{IES\text{-}KEM}}}(1^\lambda) = 1].$$

Lemma D.2. *Let* q *denote the number of queries made by the differentiator to both the honest and adversarial interfaces. Then,*

$$|\Pr[\mathbf{G}_3(\mathcal{D}) = 1] - \Pr[\mathbf{G}_2(\mathcal{D}) = 1]| \leq O\left(\frac{(\lambda \cdot q)^4}{N} + \frac{(\lambda \cdot q)^2}{2^\lambda}\right).$$

Proof. To prove that the views in Game 2 and Game 3 are statistically close, we argue that the differentiator can only exploit an inconsistency if the simulator $\mathcal{S}_{\mathsf{fail}}$ fails in at least one of the two games.

For Game 3, recall that the honest interfaces produce their responses by querying the adversarial interfaces, namely the simulator $\mathcal{S}_{\mathsf{fail}}$. Hence, as long as the simulator provides consistent answers to identical queries across all invocations, the responses of the honest interface necessarily coincide with those of the

adversarial interface. For random oracle queries, this consistency holds trivially. Propositions D.3 and D.4 guarantee that a permutation query cannot produce more than one distinct response, while Proposition D.2 ensures the consistency of T_{eq}, i.e., that the responses to $\mathcal{G}$ queries are proper.

For Game 2, any inconsistency between the honest interface $\mathcal{F}$ and the adversarial interface $\mathcal{S}_{\mathsf{fail}}$ can occur only when the response generated by $\mathcal{F}$ deviates from that of the simulator implementing the Π_{IES} construction. We now establish that, provided $\mathcal{S}_{\mathsf{fail}}$ does not fail, such an inconsistency cannot arise.

Inconsistency with respect to $\mathcal{F}$.KG. For any public-key generation query on input sk, the relay algorithm $\mathcal{F}$ always returns $\mathsf{pk} \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{KG}(\mathsf{sk})$. Thus, an inconsistency w.r.t. $\mathcal{F}$.KG arises if and only if $\mathcal{S}_{\mathsf{fail}}$ responds with some str such that $\mathsf{str} \neq \mathsf{pk}$.

Concretely, when the differentiator generates the public key for sk through $\mathcal{S}_{\mathsf{fail}}$ according to the Π_{IES} construction, it queries $\mathcal{H}_x(\mathsf{sk})$ to obtain x, and then queries for the label str of x. This ensures that (sk, x) in $T_{\mathcal{H}_x}$ and $(\mathsf{str} - x \cdot \diamondsuit = 0)$ in the span of T_{eq}. By Proposition D.5, we deduce that

$$\mathsf{str} = \mathbf{IES\text{-}KEM}.\mathsf{KG}(\mathsf{sk})$$

unless the simulator $\mathcal{S}_{\mathsf{fail}}$ fails.

Inconsistency with respect to $\mathcal{F}$.Encaps. For any encapsulation query on input pk and r, the relay algorithm $\mathcal{F}$ always returns $\mathbf{IES\text{-}KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$. Consistency therefore requires that the following two hold:

1. If pk is invalid with respect to the ideal KEM, then $\mathcal{S}_{\mathsf{fail}}.\mathcal{G}^{\mathsf{add}}$ on input pk must return $\perp$.
2. Otherwise, let $(\mathsf{c}, \mathsf{k}) \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$, the simulator $\mathcal{S}_{\mathsf{fail}}$ must return the same ciphertext c and key k.

Based on the description of the simulator, when processing an addition query with inputs $(\mathsf{str}_1, \mathsf{str}_2)$, the simulator determines their validity by querying $\mathbf{IES\text{-}KEM}.\mathsf{Encaps}$. Therefore, for any public key pk that is invalid with respect to the ideal KEM, we must have

$$\mathcal{S}_{\mathsf{fail}}.\mathcal{G}^{\mathsf{add}}(\mathsf{pk}, \mathsf{pk}) = \perp.$$

By the definition of $\mathbf{IES\text{-}KEM}$, the ciphertext c output by Encaps is determined solely by the input randomness r. For any given r, the differentiator derives the corresponding ciphertext through the following procedure:

- Query $\mathcal{P}(\tilde{\mathsf{c}})$ with $(\mathsf{r}, y) \in T_{\mathcal{H}_y}$ such that $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$ lies in the span of T_{eq}.

In this case, Proposition D.6 ensures that whenever $(\mathsf{r}, y) \in T_{\mathcal{H}_y}$, $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$, and $(\tilde{\mathsf{c}}, \mathsf{c}) \in T_{\mathcal{P}}$, it follows that

$$(\mathsf{c}, *) = \mathbf{IES\text{-}KEM}.\mathsf{Encaps}(*, \mathsf{r}).$$

Furthermore, the differentiator may also query $\tilde{\mathsf{c}} \leftarrow \mathcal{P}^{-1}(\mathsf{c})$ with $(\mathsf{r}, y) \in T_{\mathcal{H}_y}$. In this case, it must hold that $\mathcal{G}^{\mathsf{label}}(y) = \tilde{\mathsf{c}}$, and Proposition D.7 guarantees this consistency.

Now consider any private key $\mathsf{sk} \in \mathbb{Z}_N$, with $\mathsf{pk} \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{KG}(\mathsf{sk})$, and any randomness r. When the differentiator derives the key k corresponding to $(\mathsf{pk}, \mathsf{r})$, it must query $\mathcal{H}_{\mathsf{key}}$ with $(\mathsf{pk}, \tilde{\mathsf{c}}, \mathsf{str})$, where $\tilde{\mathsf{c}}$ corresponds to r and the discrete logarithm of str equals the product of the discrete logarithms of pk and $\tilde{\mathsf{c}}$. Specifically, the relationship between $(\mathsf{pk}, \mathsf{r})$ and $(\mathsf{pk}, \tilde{\mathsf{c}}, \mathsf{str})$ must fall into one of the following associations:

- Form 1: There exists $(\mathsf{r}, y) \in T_{\mathcal{H}_y}$, and both $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$ and $(\mathsf{str} - y \cdot \mathsf{pk} = 0)$ lie in the span of T_{eq}.
- Form 2: There exists $(\tilde{\mathsf{c}}, \mathsf{c}) \in T_{\mathcal{P}}$ such that c corresponds to r, and both $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ and $(\mathsf{str} - x \cdot \tilde{\mathsf{c}} = 0)$ lie in $\mathsf{Span}(T_{\mathsf{eq}})$.

Let $(\mathsf{c}, \mathsf{k}) \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$. For Form 1, by Proposition D.8, it follows that

$$\mathcal{H}_{\mathsf{key}}(\mathsf{pk}, \tilde{\mathsf{c}}, \mathsf{str}) = \mathsf{k}.$$

For Form 2, by Proposition D.1, ther presence of $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ in the span of T_{eq} implies that either $(\mathsf{sk}, x) \in T_{\mathcal{H}_x}$ or $(\mathsf{sk}, \mathsf{pk}) \in T^*_{\mathsf{sk}}$. In either case, Proposition D.9 and Proposition D.10 guarantee that

$$\mathcal{H}_{\mathsf{key}}(\mathsf{pk}, \tilde{\mathsf{c}}, \mathsf{str}) = \mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c}) = \mathsf{k}.$$

*Inconsistency with respect to $\mathcal{F}$.*Decaps. For any decapsulation query with input sk and c, the relay algorithm $\mathcal{F}$ always returns $\mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c})$. Consistency therefore requires that the following two hold:

1. If c is invalid with respect to the ideal KEM, and there exists $(\tilde{\mathsf{c}}, \mathsf{c})$ in $T_{\mathcal{P}}$, then $\mathcal{S}_{\mathsf{fail}}.\mathcal{G}^{\mathsf{add}}$ on input $\tilde{\mathsf{c}}$ must return $\perp$.
2. Otherwise, Otherwise, letting $\mathsf{k} \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c})$, the simulator $\mathcal{S}_{\mathsf{fail}}$ must return the same k.

If there exists $(\tilde{\mathsf{c}}, \mathsf{c}) \in T_{\mathcal{P}}$ with $\tilde{\mathsf{c}}$ valid and c invalid, then it must be inserted by $\mathcal{P}^{-1}$, and condition $\mathsf{C}_{\text{2-}\mathcal{P}^{-1}}$ is necessarily triggered. Otherwise, if $\tilde{\mathsf{c}}$ is also invalid, then, as noted earlier,

$$\mathcal{S}_{\mathsf{fail}}.\mathcal{G}^{\mathsf{add}}(\tilde{\mathsf{c}}, \tilde{\mathsf{c}}) = \perp.$$

Now consider any private key $\mathsf{sk} \in \mathbb{Z}_N$ and any valid ciphertext c. When the differentiator derives the key k corresponding to $(\mathsf{sk}, \mathsf{c})$, it must query $\mathcal{H}_{\mathsf{key}}$ with $(\mathsf{pk}, \tilde{\mathsf{c}}, \mathsf{str})$, where $\mathsf{pk} = \mathbf{IES\text{-}KEM}.\mathsf{KG}(\mathsf{sk})$, $\tilde{\mathsf{c}}$ corresponds to c, and the discrete logarithm of str equals the product of the discrete logarithms of pk and $\tilde{\mathsf{c}}$. Specifically, letting $\mathsf{pk} \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{KG}(\mathsf{sk})$, the relationship between $(\mathsf{sk}, \mathsf{c})$ and $(\mathsf{pk}, \tilde{\mathsf{c}}, \mathsf{str})$ falls into one of the following associations:

- Form 1: There exists $(\tilde{\mathsf{c}}, \mathsf{c}) \in T_{\mathcal{P}}$, and both $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$ and $(\mathsf{str} - y \cdot \mathsf{pk} = 0)$ lie in the span of T_{eq}.
- Form 2: There exists $(\mathsf{sk}, x) \in T_{\mathcal{H}_x}$, $(\tilde{\mathsf{c}}, \mathsf{c}) \in T_{\mathcal{P}}$, and $(\mathsf{str} - x \cdot \tilde{\mathsf{c}} = 0)$ lies in the span of T_{eq}.

For Form 1, by Proposition D.12, it follows that

$$\mathcal{H}_{\mathsf{key}}(\mathsf{pk}, \tilde{\mathsf{c}}, \mathsf{str}) = \mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c}).$$

For Form 2, by Proposition D.5, the simulator records (sk, x) in $T_{\mathcal{H}_x}$ together with $(\mathsf{pk} - x \cdot \diamondsuit = 0)$ in T_{eq}, and then Proposition D.12 guarantees that

$$\mathcal{H}_{\mathsf{key}}(\mathsf{pk}, \tilde{\mathsf{c}}, \mathsf{str}) = \mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c}).$$

Hence, as long as the simulator $\mathcal{S}_{\mathsf{fail}}$ does not fail in either Game 2 or Game 3, the differentiator's view is statistically indistinguishable between the two games. Let q_{hon} and q_{adv} denote the number of queries made to the honest and adversarial interfaces, with $q_{\mathrm{hon}} + q_{\mathrm{adv}} \leq q$. Furthermore, each query to the honest interface in Game 3 requires at most λ queries to the simulator. Then,

$$\begin{aligned} |\Pr[\mathbf{G}_3(\mathcal{D}) = 1] - \Pr[\mathbf{G}_2(\mathcal{D}) = 1]| &\leq 2 \cdot \Pr[\mathcal{S}_{\mathsf{fail}} \text{ fails}] \\ &\leq O\left(\frac{(\lambda \cdot q)^4}{N} + \frac{(\lambda \cdot q)^2}{2^\lambda}\right). \end{aligned}$$

□
□

D.5 Transition from G_3 to G_4

Game 4. In this game, we modify the simulator such that its responses no longer depend on the ideal KEM. Specifically, the new simulator $\mathcal{S}_{\mathsf{samp}}$ implements each interface using lazy sampling, and all failure conditions are removed from $\mathcal{S}_{\mathsf{samp}}$.

By $\mathsf{Coin} \xleftarrow{\$} \mathrm{Ber}(p)$ we denote that a single bit Coin is sampled from a Bernoulli distribution with success probability p (i.e., $\Pr[\mathsf{Coin} = 1] = p$). We write $\mathsf{Ele}(T_{\mathsf{eq}})$ for the set of all group elements appearing in T_{eq}.

Simulator $\mathcal{S}_{\mathsf{samp}}$

$\underline{\mathcal{S}_{\mathsf{samp}}.\mathcal{H}_x(\mathsf{sk})}$:

if $\exists(\mathsf{sk}, x) \in T_{\mathcal{H}_x}$, then **return** x;

$x \xleftarrow{\$} \mathbb{Z}_N$; $T_{\mathcal{H}_x} \leftarrow T_{\mathcal{H}_x} \cup \{(\mathsf{sk}, x)\}$;

return x;

$\underline{\mathcal{S}_{\mathsf{samp}}.\mathcal{H}_y(\mathsf{r})}$:

if $\exists(\mathsf{r}, y) \in T_{\mathcal{H}_y}$, then **return** y;

$y \xleftarrow{\$} \mathbb{Z}_N$; $T_{\mathcal{H}_y} \leftarrow T_{\mathcal{H}_y} \cup \{(\mathsf{r}, y)\}$;

return y;

$\underline{\mathcal{S}_{\mathsf{samp}}.\mathcal{H}_{\mathsf{key}}(\mathsf{pk}, \tilde{\mathsf{c}}, \mathsf{str})}$:

if $\exists(\mathsf{pk}\|\tilde{\mathsf{c}}\|\mathsf{str}, \mathsf{k}) \in T_{\mathcal{H}_{\mathsf{key}}}$, then **return** k;

$\mathsf{k} \xleftarrow{\$} \mathcal{K}$; $T_{\mathcal{H}_{\mathsf{key}}} \leftarrow T_{\mathcal{H}_{\mathsf{key}}} \cup \{(\mathsf{pk}\|\tilde{\mathsf{c}}\|\mathsf{str}, \mathsf{k})\}$;

return k;

$\underline{\mathcal{S}_{\mathsf{samp}}.\mathcal{P}(\tilde{\mathsf{c}})}$:

if $\exists(\tilde{\mathsf{c}}, \mathsf{c}) \in T_{\mathcal{P}}$, then **return** c;

$\mathsf{c} \xleftarrow{\$} \{0,1\}^{\ell}$; $T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\tilde{\mathsf{c}}, \mathsf{c})\}$;

return c;

$\underline{\mathcal{S}_{\mathsf{samp}}.\mathcal{P}^{-1}(\mathsf{c})}$:

if $\exists(\tilde{\mathsf{c}}, \mathsf{c}) \in T_{\mathcal{P}}$, then **return** $\tilde{\mathsf{c}}$;

$\tilde{\mathsf{c}} \xleftarrow{\$} \{0,1\}^{\ell}$; $T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\tilde{\mathsf{c}}, \mathsf{c})\}$;

return $\tilde{\mathsf{c}}$.

$\underline{\mathcal{S}_{\mathsf{samp}}.\mathcal{G}^{\mathsf{label}}(x)}$:

if $\exists(\mathsf{str} - x \cdot \diamondsuit = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$, then **return** str;

$\mathsf{str} \xleftarrow{\$} \{0,1\}^\ell$; $T_{\mathsf{eq}} \leftarrow T_{\mathsf{eq}} \cup \{\mathsf{str} - x \cdot \diamondsuit = 0\}$;

return str;

$\underline{\mathcal{S}_{\mathsf{samp}}.\mathcal{G}^{\mathsf{add}}(\mathsf{str}_1, \mathsf{str}_2)}$:

if $\mathsf{str}_1 \in T_\perp \vee \mathsf{str}_2 \in T_\perp$: return $\perp$;

if $\exists(\mathsf{str} - \mathsf{str}_1 - \mathsf{str}_2 = 0) \in \mathsf{Span}(T_{\mathsf{eq}})$, then **return** str;

for str_i that does not appear in T_{eq} with $i \in \{1,2\}$:

$\mathsf{Coin} \leftarrow \mathrm{Ber}(\frac{N - |\mathsf{Ele}(T_{\mathsf{eq}})|}{2^\ell - |\mathsf{Ele}(T_{\mathsf{eq}})| - |T_\perp|})$;

if $\mathsf{Coin} = 0$: $T_\perp \leftarrow T_\perp \cup \{\mathsf{str}_i\}$; return $\perp$;

$\mathsf{str} \xleftarrow{\$} \{0,1\}^\ell$; $T_{\mathsf{eq}} \leftarrow T_{\mathsf{eq}} \cup \{\mathsf{str} - \mathsf{str}_1 - \mathsf{str}_2 = 0\}$;

return str.

We now show that the views of the differentiator in Game 3 and Game 4 are statistically close. Recall that the only difference between the two games lies in how the simulator answers queries. Thus, the proof reduces to analyzing, for each interface, whether the responses generated by $\mathcal{S}_{\mathsf{samp}}$ faithfully reproduce the distribution enforced by $\mathcal{S}_{\mathsf{fail}}$, and whether any statistical deviation can only occur through rare collision events.

Responses from the $\mathcal{H}_x$ and $\mathcal{H}_y$ interfaces. For both $\mathcal{H}_x$ and $\mathcal{H}_y$ queries, the responses generated by $\mathcal{S}_{\mathsf{fail}}.\mathcal{H}_x$ and $\mathcal{S}_{\mathsf{samp}}.\mathcal{H}_x$ are sampled uniformly from $\mathbb{Z}_N$. Hence, their output distributions are identical.

Response from the $\mathcal{H}_{\mathsf{key}}$ interface. For a $\mathcal{H}_{\mathsf{key}}$ query with input $(\mathsf{pk}, \tilde{\mathsf{c}}, \mathsf{str})$, assuming no failure condition is triggered, the only discrepancy arises when the inputs satisfy a valid linear relation. In this case, the simulator $\mathcal{S}_{\mathsf{samp}}$ produces the response k by sampling uniformly from $\mathcal{K}$, rather than deriving it through **IES-KEM**.Encaps or **IES-KEM**.Decaps. However, since the ideal KEM **IES-KEM** is itself a random mapping from a valid public key and randomness to a key, the resulting output distributions remain identical.

Response from the $\mathcal{P}$ interface. For a $\mathcal{P}$ query with input $\tilde{\mathsf{c}}$, assuming no failure condition is triggered, the only difference is that $\mathcal{S}_{\mathsf{samp}}$ generates the response c by sampling uniformly from $\{0,1\}^\ell$ rather than invoking **IES-KEM**.Encaps. Since the ideal KEM **IES-KEM** is itself a random injection from randomness r to c, the only bad event here is that the sample c collides with a recorded ciphertext in $T_{\mathcal{P}}$, which occurs with probability at most $q/2^\ell$.

Response from the $\mathcal{P}^{-1}$ interface. For a $\mathcal{P}^{-1}$ query with input c, assuming no failure condition is triggered, the only bad event is that the sampled value $\tilde{\mathsf{c}}$ collides with an existing entry in $T_{\mathcal{P}}$, which occurs with probability at most $q/2^\ell$.

Response from the $\mathcal{G}^{\mathsf{label}}$ interface. For a labeling query with input x, and assuming no failure condition is triggered, the only difference is that $\mathcal{S}_{\mathsf{samp}}$ generates the group element by sampling uniformly from $\{0,1\}^\ell$ rather than invoking **IES-KEM**.KG. The only bad event here is that the sampled str falls into $T_\perp$, the set recording all currently considered invalid group encodings. The probability of this event is at most $2q/2^\ell$.

Response from the $\mathcal{G}^{\mathsf{add}}$ interface. For an addition query with input $(\mathsf{str}_1, \mathsf{str}_2)$, the simulator $\mathcal{S}_{\mathsf{samp}}$ decides whether an unknown encoding is valid by sampling from a Bernoulli distribution with probability $\frac{N - |\mathsf{Ele}(T_{\mathsf{eq}})|}{2^\ell - |\mathsf{Ele}(T_{\mathsf{eq}})| - |T_\perp|}$ of being valid, rather than querying **IES-KEM**.Encaps. These two approaches yield identical distributions. Assuming no failure condition is triggered, the only bad event is that the sampled response str lies in $T_\perp$, which occurs with probability at most $2q/2^\ell$.

Therefore, if the simulator $\mathcal{S}_{\mathsf{fail}}$ does not fail in Game 3 and no response generated by $\mathcal{S}_{\mathsf{samp}}$ in Game 4 triggers any of the failure conditions defined for $\mathcal{S}_{\mathsf{fail}}$, the distinguisher's view remains statistically close

between the two games. Let $\mathbf{G}_4$ denote Game 4, which outputs 1 if and only if the differentiator outputs 1 after interacting with Π_{IES} and $\mathcal{S}_{\mathsf{samp}}$. Formally,

$$\Pr[\mathbf{G}_4(\mathcal{D}) = 1] = \Pr[\mathcal{D}^{\Pi_{\mathsf{IES}}^{\mathcal{S}_{\mathsf{samp}}}, \mathcal{S}_{\mathsf{samp}}}(1^\lambda) = 1].$$

Then,

$$\begin{aligned}|\Pr[\mathbf{G}_4(\mathcal{D}) = 1] - \Pr[\mathbf{G}_3(\mathcal{D}) = 1]| &\leq O\left(\frac{(\lambda \cdot q)^4}{N} + \frac{(\lambda \cdot q)^2}{2^\lambda}\right) + q \cdot \frac{6q}{2^\ell} \\ &= O\left(\frac{(\lambda \cdot q)^4}{N} + \frac{(\lambda \cdot q)^2}{2^\lambda}\right).\end{aligned}$$

D.6 Transition from $\mathbf{G}_4$ to Real

Real Game. By replacing the simulator $\mathcal{S}_{\mathsf{samp}}$ in Game 4 with the real idealized oracles

$$\mathcal{O} = (\mathcal{H}_x, \mathcal{H}_y, , \mathcal{H}_{\mathsf{key}}, \mathcal{P}, \mathcal{P}^{-1}, \mathcal{G}^{\mathsf{label}}, \mathcal{G}^{\mathsf{add}}),$$

we arrive at the final game of our proof, which we refer to as the real game. In this game, the differentiator $\mathcal{D}$ interacts with $\Pi_{\mathsf{IES}}^{\mathcal{O}}$ and $\mathcal{O}$. The real game outputs 1 if and only if $\mathcal{D}$ outputs 1. Formally,

$$\Pr[\mathbf{Real}^{\Pi_{\mathsf{IES}}^{\mathcal{O}}}(\mathcal{D}) = 1] = \Pr[\mathcal{D}^{\Pi_{\mathsf{IES}}^{\mathcal{O}}, \mathcal{O}}(1^\lambda) = 1].$$

We now demonstrate that the views of the differentiator in Game 4 and the Real Game are statistically close.

For the random oracle queries $\mathcal{H}_x$, $\mathcal{H}_y$, and $\mathcal{H}_{\mathsf{key}}$, the responses in Game 4 and Real Game are uniformly distributed. For the random permutation queries $\mathcal{P}$ and $\mathcal{P}^{-1}$, the differentiator may observe a difference if the uniformly sampled response by $\mathcal{S}_{\mathsf{samp}}$ collides with the existing one in $T_{\mathcal{P}}$, each with probability at most $q/2^\ell$. For labeling and addition queries, the simulator in Game 4 uses T_{eq} to emulate the algebraic structure of the GGM and determines the validity of group encodings via the Bernoulli distribution and $T_\perp$. Specifically, an unknown group encoding is deemed valid with probability

$$\frac{N - |\mathsf{Ele}(T_{\mathsf{eq}})|}{2^\ell - |\mathsf{Ele}(T_{\mathsf{eq}})| - |T_\perp|}.$$

Since $\mathsf{Ele}(T_{\mathsf{eq}})$ and $T_\perp$ contain all valid and invalid encodings known to the differentiator, the views in the two games are identical. For each equation in T_{eq}, two cases arise:

Case 1: The equation is of the form $(\mathsf{str} - x \cdot \diamondsuit = 0)$, where str is uniformly sampled. A distinguishing difference may occur if str collides with a group element in T_{eq} with known discrete logarithm or if $\mathsf{str} \in T_\perp$, each with probability at most $q/2^\ell$.

Case 2: The equation is of the form $(\mathsf{str} - \mathsf{str}_1 - \mathsf{str}_2 = 0)$, where str is uniformly sampled. A difference is observed if str appears in either T_{eq} or $T_\perp$, which occurs with probability at most $q/2^\ell$.

Therefore, we deduce that

$$\left|\Pr[\mathbf{Real}^{\Pi_{\mathsf{IES}}^{\mathcal{O}}}(\mathcal{D}) = 1] - \Pr[\mathbf{G}_4(\mathcal{D}) = 1]\right| \leq O\left(\frac{q^2}{2^\ell}\right).$$

Combining all the games above, we conclude that:

$$\left|\Pr[\mathbf{Real}^{\Pi_{\mathsf{IES}}^{\mathcal{O}}}(\mathcal{D}) = 1] - \Pr[\mathbf{Ideal}_{\mathcal{S}}^{\mathbf{IES\text{-}KEM}}(\mathcal{D}) = 1]\right| \leq O\left(\frac{(\lambda \cdot q)^4}{N} + \frac{(\lambda \cdot q)^2}{2^\lambda}\right).$$

□

D.7 Bounding the failure conditions

Note that the simulator $\mathcal{S}_{\mathsf{fail}}$ in Games 2 and 3 may fail under various conditions. In this section, we analyze the probability of each such failure occurring.

Lemma D.1. $\Pr[\text{Failure Conditions Hold}] \leq O\left(q^4/N + q^2/2^\lambda\right)$.

Proof. As described in Appendix D.2, the simulator evaluates potential failure conditions whenever it handles a $\mathcal{H}_x$, $\mathcal{H}_y$, $\mathcal{H}_{\mathsf{key}}$, $\mathcal{P}$, $\mathcal{P}^{-1}$, or $\mathcal{G}$ query.

When processing a $\mathcal{H}_x$ query on input sk, the simulator samples a value $x \in \mathbb{Z}_N$ uniformly at random as the candidate response. Let $\mathsf{pk} \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{KG}(\mathsf{sk})$. Seven distinct failure conditions arise:

Condition $\mathsf{C}_{1\text{-}\mathcal{H}_x}$. The simulator fails if the differentiator queries with some sk such that sk already appears in T^*_{sk}. Since the simulator makes at most q queries to $\mathcal{S}_{\mathsf{fail}}$, each table maintained by $\mathcal{S}_{\mathsf{fail}}$ contains at most q entries. Thus, T^*_{sk} can hold at most q private keys, each sampled uniformly at random and hidden from the differentiator. Accordingly, the probability of this event is bounded by

$$\Pr[\mathsf{C}_{1\text{-}\mathcal{H}_x}] \leq \frac{q}{N}.$$

Condition $\mathsf{C}_{2\text{-}\mathcal{H}_x}$. The simulator fails if a group element already exists with x as its discrete logarithm. As noted earlier, there are at most q discrete logarithms of group elements in T_{eq}. Therefore, the probability that a uniformly sampled x collides with one of these discrete logarithms is bounded by

$$\Pr[\mathsf{C}_{2\text{-}\mathcal{H}_x}] \leq \frac{q}{N}.$$

Condition $\mathsf{C}_{3\text{-}\mathcal{H}_x}$. The simulator fails if there exist two group elements str_1 and str_2 such that the uniformly sampled x satisfies $(\mathsf{str}_1 - x \cdot \mathsf{str}_2 = 0)$ in $\mathsf{Span}(T_{\mathsf{eq}})$. Assuming Proposition D.2 holds, each pair $(\mathsf{str}_1, \mathsf{str}_2)$ can generate at most one equation of the form $(\mathsf{str}_1 - \alpha \cdot \mathsf{str}_2 = 0)$ in $\mathsf{Span}(T_{\mathsf{eq}})$. Hence, the probability that the uniformly sampled x collides with one of these values α is bounded by

$$\Pr\left[\mathsf{C}_{3\text{-}\mathcal{H}_x} | \text{Proposition D.2 holds}\right] \leq \frac{9q^2}{N}.$$

Condition $\mathsf{C}_{4\text{-}\mathcal{H}_x}$. The simulator fails if there exist two group elements str_1 and str_2 such that the equation $(\mathsf{str}_1 - \alpha \cdot \mathsf{str}_2 - \beta \cdot \mathsf{pk} - \gamma \cdot \diamondsuit = 0)$ lies in the span of T_{eq}, and the uniformly sampled x satisfies $\beta \cdot x + \gamma = 0$. For each pair $(\mathsf{str}_1, \mathsf{str}_2)$, if more than two equations of this form exist, then pk must have a discrete logarithm in T_{eq}. By Proposition D.1, this implies that $(\mathsf{sk}, \mathsf{pk})$ exists in T^*_{sk}, which triggers the failure condition $\mathsf{C}_{1\text{-}\mathcal{H}_x}$. Therefore, assuming Proposition D.1, the probability is bounded by

$$\Pr\left[\mathsf{C}_{4\text{-}\mathcal{H}_x} | \text{Proposition D.1 holds}\right] \leq \frac{18q^2}{N}.$$

Conditions $\mathsf{C}_{5,6,7,8\text{-}\mathcal{H}_x}$. The simulator fails if there exists a group element str such that the equation $(\mathsf{str} - \alpha \cdot \mathsf{pk} - \beta \cdot \diamondsuit = 0)$ lies in the span of T_{eq}, and the uniformly sampled x yields a value $(\alpha \cdot x + \beta)$ that triggers conditions analogous to $\mathsf{C}_{2\text{-}\mathcal{H}_x}$, $\mathsf{C}_{3\text{-}\mathcal{H}_x}$, or $\mathsf{C}_{4\text{-}\mathcal{H}_x}$, or already exists in $T_{\mathcal{H}_y}$. As noted earlier, there are at most $3q$ such group elements str. Provided that Proposition D.1 holds, for each str there is at most one equation of this form, yielding at most $3q$ uniformly random values. Accordingly, the probabilities are bounded by

$$\Pr\left[\mathsf{C}_{5\text{-}\mathcal{H}_x} | \text{Proposition D.1 holds}\right] \leq 3q \cdot \frac{q}{N} = \frac{3q^2}{N},$$

$$\Pr\left[\mathsf{C}_{6\text{-}\mathcal{H}_x} | \text{Proposition D.2 holds}\right] \leq 3q \cdot \frac{9q^2}{N} = \frac{27q^3}{N},$$

$$\Pr\left[\mathsf{C}_{7\text{-}\mathcal{H}_x} | \text{Proposition D.1 holds}\right] \leq 3q \cdot \frac{q}{N} = \frac{3q^2}{N},$$

and

$$\Pr\left[\mathsf{C}_{8\text{-}\mathcal{H}_x} | \text{Proposition D.1 holds}\right] \leq 3q \cdot \frac{18q^2}{N} = \frac{54q^3}{N}.$$

When processing a $\mathcal{H}_y$ query with input r, the simulator samples a value $y \in \mathbb{Z}_N$ uniformly at random as the candidate response. If a pair $(\tilde{\mathsf{c}}, \mathsf{c}) \in T_{\mathcal{P}}$ corresponds to r, the simulator additionally inserts the equation $(\tilde{\mathsf{c}} - y \cdot \diamondsuit = 0)$ into T_{eq}. Nine distinct failure conditions arise:

Condition $\mathsf{C}_{1\text{-}\mathcal{H}_y}$. The simulator fails if the differentiator queries with some r such that r already appears in T^*_{r}. Since each randomness in T^*_{r} is sampled uniformly at random and remains hidden from the differentiator, the probability of this event is bounded by

$$\Pr[\mathsf{C}_{1\text{-}\mathcal{H}_y}] \leq \frac{q}{N}.$$

Condition $\mathsf{C}_{2\text{-}\mathcal{H}_y}$. The simulator fails if the uniformly sampled value y already exists in $T_{\mathcal{H}_y}$. As noted previously, $T_{\mathcal{H}_y}$ contains at most q tuples. Thus, the probability that a random y collides with an existing value is bounded by

$$\Pr[\mathsf{C}_{2\text{-}\mathcal{H}_y}] \leq \frac{q}{N}.$$

Condition $\mathsf{C}_{3\text{-}\mathcal{H}_y}$. The simulator fails if a group element already exists with y as its discrete logarithm. As noted previously, T_{eq} contains at most q equations, leading to at most q discrete logarithms of group elements in T_{eq}. Therefore, the probability that a uniformly sampled y collides with one of these discrete logarithms is bounded by

$$\Pr[\mathsf{C}_{3\text{-}\mathcal{H}_y}] \leq \frac{q}{N}.$$

Condition $\mathsf{C}_{4\text{-}\mathcal{H}_y}$. The simulator fails if there exist two group elements str_1 and str_2 such that the uniformly sampled y satisfies $(\mathsf{str}_1 - y \cdot \mathsf{str}_2 = 0)$ in $\mathsf{Span}(T_{\mathsf{eq}})$. By Proposition D.2, for each pair $(\mathsf{str}_1, \mathsf{str}_2)$ there is at most one equation of the form $(\mathsf{str}_1 - y \cdot \mathsf{str}_2 = 0)$ in $\mathsf{Span}(T_{\mathsf{eq}})$. Hence, the probability is bounded by

$$\Pr\left[\mathsf{C}_{4\text{-}\mathcal{H}_y} | \text{Proposition D.2 holds}\right] \leq \frac{9q^2}{N}.$$

Condition $\mathsf{C}_{5\text{-}\mathcal{H}_y}$. The simulator fails if there exist two group elements $\mathsf{str}_1, \mathsf{str}_2$ such that the equation $(\mathsf{str}_1 - \alpha \cdot \mathsf{str}_2 - \beta \cdot \tilde{\mathsf{c}} - \gamma \cdot \diamondsuit = 0)$ lies in the span of T_{eq}, and the uniformly sampled y satisfies the linear relation $\beta \cdot y + \gamma = 0$. Similarly, By Proposition D.1, the probability is bounded by

$$\Pr\left[\mathsf{C}_{5\text{-}\mathcal{H}_y} | \text{Proposition D.1 holds}\right] \leq \frac{18q^2}{N}.$$

Conditions $\mathsf{C}_{6,7,8,9\text{-}\mathcal{H}_y}$. The simulator fails if there exists a group element str such that the equation $(\mathsf{str} - \alpha \cdot \tilde{\mathsf{c}} - \beta \cdot \diamondsuit = 0)$ lies in the span of T_{eq}, and the uniformly sampled y yields a value $(\alpha \cdot y + \beta)$ that triggers conditions analogous to $\mathsf{C}_{2\text{-}\mathcal{H}_y}$, $\mathsf{C}_{3\text{-}\mathcal{H}_y}$, $\mathsf{C}_{4\text{-}\mathcal{H}_y}$, or $\mathsf{C}_{5\text{-}\mathcal{H}_y}$. As noted earlier, there are at most $3q$ such group elements str. Proposition D.1 ensures that, for each str, there is at most one equation of this form. Accordingly, the probabilities are bounded by

$$\Pr[\mathsf{C}_{6,7\text{-}\mathcal{H}_y} | \text{Proposition D.1 holds}] \leq 3q \cdot \frac{q}{N} = \frac{3q^2}{N},$$

$$\Pr[\mathsf{C}_{8\text{-}\mathcal{H}_y} | \text{Proposition D.2 holds}] \leq 3q \cdot \frac{9q^2}{N} = \frac{27q^3}{N},$$

and

$$\Pr[\mathsf{C}_{9\text{-}\mathcal{H}_y} | \text{Proposition D.1 holds}] \leq 3q \cdot \frac{18q^2}{N} = \frac{54q^3}{N}.$$

When processing a $\mathcal{H}_{\mathsf{key}}$ query on input $(\mathsf{pk}, \tilde{\mathsf{c}}, \mathsf{str})$, if these values satisfy a valid linear relation but no corresponding r exists, the simulator samples r uniformly at random and generates k as the candidate response by invoking $(\mathsf{c}, \mathsf{k}) \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{Encaps}(\mathsf{pk}, \mathsf{r})$. Two distinct failure conditions may then arise:

Condition $\mathsf{C}_{1\text{-}\mathcal{H}_{\mathsf{key}}}$. The simulator fails if the uniformly sampled value $\mathsf{r} \in \mathbb{Z}_N$ already exists in $T_{\mathcal{H}_y}$ or T^*_{r}. As noted previously, there are at most q randomness recorded in $T_{\mathcal{H}_y}$ and T^*_{r}. Thus, the probability that a random r collides with an existing value is bounded by

$$\Pr\left[\mathsf{C}_{1\text{-}\mathcal{H}_{\mathsf{key}}}\right] \leq \frac{q}{N}.$$

Condition $\mathsf{C}_{2\text{-}\mathcal{H}_{\mathsf{key}}}$. The simulator fails if the randomly generated ciphertext c already exists in $T_{\mathcal{P}}$. As noted previously, the probability is bounded by

$$\Pr\left[\mathsf{C}_{2\text{-}\mathcal{H}_{\mathsf{key}}}\right] \leq \frac{q}{N}.$$

When processing a $\mathcal{P}$ query with input $\tilde{\mathsf{c}}$, the simulator first checks the validity of $\tilde{\mathsf{c}}$ using $\mathbf{IES\text{-}KEM}.\mathsf{Encaps}$. If $\tilde{\mathsf{c}}$ is invalid, the simulator samples an invalid ciphertext c as the candidate response and two distinct failure conditions may then arise:

Condition $\mathsf{C}_{1\text{-}\mathcal{P}}$. The simulator fails if the uniformly sampled ciphertext c already exists in $T_{\mathcal{P}}$. The probability is bounded by

$$\Pr\left[\mathsf{C}_{1\text{-}\mathcal{P}}\right] \leq \frac{q}{2^\ell}.$$

Condition $\mathsf{C}_{2\text{-}\mathcal{P}}$. The simulator fails if, even after λ iterations, the uniformly sampled ciphertext c remains valid. The probability is bounded by

$$\Pr\left[\mathsf{C}_{2\text{-}\mathcal{P}}\right] \leq \left(\frac{N}{2^\ell}\right)^\lambda \leq \frac{1}{2^\lambda}.$$

For a valid $\tilde{\mathsf{c}}$, if a corresponding r exists in $T_{\mathcal{H}_y}$ or T^*_{r}, the response is determined by r. Otherwise, the simulator generates a fresh ciphertext c as the candidate response by sampling r uniformly at random and two distinct failure conditions may then arise:

Condition $\mathsf{C}_{3\text{-}\mathcal{P}}$. The simulator fails if the uniformly sampled value $\mathsf{r} \in \mathbb{Z}_N$ already exists in $T_{\mathcal{H}_y}$ or T^*_{r}. As noted previously, the probability is bounded by

$$\Pr\left[\mathsf{C}_{3\text{-}\mathcal{P}}\right] \leq \frac{q}{N}.$$

Condition $\mathsf{C}_{4\text{-}\mathcal{P}}$. The simulator fails if the randomly generated ciphertext c already appears in $T_{\mathcal{P}}$. The probability is bounded by

$$\Pr\left[\mathsf{C}_{4\text{-}\mathcal{P}}\right] \leq \frac{q}{N}.$$

When processing a $\mathcal{P}^{-1}$ query with input c, the simulator first checks the validity of c using $\mathbf{IES\text{-}KEM}.\mathsf{Decaps}$. If c is invalid, the simulator samples an invalid public key $\tilde{\mathsf{c}}$ as the candidate response and two distinct failure conditions may then arise:

Condition $\mathsf{C}_{1\text{-}\mathcal{P}^{-1}}$. The simulator fails if the uniformly sampled public key $\tilde{\mathsf{c}}$ already exists in $T_{\mathcal{P}}$. The probability is bounded by

$$\Pr\left[\mathsf{C}_{1\text{-}\mathcal{P}^{-1}}\right] \leq \frac{q}{2^\ell}.$$

Condition $\mathsf{C}_{2\text{-}\mathcal{P}^{-1}}$. The simulator fails if, even after λ iterations, the uniformly sampled public key $\tilde{\mathsf{c}}$ remains valid. The probability is bounded by

$$\Pr\left[\mathsf{C}_{2\text{-}\mathcal{P}^{-1}}\right] \leq \left(\frac{N}{2^\ell}\right)^\lambda \leq \frac{1}{2^\lambda}.$$

For a valid c, if the response is not determined, the simulator generates a fresh public key by sampling sk uniformly and setting $\tilde{c} \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{KG}(\mathsf{sk})$. Two distinct failure conditions may then arise:

Condition $\mathsf{C}_{3\text{-}\mathcal{P}^{-1}}$. The simulator fails if the randomly generated public key $\tilde{c}$ already appears in T_{eq}, T_{r}^*, $T_{\mathcal{P}}$, or $T_{\mathcal{H}_{\mathsf{key}}}$. The probability is bounded by

$$\Pr\left[\mathsf{C}_{3\text{-}\mathcal{P}^{-1}}\right] \leq \frac{3q}{N}.$$

Condition $\mathsf{C}_{4\text{-}\mathcal{P}^{-1}}$. The simulator fails if the uniformly sampled $\mathsf{sk} \in \mathbb{Z}_N$ already appears in $T_{\mathcal{H}_x}$. The probability is bounded by

$$\Pr\left[\mathsf{C}_{4\text{-}\mathcal{P}^{-1}}\right] \leq \frac{q}{N}.$$

When processing a $\mathcal{G}$ query with no corresponding entry in $\mathsf{Span}(T_{\mathsf{eq}})$, the simulator generates a fresh random public key by sampling sk uniformly as the candidate response. Let $\mathsf{str} \leftarrow \mathbf{IES\text{-}KEM}.\mathsf{KG}(\mathsf{sk})$. Two distinct failure conditions may occur:

Condition $\mathsf{C}_{1\text{-}\mathcal{G}}$. The simulator fails if the randomly generated group element str already exists in T_{eq}, T_{r}^*, $T_{\mathcal{P}}$, or $T_{\mathcal{H}_{\mathsf{key}}}$. The probability is bounded by

$$\Pr[\mathsf{C}_{1\text{-}\mathcal{G}}] \leq \frac{3q}{N}.$$

Condition $\mathsf{C}_{2\text{-}\mathcal{G}}$. The simulator fails if the sampled private key sk already exists in $T_{\mathcal{H}_x}$. Clearly, this probability is bounded by

$$\Pr[\mathsf{C}_{2\text{-}\mathcal{G}}] \leq \frac{q}{N}.$$

Combine together, we have

$$\begin{aligned}\Pr\left[\text{Failure Conditions Hold}\right] &\leq q \cdot \left(\frac{18q + 63q^2 + 162q^3}{N} + \frac{2}{2^\lambda} + \frac{2q}{2^\ell}\right) \\ &= O\left(\frac{q^4}{N} + \frac{q^2}{2^\lambda}\right)\end{aligned}$$

□
□

E Proof of Theorem 6.2

Theorem 6.2. $\Pi^{\mathcal{O}}_{\mathsf{IES\text{-}PKE}}$ is indifferentiable from the ideal PKE with validity testing $\mathbf{PKE}$. More precisely, there exists a simulator $\mathcal{S}$ such that for all q-query differentiator $\mathcal{D}$, we have

$$\mathrm{Adv}^{\mathsf{indiff}}_{\Pi^{\mathcal{O}}_{\mathsf{IES\text{-}PKE}},\mathbf{PKE},\mathcal{S}}(\mathcal{D}) \leq O\left(\frac{(\lambda \cdot q)^2}{N} + \frac{(\lambda \cdot q)^2}{2^\lambda}\right).$$

Here, the simulator makes at most $O(\lambda \cdot q)$ queries to its oracles.

Proof. According to the definition of indifferentiability, we have that, in the real world, the differentiator has three honest interfaces: $\Pi_{\mathsf{IES\text{-}PKE}}.\mathsf{KG}$, $\Pi_{\mathsf{IES\text{-}PKE}}.\mathsf{Enc}$, and $\Pi_{\mathsf{IES\text{-}PKE}}.\mathsf{Dec}$, and adversarial interfaces including random oracles $\mathcal{H}$, random permutation $(\mathcal{P}, \mathcal{P}^{-1})$, key encapsulation mechanism $\mathbf{IES\text{-}KEM}$, and authenticated encryption $(\mathcal{AE}, \mathcal{AD})$. Therefore, to complete the proof, we build an efficient simulator $\mathcal{S}$ in the ideal world, such that

1. $\mathcal{S}$ has access to the ideal PKE with validity testing $\mathbf{PKE}$;
2. $\mathcal{S}$ simulates those adversarial interfaces $(\mathcal{S}.\mathcal{H}, \mathcal{S}.\mathcal{P}, \mathcal{S}.\mathcal{P}^{-1}, \mathcal{S}.\mathbf{IES\text{-}KEM}, \mathcal{S}.\mathcal{AE}, \mathcal{S}.\mathcal{AD})$.

To describe the simulator clearly, we first clarify some notions. Let $\mathsf{sk}_{\mathsf{test}}$ denote a specific private key in $\mathcal{SK}$. Let $\mathsf{m}_{\mathsf{test}}$ denote a specific message in $\mathcal{M}$. Let $\mathsf{r}_{\mathsf{test}}$ denote a specific randomness in $\mathcal{R}$. By $\mathsf{Coin} \xleftarrow{\$} \mathrm{Ber}(p)$, we mean that a single bit Coin is sampled from a Bernoulli distribution with probability p for $\mathsf{Coin} = 1$. We denote by $\mathsf{Ele}(T)$ the set of all group elements that appear in the table T.

In the following, we illustrate the full description of our simulator.

Simulator $\mathcal{S}$

$\underline{\mathcal{S}.\mathcal{H}(\mathsf{pk}, \mathsf{m}, \mathsf{r}):}$

if $\exists(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}}) \in T_{\mathcal{H}}$, then **return** $\tilde{\mathsf{r}}$; ..... Case 1

else:

$\tilde{\mathsf{r}} \xleftarrow{\$} \mathcal{R}$; $T_{\mathcal{H}} \leftarrow T_{\mathcal{H}} \cup \{(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}})\}$;

return $\tilde{\mathsf{r}}$; ..... Case 2

$\underline{\mathcal{S}.\mathbf{IES\text{-}KEM}.\mathsf{KG}(\mathsf{sk}):}$

return $\mathbf{PKE}.\mathsf{KG}(\mathsf{sk})$;

$\underline{\mathcal{S}.\mathbf{IES\text{-}KEM}.\mathsf{Encaps}(\mathsf{pk}, \tilde{\mathsf{r}}):}$

if $\exists(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \in T_{\mathsf{KEM}}$, then **return** $(\mathsf{c_k}, \mathsf{k}||\mathsf{c})$; ..... Case 1

if $\mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}_{\mathsf{test}}, \mathsf{r}_{\mathsf{test}}) = \perp_{\mathsf{pk}}$: **return** $\perp_{\mathsf{pk}}$; ..... Case 2

if $\exists(*, \mathsf{pk}', \tilde{\mathsf{r}}, \mathsf{c_k}, *) \in T_{\mathsf{KEM}}$:

if $\exists(\mathsf{sk}, \mathsf{pk}, \triangle, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$: replace $\triangle$ by $\tilde{\mathsf{r}}$;

return $(\mathsf{c_k}, \mathsf{k}||\mathsf{n})$; ..... Case 3

else: $\mathsf{k}||\mathsf{n} \xleftarrow{\$} \mathcal{K} \times \mathcal{N}$; $T_{\mathsf{KEM}} \leftarrow T_{\mathsf{KEM}} \cup \{(\triangle, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n})\}$;

return $(\mathsf{c_k}, \mathsf{k}||\mathsf{n})$; ..... Case 4

if $\exists(\mathsf{pk}'||\mathsf{m}'||\mathsf{r}', \tilde{\mathsf{r}}) \in T_{\mathcal{H}}$:

$\mathsf{c}' \leftarrow \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}', \mathsf{m}', \mathsf{r}')$;

if $\exists(\mathsf{c_k}||*, \mathsf{c}') \in T_{\mathcal{P}}$:

if $\exists(\mathsf{sk}, \mathsf{pk}, \triangle, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \in T_{\mathsf{KEM}}$: replace $\triangle$ by $\tilde{\mathsf{r}}$;

return $(\mathsf{c_k}, \mathsf{k}||\mathsf{n})$; ..... Case 5

else: $\mathsf{k}||\mathsf{n} \xleftarrow{\$} \mathcal{K} \times \mathcal{N}$; $T_{\mathsf{KEM}} \leftarrow T_{\mathsf{KEM}} \cup \{(\triangle, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n})\}$;

return $(\mathsf{c_k}, \mathsf{k}||\mathsf{n})$; ..... Case 6

$(\mathsf{c_k}, \mathsf{k}||\mathsf{n}) \xleftarrow{\$} \mathcal{C}_{\mathsf{k}} \times \mathcal{K} \times \mathcal{N}$; $T_{\mathsf{KEM}} \leftarrow T_{\mathsf{KEM}} \cup \{(\triangle, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n})\}$;

return $(\mathsf{c_k}, \mathsf{k}||\mathsf{n})$. ..... Case 7

$\underline{\mathcal{S}.\mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c_k}):}$

if $\exists(\mathsf{sk}, *, *, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$, then **return** $\mathsf{k}||\mathsf{n}$; ..... Case 1

$\mathsf{pk} \leftarrow \mathbf{PKE}.\mathsf{KG}(\mathsf{sk})$;

if $\exists(\triangle, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \in T_{\mathsf{KEM}}$: replace $\triangle$ by sk;

return $\mathsf{k}||\mathsf{n}$; ..... Case 2

if $\mathsf{c_k} \in T_\perp$, then **return** $\perp_{\mathsf{c}}$; ..... Case 3

if $\mathsf{c_k}$ does not appear in T_{KEM} and $T_{\mathcal{P}}$:

$b \xleftarrow{\$} \mathrm{Ber}(\frac{|\mathcal{R}|-|\mathsf{Ele}(T_{\mathsf{KEM}} \cup T_{\mathcal{P}} \cup T_\perp)|+|\mathsf{Ele}(T_\perp)|}{|\mathcal{C}_\mathsf{k}|-|\mathsf{Ele}(T_{\mathsf{KEM}} \cup T_{\mathcal{P}} \cup T_\perp)|})$;

if $b = 0$: $T_\perp \leftarrow T_\perp \cup \{\mathsf{c_k}\}$; **return** $\perp_{\mathsf{c}}$; ..... Case 4

$\mathsf{k}||\mathsf{n} \xleftarrow{\$} \mathcal{K} \times \mathcal{N}$; $T_{\mathsf{KEM}} \leftarrow T_{\mathsf{KEM}} \cup \{(\mathsf{sk}, \mathsf{pk}, \triangle, \mathsf{c_k}, \mathsf{k}||\mathsf{n})\}$;

return $\mathsf{k}||\mathsf{n}$; ..... Case 5

$\underline{\mathcal{S}.\mathcal{AE}(\mathsf{k}, \mathsf{n}, \mathsf{m})}$:

if $\exists(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}) \in T_{\mathcal{AE}}$, **then return** $\mathsf{c_m}$; ..... Case 1

if $\exists(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \in T_{\mathsf{KEM}}$:

if $\exists(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}}) \in T_{\mathcal{H}}$:

$\mathsf{c} \leftarrow \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$;

if $\exists(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}) \in T_{\mathcal{P}}$: $T_{\mathcal{AE}} \leftarrow T_{\mathcal{AE}} \cup \{\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}\}$;

return $\mathsf{c_m}$; ..... Case 2

if $\exists(\mathsf{sk}, \mathsf{pk}, \triangle, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \in T_{\mathsf{KEM}}$:

if $\exists(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}) \in T_{\mathcal{P}}$ s.t. $\mathbf{PKE}.\mathsf{Dec}(\mathsf{sk}, \mathsf{c}) = \mathsf{m}$:

$T_{\mathcal{AE}} \leftarrow T_{\mathcal{AE}} \cup \{\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}\}$;

return $\mathsf{c_m}$; ..... Case 3

$\mathsf{c_m} \xleftarrow{\$} \mathcal{C}_\mathsf{m}$; $T_{\mathcal{AE}} \leftarrow T_{\mathcal{AE}} \cup \{(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})\}$;

return $\mathsf{c_m}$; ..... Case 4

$\underline{\mathcal{S}.\mathcal{AD}(\mathsf{k}, \mathsf{n}, \mathsf{c_m})}$:

if $\exists(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}) \in T_{\mathcal{AE}}$, then **return** m; ..... Case 1

if $\exists(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \in T_{\mathsf{KEM}}$:

if $\exists(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}}) \in T_{\mathcal{H}}$:

$\mathsf{c} \leftarrow \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$;

if $\exists(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}) \in T_{\mathcal{P}}$: $T_{\mathcal{AE}} \leftarrow T_{\mathcal{AE}} \cup \{(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})\}$;

return m; ..... Case 2

if $\exists(\mathsf{sk}, \mathsf{pk}, \triangle, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \in T_{\mathsf{KEM}}$:

if $\exists(\mathsf{c_k}||\mathsf{c_m},\mathsf{c}) \in T_{\mathcal{P}}$: $\mathsf{m} \leftarrow \mathbf{PKE}.\mathsf{Dec}(\mathsf{sk},\mathsf{c})$;
$T_{\mathcal{AE}} \leftarrow T_{\mathcal{AE}} \cup \{(\mathsf{k},\mathsf{n},\mathsf{m},\mathsf{c_m})\}$;
return m; ..... Case 3

$T_{\mathcal{AE}} \leftarrow T_{\mathcal{AE}} \cup \{(\mathsf{k},\mathsf{n},\perp,\mathsf{c_m})\}$;

return $\perp$; ..... Case 4

$\underline{\mathcal{S}.\mathcal{P}(\mathsf{c_k}||\mathsf{c_m})}$:

if $\exists(\mathsf{c_k}||\mathsf{c_m},\mathsf{c}) \in T_{\mathcal{P}}$, then **return** c; ..... Case 1

if $\mathsf{c_k} \in T_{\perp}$:

for attempt $= 1$ to λ do: $\mathsf{c} \xleftarrow{\$} \mathcal{C}$;
if $\mathbf{PKE}.\mathsf{Dec}(\mathsf{sk_{test}},\mathsf{c}) = \perp_{\mathsf{c}}$ **then break**;

$T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\mathsf{c_k}||\mathsf{c_m},\mathsf{c})\}$;

return c; ..... Case 2

for any $(*,\mathsf{pk},\tilde{\mathsf{r}},\mathsf{c_k},\mathsf{k}||\mathsf{n}) \in T_{\mathsf{KEM}}$:

if $\exists(\mathsf{k},\mathsf{n},\mathsf{m},\mathsf{c_m}) \in T_{\mathcal{AE}}$:

if $\exists(\mathsf{pk}||\mathsf{m}||\mathsf{r},\tilde{\mathsf{r}}) \in T_{\mathcal{H}}$: $\mathsf{c} \leftarrow \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk},\mathsf{m},\mathsf{r})$;
$T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\mathsf{c_k}||\mathsf{c_m},\mathsf{c})\}$;
return c; ..... Case 3

else: $\mathsf{r} \xleftarrow{\$} \mathcal{R}$; $\mathsf{c} \leftarrow \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk},\mathsf{m},\mathsf{r})$;
$T^*_{\mathsf{r}} \leftarrow T^*_{\mathsf{r}} \cup \{\mathsf{r}\}$; $T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\mathsf{c_k}||\mathsf{c_m},\mathsf{c})\}$;
return c; ..... Case 4

for any $(\mathsf{sk},\mathsf{pk},\triangle,\mathsf{c_k},\mathsf{k}||\mathsf{n}) \in T_{\mathsf{KEM}}$:

if $\exists(\mathsf{k},\mathsf{n},\mathsf{m},\mathsf{c_m}) \in T_{\mathcal{AE}}$:

if $\exists(\mathsf{pk}||\mathsf{m}||\mathsf{r},\tilde{\mathsf{r}}) \in T_{\mathcal{H}}$ with $(*,\mathsf{pk}',\tilde{\mathsf{r}},\mathsf{c_k},*) \in T_{\mathsf{KEM}}$:
$\mathsf{c} \leftarrow \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk},\mathsf{m},\mathsf{r})$; $T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\mathsf{c_k}||\mathsf{c_m},\mathsf{c})\}$;
return c; ..... Case 5

else: $\mathsf{r} \xleftarrow{\$} \mathcal{R}$; $\mathsf{c} \leftarrow \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk},\mathsf{m},\mathsf{r})$;
$T^*_{\mathsf{r}} \leftarrow T^*_{\mathsf{r}} \cup \{\mathsf{r}\}$; $T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\mathsf{c_k}||\mathsf{c_m},\mathsf{c})\}$;
return c; ..... Case 6

if $\mathsf{c_k}$ does not appear in T_{KEM} and $T_{\mathcal{P}}$:

$b \xleftarrow{\$} \mathrm{Ber}(\frac{|\mathcal{R}|-|\mathsf{Ele}(T_{\mathsf{KEM}} \cup T_{\mathcal{P}} \cup T_{\perp})|+|\mathsf{Ele}(T_{\perp})|}{|\mathcal{C}_{\mathsf{k}}|-|\mathsf{Ele}(T_{\mathsf{KEM}} \cup T_{\mathcal{P}} \cup T_{\perp})|})$;

if $b = 0$: $T_{\perp} \leftarrow T_{\perp} \cup \{\mathsf{c_k}\}$;
for attempt $= 1$ to λ do:
$\mathsf{c} \xleftarrow{\$} \mathcal{C}$;
if $\mathbf{PKE}.\mathsf{Dec}(\mathsf{sk_{test}},\mathsf{c}) = \perp_{\mathsf{c}}$ **then break**;

$T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})\}$;
return c; ..... Case 7

$(\mathsf{sk}, \mathsf{m}, \mathsf{r}) \xleftarrow{\$} \mathcal{SK} \times \mathcal{M} \times \mathcal{R}$; $\mathsf{c} \leftarrow \mathbf{PKE}.\mathsf{Enc}(\mathbf{PKE}.\mathsf{KG}(\mathsf{sk}), \mathsf{m}, \mathsf{r})$;

$T^*_{\mathsf{r}} \leftarrow T^*_{\mathsf{r}} \cup \{\mathsf{r}\}$; $T^*_{\mathsf{sk}} \leftarrow T^*_{\mathsf{sk}} \cup \{\mathsf{sk}\}$; $T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})\}$;

return c; ..... Case 8

$\underline{\mathcal{S}.\mathcal{P}^{-1}(\mathsf{c})}$:

if $\exists(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}) \in T_{\mathcal{P}}$, then **return** $\mathsf{c_k}||\mathsf{c_m}$; ..... Case 9

if $\mathbf{PKE}.\mathsf{Dec}(\mathsf{sk_{test}}, \mathsf{c}) = \perp_{\mathsf{c}}$: $(\mathsf{c_k}, \mathsf{c_m}) \xleftarrow{\$} \mathcal{C}_{\mathsf{k}} \times \mathcal{C}_{\mathsf{m}}$; $T_{\perp} \leftarrow T_{\perp} \cup \{\mathsf{c_k}\}$;

$T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})\}$;
return $\mathsf{c_k}||\mathsf{c_m}$; ..... Case 10

if $\exists(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}}) \in T_{\mathcal{H}}$ s.t. $\mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r}) = \mathsf{c}$:

if $\exists(*, \mathsf{pk}', \tilde{\mathsf{r}}, \mathsf{c_k}, *) \in T_{\mathsf{KEM}}$:

if $\exists(*, \mathsf{pk}, *, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \in T_{\mathsf{KEM}}$ with $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}) \in T_{\mathcal{AE}}$:

$T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})\}$;
return $\mathsf{c_k}||\mathsf{c_m}$; ..... Case 11

else: $\mathsf{c_m} \xleftarrow{\$} \mathcal{C}_{\mathsf{m}}$; $T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})\}$;
return $\mathsf{c_k}||\mathsf{c_m}$; ..... Case 12

$(\mathsf{c_k}, \mathsf{c_m}) \xleftarrow{\$} \mathcal{C}_{\mathsf{k}} \times \mathcal{C}_{\mathsf{m}}$; $T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})\}$;

return $\mathsf{c_k}||\mathsf{c_m}$. ..... Case 13

We immediately note that if the differentiator $\mathcal{D}$ makes q queries to the honest and adversarial interfaces, then the simulator $\mathcal{S}$ makes at most $O(q^2)$ number of queries to the ideal PKE with validity testing $\mathbf{PKE}$, indicating that the simulator $\mathcal{S}$ above is efficient.

Next, we establish that the view of $\mathcal{D}$ in the real world is indistinguishable from its view in the ideal world. To this end, we construct a sequence of indistinguishable hybrid games transitioning from the ideal game $\mathbf{Ideal}^{\mathbf{PKE}}_{\mathcal{S}}$ to the real game $\mathbf{Real}^{\Pi^{\mathcal{O}}_{\mathsf{IES\text{-}PKE}}}$. Specifically,

1. **Ideal Game**. We begin with the ideal game $\mathbf{Ideal}^{\mathbf{PKE}}_{\mathcal{S}}$, in which $\mathcal{D}$ interacts with the honest interface $\mathbf{PKE}$ and the adversarial interface $\mathcal{S}^{\mathbf{PKE}}$.

2. **Game $\mathbf{G}_1$**. A relay algorithm $\mathcal{F}$ is introduced as an intermediary between $\mathcal{D}$ and ideal PKE. Here, $\mathcal{D}$ interacts with $\mathcal{F}^{\mathbf{PKE}}$ and $\mathcal{S}^{\mathbf{PKE}}$, where $\mathcal{F}$ simply forwards queries to $\mathbf{PKE}$ and returns the corresponding outputs.

3. **Game $\mathbf{G}_2$**. We refine the simulator by introducing several failure conditions on its query–response pairs. Whenever a failure condition is triggered, the modified simulator $\mathcal{S}_{\mathsf{fail}}$ aborts explicitly. In this game, $\mathcal{D}$ interacts with $\mathcal{F}^{\mathbf{PKE}}$ and $\mathcal{S}^{\mathbf{PKE}}_{\mathsf{fail}}$.

4. **Game $\mathbf{G}_3$**. The relay algorithm $\mathcal{F}$ is replaced with the construction $\Pi_{\mathsf{IES\text{-}PKE}}$, as defined earlier. The

construction $\Pi_{\mathsf{IES\text{-}PKE}}$ employs $\mathcal{S}_{\mathsf{fail}}$ in place of the ideal model. The central goal is to show that, except with negligible probability of triggering a failure condition (from Game 2), the responses generated by $\Pi_{\mathsf{IES\text{-}PKE}}$ remain consistent with those of $\mathbf{PKE}$. In this game, $\mathcal{D}$ interacts with $\Pi^{\mathcal{S}_{\mathsf{fail}}}_{\mathsf{IES\text{-}PKE}}$ and $\mathcal{S}^{\mathbf{PKE}}_{\mathsf{fail}}$.

5. **Game $\mathbf{G}_4$**. The simulator is further modified to sample its responses independently of the ideal PKE. Moreover, the new simulator $\mathcal{S}_{\mathsf{samp}}$ no longer checks for failure conditions. In this game, $\mathcal{D}$ interacts with $\Pi^{\mathcal{S}_{\mathsf{samp}}}_{\mathsf{IES\text{-}PKE}}$ and $\mathcal{S}_{\mathsf{samp}}$.

6. **Real Game**. Finally, we arrive at the real game $\mathbf{Real}^{\Pi^{\mathcal{O}}_{\mathsf{IES\text{-}PKE}}}$, in which the simulator $\mathcal{S}_{\mathsf{samp}}$ from Game 4 is replaced with the real oracle $\mathcal{O}$. Thus, $\mathcal{D}$ interacts with $\Pi^{\mathcal{O}}_{\mathsf{IES\text{-}PKE}}$ and $\mathcal{O}$.

In what follows, we give a detailed description of each hybrid game and demonstrate that the statistical distance between any pair of consecutive games is negligible.

Ideal Game. In this game, the differentiator $\mathcal{D}$ interacts with the ideal PKE and the simulator $\mathcal{S}$, as defined previously. The game outputs 1 if and only if $\mathcal{D}$ outputs 1 after this interaction. Formally,

$$\Pr[\mathbf{Ideal}^{\mathbf{PKE}}_{\mathcal{S}}(\mathcal{D}) = 1] = \Pr[\mathcal{D}^{\mathbf{PKE},\mathcal{S}^{\mathbf{PKE}}}(1^\lambda) = 1].$$

E.1 Transition from Ideal to $\mathbf{G}_1$

Game 1. This game is identical to the ideal game, except that the differentiator $\mathcal{D}$ interacts with a relay algorithm $\mathcal{F} = (\mathcal{F}.\mathsf{KG}, \mathcal{F}.\mathsf{Enc}, \mathcal{F}.\mathsf{Dec})$ rather than directly with the ideal PKE. The relay algorithm $\mathcal{F}$ forwards all queries to the ideal PKE and provides the following two interfaces:

- $\mathcal{F}.\mathsf{KG}(\mathsf{sk})$: query $\mathsf{pk} \leftarrow \mathbf{PKE}.\mathsf{KG}(\mathsf{sk})$ and return pk.
- $\mathcal{F}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$: query $\mathsf{c} \leftarrow \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$ and return c.
- $\mathcal{F}.\mathsf{Dec}(\mathsf{sk}, \mathsf{c})$: query $\mathsf{m} \leftarrow \mathbf{PKE}.\mathsf{Dec}(\mathsf{sk}, \mathsf{c})$ and return m.

Since all queries issued by $\mathcal{D}$ are answered exactly as in the ideal game, its view in Game 1 is identical to that in Ideal Game. Defining $\mathbf{G}_1$ as Game 1, which outputs 1 if and only if $\mathcal{D}$ outputs 1 after interacting with $\mathcal{F}$ and $\mathcal{S}$, we obtain

$$\begin{aligned}\Pr[\mathbf{G}_1(\mathcal{D}) = 1] &= \Pr[\mathcal{D}^{\mathcal{F}^{\mathbf{PKE}},\mathcal{S}^{\mathbf{PKE}}}(1^\lambda) = 1] \\ &= \Pr[\mathbf{Ideal}^{\mathbf{PKE}}_{\mathcal{S}}(\mathcal{D}) = 1].\end{aligned}$$

E.2 Transition from $\mathbf{G}_1$ to $\mathbf{G}_2$

Game 2. In this game, the simulator is modified to incorporate some conditional tests before returning partial responses. These conditions are specifically designed to capture correlations between the simulator's responses that could potentially be exploited by the differentiator $\mathcal{D}$. If the modified simulator $\mathcal{S}_{\mathsf{fail}}$ produces a response that triggers any of the failure conditions below, it explicitly fails instead of returning that response.

In response to a $\mathcal{H}$ query with input $(\mathsf{pk}, \mathsf{m}, \mathsf{r})$, the simulator samples a value $\tilde{\mathsf{r}} \in \mathcal{R}$ uniformly as the candidate response, and then checks the following failure conditions:

- $\mathsf{C}_{1\text{-}\mathcal{H}}(\tilde{\mathsf{r}})$: The value $\tilde{\mathsf{r}}$ already appears in T_{KEM} or $T_{\mathcal{H}}$.
- $\mathsf{C}_{2\text{-}\mathcal{H}}(\mathsf{r})$: The value r already appears in T^*_{r}.

In response to a $\mathbf{IES\text{-}KEM}.\mathsf{KG}$ query with input sk, the simulator simply returns $\mathbf{PKE}.\mathsf{KG}(\mathsf{sk})$ and performs no failure checks.

In response to a $\mathbf{IES\text{-}KEM}.\mathsf{Encaps}$ query with input $(\mathsf{pk}, \tilde{\mathsf{r}})$, the simulator first verifies the validity of pk via $\mathbf{PKE}.\mathsf{Enc}$. If pk is invalid, the simulator returns $\perp_{\mathsf{pk}}$. If pk is valid and $\tilde{\mathsf{r}}$ already appears in T_{KEM}, then a corresponding $\mathsf{c_k}$ is determined. In this case, if the differentiator has not yet decapsulated $\mathsf{c_k}$ using sk, the simulator samples k and n uniformly at random as candidate responses, and subsequently checks the following failure conditions:

- $\mathsf{C}_{\text{1-Encaps}}(\mathsf{k}||\mathsf{n})$: The value $\mathsf{k}||\mathsf{n}$ already appears in T_{KEM} or $T_{\mathcal{AE}}$.

Moreover, if $\tilde{\mathsf{r}}$ already appears in $T_{\mathcal{H}}$ with the corresponding ciphertext recorded in $T_{\mathcal{P}}$, then $\mathsf{c_k}$ is also determined. Here again, if the differentiator has not yet decapsulated $\mathsf{c_k}$ using sk, the simulator samples k and n uniformly at random as candidate responses, and subsequently checks the failure condition $\mathsf{C}_{\text{1-Encaps}}(\mathsf{k}||\mathsf{n})$. Otherwise, the simulator samples $\mathsf{c_k}$, k, and n uniformly at random as candidate responses, and then checks the failure conditions $\mathsf{C}_{\text{1-Encaps}}(\mathsf{k}||\mathsf{n})$ and:

- $\mathsf{C}_{\text{2-Encaps}}(\mathsf{c_k})$: The value $\mathsf{c_k}$ already appears in $T_{\perp}$ or T_{KEM}.

In response to a $\mathbf{IES\text{-}KEM}.\mathsf{Decaps}$ query with input $(\mathsf{sk}, \mathsf{c_k})$, the simulator proceeds as follows. Let $\mathsf{pk} \leftarrow \mathbf{PKE}.\mathsf{KG}(\mathsf{sk})$. If $\mathsf{c_k}$ is obtained through $\mathbf{IES\text{-}KEM}.\mathsf{Encaps}$ under pk, then the response $\mathsf{k}\|\mathsf{n}$ is already determined. Otherwise, if $\mathsf{c_k}$ is regarded as a valid ciphertext, the simulator samples k and n uniformly at random as the candidate response, and subsequently checks the following failure conditions:

- $\mathsf{C}_{\text{1-Decaps}}(\mathsf{k}||\mathsf{n})$: The value $\mathsf{k}||\mathsf{n}$ already appears in T_{KEM} or $T_{\mathcal{AE}}$.

- $\mathsf{C}_{\text{2-Decaps}}(\mathsf{sk})$: The value sk already appears in T^*_{sk}.

In response to a $\mathcal{AE}$ query, if the response is not already determined by the relationships among other interfaces, the simulator samples $\mathsf{c_m}$ uniformly at random as the candidate response, and then checks the following failure conditions:

- $\mathsf{C}_{\text{1-}\mathcal{AE}}(\mathsf{c_m})$: The value $\mathsf{c_m}$ already exists in $T_{\mathcal{P}}$ or $T_{\mathcal{AE}}$.

In response to a $\mathcal{P}$ query with input $\mathsf{c_k}||\mathsf{c_m}$, the simulator first checks the validity of $\mathsf{c_k}$. If $\mathsf{c_k}$ is regarded as invalid (i.e., $\mathsf{c_k} \in T_{\perp}$), the simulator samples an invalid ciphertext c as the candidate response and then checks the following failure conditions:

- $\mathsf{C}_{\text{1-}\mathcal{P}}(\mathsf{c})$: The generated c already appears in $T_{\mathcal{P}}$.

- $\mathsf{C}_{\text{2-}\mathcal{P}}(\mathsf{c})$: The generated c is valid, i.e., $\mathbf{PKE}.\mathsf{Dec}(\mathsf{sk}_{\mathsf{test}}, \mathsf{c}) \neq \perp_{\mathsf{c}}$.

If $\mathsf{c_k}$ and $\mathsf{c_m}$ are generated through Encaps and $\mathcal{AE}$ respectively, and correspond to the same pk and m, then the response is the ciphertext under $(\mathsf{pk}, \mathsf{m})$. If no corresponding randomness r exists, the simulator generates a fresh ciphertext c as the candidate response by sampling r uniformly at random, and then checks the following failure conditions:

- $\mathsf{C}_{\text{3-}\mathcal{P}}(\mathsf{r})$: The value r already appears in $T_{\mathcal{H}}$ or T^*_{r}.

- $\mathsf{C}_{\text{4-}\mathcal{P}}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$: The exists c in $T_{\mathcal{P}}$ such that $\mathsf{c} = \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$.

Otherwise, the simulator determines the validity of an unknown $\mathsf{c_k}$ by sampling from a Bernoulli distribution. If $\mathsf{c_k}$ is regarded as invalid, the simulator samples an invalid ciphertext c as the candidate response and then checks the failure conditions $\mathsf{C}_{\text{1-}\mathcal{P}}(\mathsf{c})$ and $\mathsf{C}_{\text{2-}\mathcal{P}}(\mathsf{c})$. Otherwise, for valid $\mathsf{c_k}$, the simulator generates a fresh ciphertext c as the candidate response by sampling sk, m, and r uniformly at random, and then checks the following failure conditions:

- $\mathsf{C}_{5\text{-}\mathcal{P}}(\mathsf{sk}, \mathsf{m}, \mathsf{r})$: The exists c in $T_{\mathcal{P}}$ such that $\mathsf{c} = \mathbf{PKE}.\mathsf{Enc}\left(\mathbf{PKE}.\mathsf{KG}(\mathsf{sk}), \mathsf{m}, \mathsf{r}\right)$.

- $\mathsf{C}_{6\text{-}\mathcal{P}}(\mathsf{sk})$: The value sk already appears in T_{KEM}.

In response to a $\mathcal{P}^{-1}$ query with input c, the simulator first checks the validity of c using $\mathbf{PKE}.\mathsf{Dec}$. If c is invalid, the simulator samples $\mathsf{c_k}$ and $\mathsf{c_m}$ uniformly at random as the candidate response, and then checks the following failure conditions:

- $\mathsf{C}_{1\text{-}\mathcal{P}^{-1}}(\mathsf{c_m})$: The value $\mathsf{c_m}$ already appears in $T_{\mathcal{P}}$ or $T_{\mathcal{AE}}$.

- $\mathsf{C}_{2\text{-}\mathcal{P}^{-1}}(\mathsf{c_k})$: The value $\mathsf{c_k}$ already appears in T_{KEM}.

If c is valid and there exists $(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}})$ such that $(\mathsf{pk}, \mathsf{m}, \mathsf{r})$ corresponds to c and $\tilde{\mathsf{r}}$ corresponds to $\mathsf{c_k}$, the simulator samples $\mathsf{c_m}$ uniformly at random and then checks the following failure conditions $\mathsf{C}_{1\text{-}\mathcal{P}^{-1}}(\mathsf{c_m})$ and:

- $\mathsf{C}_{3\text{-}\mathcal{P}^{-1}}(\mathsf{c_k})$: The value $\mathsf{c_k}$ already exists in $T_{\perp}$.

Otherwise, the simulator samples both $\mathsf{c_k}$ and $\mathsf{c_m}$ uniformly at random as the candidate response, and then checks the failure conditions $\mathsf{C}_{1\text{-}\mathcal{P}^{-1}}(\mathsf{c_m})$ and $\mathsf{C}_{3\text{-}\mathcal{P}^{-1}}(\mathsf{c_k})$.

Intuitively, each of the conditions above corresponds to an unwanted collision between a freshly sampled element and information already recorded in the simulator's tables; each such event can be bounded by a negligible probability. A formal probabilistic analysis is deferred to Appendix E.7.

Lemma E.1. $\Pr\left[\text{Failure Conditions Hold}\right] \leq O\left(q^2/N + q^2/2^{\lambda}\right)$.

The differentiator can distinguish between Game 1 and Game 2 only when the following two events occur:

- In Game 1, the simulator $\mathcal{S}$ does not fail, even when its response satisfies one of the failure conditions above;

- In Game 2, the simulator $\mathcal{S}_{\mathsf{fail}}$ does fail.

Let $\mathbf{G}_2$ denoted Game 2, which outputs 1 if and only if $\mathcal{D}$ outputs 1 after interacting with $\mathcal{F}$ and $\mathcal{S}_{\mathsf{fail}}$. Formally,

$$\Pr[\mathbf{G}_2(\mathcal{D})] = \Pr[\mathcal{D}^{\mathcal{F}^{\mathbf{PKE}}, \mathcal{S}_{\mathsf{fail}}^{\mathbf{PKE}}}(1^{\lambda}) = 1].$$

Then

$$\begin{aligned} |\Pr[\mathbf{G}_2(\mathcal{D}) = 1] - \Pr[\mathbf{G}_1(\mathcal{D}) = 1]| &\leq 2 \cdot \Pr[\text{Failure Conditions Hold}] \\ &\leq O\left(\frac{q^2}{N} + \frac{q^2}{2^{\lambda}}\right). \end{aligned}$$

E.3 Properties under the absence of failure conditions

Before describing Game 3, we state several propositions that record basic observations and will be useful in the proofs of the subsequent lemmas.

Proposition E.1. *Suppose that none of the failure conditions holds. Then there do not exist two distinct pairs* $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ *and* $(\mathsf{k}, \mathsf{n}, \mathsf{m}', \mathsf{c_m})$ *in* $T_{\mathcal{AE}}$ *with* $\mathsf{m} \neq \mathsf{m}'$.

Proof. For any pair $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ inserted by the $\mathcal{AE}(\mathsf{k}, \mathsf{n}, \mathsf{m})$ query, rather than Case 1 where the corresponding response is already recorded in $T_{\mathcal{AE}}$, the simulator proceeds according to one of the remaining three cases, which correspond to those already specified in the simulator's description.

Case 2. In this case, k and n are generated via the KEM interfaces, with the corresponding $\tilde{\mathsf{r}}$ recorded; that is, there exists a tuple $(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$ in T_{KEM}. If there further exists $(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}}) \in T_{\mathcal{H}}$ and $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}) \in T_{\mathcal{P}}$ such that $\mathsf{c} = \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$, then the simulator returns $\mathsf{c_m}$.

Case 3. In this case, k and n are generated via the Decaps interface, i.e., there exists a tuple $(\mathsf{sk}, \mathsf{pk}, \triangle, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$ in T_{KEM}. If there further exists $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}) \in T_{\mathcal{P}}$ such that $\mathsf{m} = \mathbf{PKE}.\mathsf{Dec}(\mathsf{sk}, \mathsf{c})$, then the simulator returns $\mathsf{c_m}$.

Case 4. If none of the above cases applies, the simulator returns a uniformly random $\mathsf{c_m}$.

If the pair is inserted by the $\mathcal{AD}(\mathsf{k}, \mathsf{n}, \mathsf{c_m})$ query, then, as described for the simulator, rather than Case 1 where the corresponding response is already recorded in $T_{\mathcal{AE}}$, the situation falls into one of three possibilities. Case 2 and Case 3 for $\mathcal{AD}$ correspond directly to Case 2 and Case 3 of $\mathcal{AE}$, with identical conditions. Finally, in Case 4, if the response is not already determined by the above relationships, the simulator returns $\perp$.

Suppose there exist two distinct pairs $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ and $(\mathsf{k}, \mathsf{n}, \mathsf{m}', \mathsf{c_m})$ in $T_{\mathcal{AE}}$. Then $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ must have been inserted first, either by a $\mathcal{AE}$ or $\mathcal{AD}$ query, and only afterwards $(\mathsf{k}, \mathsf{n}, \mathsf{m}', \mathsf{c_m})$ must have inserted by a $\mathcal{AE}$ query. Any such collision necessarily implies that one of the failure conditions is triggered. Specifically,

- If both $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ and $(\mathsf{k}, \mathsf{n}, \mathsf{m}', \mathsf{c_m})$ are inserted under Cases 2 or 3, and no $\mathsf{k}||\mathsf{n}$ collision exists in T_{KEM} (i.e., conditions $\mathsf{C}_{\text{1-Encaps}}$ and $\mathsf{C}_{\text{1-Decaps}}$ are not triggered), then they correspond to the same tuple $(*, \mathsf{pk}, *, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$ in T_{KEM}. Moreover, the associated tuples $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ and $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}')$ must have been inserted via a $\mathcal{P}^{-1}$ query, with corresponding $(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}})$ and $(\mathsf{pk}||\mathsf{m}'||\mathsf{r}', \tilde{\mathsf{r}})$ in $T_{\mathcal{H}}$; otherwise, $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ or $(\mathsf{k}, \mathsf{n}, \mathsf{m}', \mathsf{c_m})$ would already exist. Consequently, a $\tilde{\mathsf{r}}$ collision occurs in $T_{\mathcal{H}}$, triggering condition $\mathsf{C}_{\text{2-}\mathcal{H}}$.
- If $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ is first inserted by Case 3 of $\mathcal{AE}$, where $\mathsf{c_m}$ is sampled uniformly, and later $(\mathsf{k}, \mathsf{n}, \mathsf{m}', \mathsf{c_m})$ is inserted by Case 2 or 3, then since $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}')$ corresponding to $(\mathsf{k}, \mathsf{n}, \mathsf{m}', \mathsf{c_m})$ is inserted via $\mathcal{P}^{-1}$ with $\mathsf{c_m}$ also uniformly sampled, a collision occurs between $\mathsf{c_m}$ in $T_{\mathcal{AE}}$ and $T_{\mathcal{P}}$, thereby triggering the failure conditions $\mathsf{C}_{\text{1-}\mathcal{AE}}$ or $\mathsf{C}_{\text{2-}\mathcal{P}^{-1}}$.

□
□

Proposition E.2. *Suppose that none of the failure conditions holds. Then there do not exist two distinct pairs* $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ *and* $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}')$ *in* $T_{\mathcal{AE}}$ *with* $\mathsf{c_m} \neq \mathsf{c_m}'$.

Proof. Suppose there exist two distinct pairs $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ and $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}')$ in $T_{\mathcal{AE}}$. Then $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ must have been inserted first, either by a $\mathcal{AE}$ or $\mathcal{AD}$ query, and only afterwards $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}')$ must have inserted by a $\mathcal{AD}$ query. Any such collision necessarily implies that one of the failure conditions is triggered. Specifically,

- If both $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ and $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}')$ are inserted under Cases 2 or 3, and no $\mathsf{k}||\mathsf{n}$ collision exists in T_{KEM} (i.e., conditions $\mathsf{C}_{\text{1-Encaps}}$ and $\mathsf{C}_{\text{1-Decaps}}$ are not triggered), then they correspond to the same tuple $(*, \mathsf{pk}, *, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$ in T_{KEM}. Moreover, the associated tuples $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ and $(\mathsf{c_k}||\mathsf{c_m}', \mathsf{c}')$ must have been inserted via a $\mathcal{P}^{-1}$ query, with corresponding $(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}})$ and $(\mathsf{pk}'||\mathsf{m}'||\mathsf{r}', \tilde{\mathsf{r}})$ in $T_{\mathcal{H}}$; otherwise, $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ or $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}')$ would already exist. Consequently, a $\tilde{\mathsf{r}}$ collision occurs in $T_{\mathcal{H}}$, triggering condition $\mathsf{C}_{\text{2-}\mathcal{H}}$.
- If $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ is first inserted by Case 3 of $\mathcal{AE}$, and then $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}')$ is inserted by Case 2 or 3, then, provided no failure conditions are triggered, the corresponding $(\mathsf{c_k}||\mathsf{c_m}', \mathsf{c})$ already exists in $T_{\mathcal{P}}$ and $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ exists in $T_{\mathcal{AE}}$. The simulator must therefore ensure that $\mathsf{c_m} = \mathsf{c_m}'$.

□
□

Proposition E.3. *Suppose that none of the failure conditions holds. Then there do not exist two distinct pairs* $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ *and* $(\mathsf{c_k}'||\mathsf{c_m}', \mathsf{c})$ *in* $T_{\mathcal{P}}$ *with* $\mathsf{c_k}||\mathsf{c_m} \neq \mathsf{c_k}'||\mathsf{c_m}'$.

Proof. For any pair $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ inserted by the $\mathcal{P}(\mathsf{c_k}||\mathsf{c_m})$ query, rather than Case 1 where the corresponding response is already recorded in $T_{\mathcal{P}}$, the simulator proceeds according to one of the remaining seven cases, which correspond to those already specified in the simulator's description.

Case 2. If $\mathsf{c_k}$ is regarded as invalid, the simulator repeatedly samples ciphertexts c uniformly at random until an invalid one is obtained, making at most λ attempts.

Case 3. If $\mathsf{c_k}$ and $\mathsf{c_m}$ are generated via Encaps and $\mathcal{AE}$ for the same pk and m, that is, there exist $(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \in T_{\mathsf{KEM}}$ and $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}) \in T_{\mathcal{AE}}$, and if the corresponding randomness r also exists, i.e., $(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}}) \in T_{\mathcal{H}}$, then the simulator returns the ciphertext c under $(\mathsf{pk}, \mathsf{m}, \mathsf{r})$.

Case 4. Under the same pk and m as in Case 3, if no corresponding r exists, the simulator samples r uniformly at random, generates c via $\mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$, and records r in T_{r}^*.

Case 5. If $\mathsf{c_k}$ and $\mathsf{c_m}$ are generated via Decaps and $\mathcal{AE}$ for the same pk and m, that is, there exist $(\mathsf{sk}, \mathsf{pk}, \triangle, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \in T_{\mathsf{KEM}}$ and $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}) \in T_{\mathcal{AE}}$, and if the corresponding randomness r also exists, i.e., $(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}}) \in T_{\mathcal{H}}$, then the simulator returns the ciphertext c under $(\mathsf{pk}, \mathsf{m}, \mathsf{r})$.

Case 6. Under the same pk and m as in Case 5, if no corresponding r exists, the simulator samples r uniformly at random, generates c via $\mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$, and records r in T_{r}^*.

Case 7. If the validity of $\mathsf{c_k}$ is unknown, the simulator decides its validity by sampling from a Bernoulli distribution. If $\mathsf{c_k}$ is deemed invalid, the simulator repeatedly samples ciphertexts c uniformly at random until an invalid one is obtained, performing at most λ attempts.

Case 8. If $\mathsf{c_k}$ is regarded as valid but none of the above cases applies, the simulator samples sk, m, and r uniformly at random to generate a ciphertext c. It then records r in T_{r}^* and sk in T_{sk}^*.

If the pair $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ inserted by the $\mathcal{P}^{-1}(\mathsf{c})$ query, rather than Case 9 where the corresponding response is already recorded in $T_{\mathcal{P}}$, the simulator proceeds according to one of the remaining four cases, which correspond to those already specified in the simulator's description.

Case 10. If c is invalid, the simulator returns uniformly sampled $\mathsf{c_k}$ and $\mathsf{c_m}$, and records $\mathsf{c_k}$ in $T_{\perp}$.

Case 11. If there exist $(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}}) \in T_{\mathcal{H}}$ such that $\mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r}) = \mathsf{c}$, and if a corresponding $\mathsf{c_k}$ exists for $\tilde{\mathsf{r}}$ with $\mathsf{c_m}$ being the encryption of m under $\mathsf{k}||\mathsf{n}$ associated with $\mathsf{c_k}$ and pk, then the response is fully determined.

Case 12. Given a determined $\mathsf{c_k}$ as in Case 11, if no corresponding $\mathsf{c_m}$ exists, the simulator samples $\mathsf{c_m}$ uniformly at random.

Case 13. If none of the above cases applies, the simulator samples both $\mathsf{c_k}$ and $\mathsf{c_m}$ uniformly at random and returns them.

Suppose there exist two distinct pairs $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ and $(\mathsf{c_k}'||\mathsf{c_m}', \mathsf{c})$ in $T_{\mathcal{P}}$. Then $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ must have been inserted first, either by a $\mathcal{P}$ or $\mathcal{P}^{-1}$ query, and only afterwards $(\mathsf{c_k}'||\mathsf{c_m}', \mathsf{c})$ must have inserted by a $\mathcal{P}$ query. Any such collision necessarily implies that one of the failure conditions is triggered. Specifically,

- If $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ is first inserted by Cases 2, 7, or 10, and then $(\mathsf{c_k}'||\mathsf{c_m}', \mathsf{c})$ is inserted by Cases 2 or 7, the randomly sampled invalid c must collide with an existing ciphertext in $T_{\mathcal{P}}$ (i.e., condition $\mathsf{C}_{1\text{-}\mathcal{P}}$ is triggered).

- If $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ is first inserted by Case 2 or 7, and later $(\mathsf{c_k}'||\mathsf{c_m}', \mathsf{c})$ is inserted by Case 3, 4, 5, 6, or 8, this indicates that in Case 2 or 7 the simulator still produced a valid ciphertext even after λ iterations (i.e., condition $\mathsf{C}_{2\text{-}\mathcal{P}}$ is triggered).
- If $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ is first inserted by Case 3, 4, 5, 6, 8, 11, 12, or 13, and later $(\mathsf{c_k}'||\mathsf{c_m}', \mathsf{c})$ is inserted by Case 2 or 7, condition $\mathsf{C}_{2\text{-}\mathcal{P}}$ is triggered as well.
- If $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ is first inserted by Case 3, 5, or 11, and later $(\mathsf{c_k}'||\mathsf{c_m}', \mathsf{c})$ is inserted by Case 3 or 5, there must exist both $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ and $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}')$, contradicting Proposition E.2.
- If $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ is first inserted by Case 3, 5, 11, or 12, and later $(\mathsf{c_k}'||\mathsf{c_m}', \mathsf{c})$ is inserted by Case 4, 6, or 8, there must exist one r that simultaneously appears in both $T_{\mathcal{H}}$ and T_{r}^* (i.e., conditions $\mathsf{C}_{3\text{-}\mathcal{H}}$ or $\mathsf{C}_{3\text{-}\mathcal{P}}$ are triggered).
- If $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ is first inserted by Case 4, 6, or 8, and later $(\mathsf{c_k}'||\mathsf{c_m}', \mathsf{c})$ is inserted by Case 3 or 5, conditions $\mathsf{C}_{3\text{-}\mathcal{H}}$ or $\mathsf{C}_{3\text{-}\mathcal{P}}$ are triggered as well.
- If both $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ and $(\mathsf{c_k}'||\mathsf{c_m}', \mathsf{c})$ are inserted by Case 4, 6, or 8, there must exist a r collision in T_{r}^* (i.e., condition $\mathsf{C}_{4\text{-}\mathcal{P}}$ is triggered).
- If $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ is first inserted by Case 12 or 13, and later $(\mathsf{c_k}'||\mathsf{c_m}', \mathsf{c})$ is inserted by Case 3 or 5, then—provided no failure conditions occur—the simulator guarantees that $\mathsf{c_k}||\mathsf{c_m} = \mathsf{c_k}'||\mathsf{c_m}'$.
- If $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ is first inserted by Case 13, and later $(\mathsf{c_k}'||\mathsf{c_m}', \mathsf{c})$ is inserted by Case 4, 6, or 5, the randomly generated valid ciphertext c must collide with an existing ciphertext in $T_{\mathcal{P}}$ (i.e., conditions $\mathsf{C}_{5\text{-}\mathcal{P}}$ or $\mathsf{C}_{6\text{-}\mathcal{P}}$ are triggered).

□
□

Proposition E.4. *Suppose that none of the failure conditions holds. Then there do not exist two distinct pairs* $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ *and* $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}')$ *in* $T_{\mathcal{P}}$ *with* $\mathsf{c} \neq \mathsf{c}'$.

Proof. Suppose there exist two distinct pairs $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ and $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}')$ in $T_{\mathcal{P}}$. Then $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ must have been inserted first, either by a $\mathcal{P}$ or $\mathcal{P}^{-1}$ query, and only afterwards $(\mathsf{c_k}'||\mathsf{c_m}', \mathsf{c})$ must have inserted by a $\mathcal{P}^{-1}$ query. Any such collision necessarily implies that one of the failure conditions is triggered. Specifically,

- If $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ is first inserted either $\mathcal{P}$ or $\mathcal{P}^{-1}$, and then $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}')$ is inserted by Case 10, 12 or 13, the randomly sampled $\mathsf{c_k}||\mathsf{c_m}$ must collide with an existing ciphertext in $T_{\mathcal{P}}$ (i.e., condition $\mathsf{C}_{1\text{-}\mathcal{P}^{-1}}$ is triggered).
- If $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ is first inserted by Case 2, 7, or 10, and then $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}')$ is inserted by Case 11, there must exist one $\mathsf{c_k}$ that simultaneously appears in both T_{KEM} and $T_{\perp}$ (i.e., conditions $\mathsf{C}_{2\text{-}\mathsf{Encaps}}$, $\mathsf{C}_{2\text{-}\mathcal{P}^{-1}}$, or $\mathsf{C}_{4\text{-}\mathcal{P}^{-1}}$ are triggered).
- If $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ is first inserted by Case 3, 4, 5, or 6, and then $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}')$ is inserted by Case 11, then—provided there is no $\tilde{\mathsf{r}}$ collision in $T_{\mathcal{H}}$ and no $\mathsf{k}||\mathsf{n}$ collision in T_{KEM}—there must exist $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ and $(\mathsf{k}', \mathsf{n}', \mathsf{m}', \mathsf{c_m})$ with $\mathsf{c_m}$ sampled uniformly at random, thereby triggering condition $\mathsf{C}_{2\text{-}\mathcal{AE}}$.
- If $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ is first inserted by Case 3, 4, 5, 6, or 11, and then $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}')$ is inserted by Case 11, then—provided there is no $\tilde{\mathsf{r}}$ collision in $T_{\mathcal{H}}$ and no $\mathsf{k}||\mathsf{n}$ collision in T_{KEM}—there must exist $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ and $(\mathsf{k}', \mathsf{n}', \mathsf{m}', \mathsf{c_m})$ with $\mathsf{c_m}$ sampled uniformly at random, thereby triggering condition $\mathsf{C}_{2\text{-}\mathcal{AE}}$.
- If $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ is first inserted by Case 8, 12, or 13, and then $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}')$ is inserted by Case 11, then—provided there is no $\tilde{\mathsf{r}}$ collision in $T_{\mathcal{H}}$ and no $\mathsf{k}||\mathsf{n}$ collision in T_{KEM}—the uniformly sampled $\mathsf{c_m}$ by either $\mathcal{P}^{-1}$ or $\mathcal{AE}$ must collide with the recorded cipher (i.e., conditions $\mathsf{C}_{1\text{-}\mathcal{AE}}$, $\mathsf{C}_{2\text{-}\mathcal{AE}}$, or $\mathsf{C}_{1\text{-}\mathcal{P}^{-1}}$ are triggered).

□
□

To facilitate the subsequent analysis, we use the symbol "$\prec$" to indicate the order in which elements are recorded in different tables by the simulator. For instance, when Item_1 is recorded in T_1 prior to Item_2 being recorded in T_2, we write

$$\mathsf{Item}_1 \in \mathsf{T}_1 \prec \mathsf{Item}_2 \in \mathsf{T}_2.$$

If the context is sufficiently clear, this can be abbreviated as $\mathsf{Item}_1 \prec \mathsf{Item}_2$.

Proposition E.5. *Suppose that none of the failure conditions holds. Then for any tuple* $(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}})$ *in* $T_{\mathcal{H}}$, $(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$ *in* T_{KEM}, $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ *in* $T_{\mathcal{AE}}$, *and* $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ *in* $T_{\mathcal{P}}$, *it holds that*

$$\mathsf{c} = \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r}).$$

Proof. When inserting $(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}})$ into $T_{\mathcal{H}}$, where $\tilde{\mathsf{r}}$ is sampled uniformly at random, provided that $\tilde{\mathsf{r}}$ does not already apppears in T_{KEM} (i.e., condition $\mathsf{C}_{1\text{-}\mathcal{H}}(\tilde{\mathsf{r}})$ does not hold), we have

$$(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}}) \in T_{\mathcal{H}} \prec (*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \in T_{\mathsf{KEM}}$$

If $(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$ is inserted through the encapsulation interface in Case 3 or 4, then there already exists $\mathsf{c_k}$ in T_{KEM} corresponding to $\tilde{\mathsf{r}}$; that is, $(*, \mathsf{pk}', \tilde{\mathsf{r}}, \mathsf{c_k}, *) \in T_{\mathsf{KEM}}$ with $\mathsf{c_k}$ sampled uniformly at random. Provided that conditions $\mathsf{C}_{1,2\text{-Encaps}}$, $\mathsf{C}_{1\text{-Decaps}}$, and $\mathsf{C}_{2\text{-}\mathcal{P}^{-1}}$ are not triggered, the order is:

$$(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}}) \prec (*, \mathsf{pk}', \tilde{\mathsf{r}}, \mathsf{c_k}, *) \prec (*, \mathsf{pk}, *, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \prec (\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}).$$

If $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ is inserted by Case 2 or 3 of $\mathcal{AE}$ or $\mathcal{AD}$, then $(\mathsf{c_m}||\mathsf{c_k}, \mathsf{c})$ must already exist in $T_{\mathcal{P}}$ with

$$\mathsf{c} = \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r}).$$

Otherwise, if $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ is inserted by Case 4 of $\mathcal{AE}$, and condition $\mathsf{C}_{1\text{-}\mathcal{AE}}$ is not triggered, we have

$$(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}) \prec (\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}).$$

If $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ is inserted by $\mathcal{P}$, the simulator encures that

$$\mathsf{c} = \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r}).$$

Otherwise, if $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ is inserted by $\mathcal{P}^{-1}$ with $\mathsf{c} \neq \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$, this is equivalent to the existence of an c-collision in $T_{\mathcal{P}}$, which is shown to be impossible in Proposition E.4.

If $(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$ is inserted through the encapsulation interface in Case 5 or 6, then—provided there is no $\tilde{\mathsf{r}}$-collision in $T_{\mathcal{H}}$ (i.e., condition $\mathsf{C}_{2\text{-}\mathcal{H}}$) and failure conditions $\mathsf{C}_{1\text{-Encaps}}$ and $\mathsf{C}_{1\text{-Decaps}}$ are not triggered—we have

$$(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}}), (\mathsf{c_k}||\mathsf{c_m}', \mathsf{c}') \prec (*, \mathsf{pk}, *, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \prec (\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}),$$

with $\mathsf{c}' = \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$. By Proposition E.2, the pair $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ must be inserted through Case 2 or 3 of $\mathcal{AE}$ or $\mathcal{AD}$, and the simulator enforces $\mathsf{c_m} = \mathsf{c_m}'$. Consequently, by Proposition E.4, we obtain

$$\mathsf{c} = \mathsf{c}' = \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r}).$$

If $(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$ is inserted through the encapsulation interface in Case 7, we obtain

$$(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}}) \prec (*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \prec (\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}), (\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}).$$

If $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ is inserted by Case 2 or 3 of $\mathcal{AE}$ or $\mathcal{AD}$, then, as noted previously, the pair $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ already exists with $\mathsf{c} = \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$. Otherwise, we have $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}) \prec (\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$, and, as argued before, $\mathsf{c} = \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$, provided that none of the failure conditions are triggered.

Thus, by considering all cases above, we conclude that, as long as none of the failure conditions occur, the simulator assigns $\mathsf{c} = \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$ as stated in the proposition.

□ □

Proposition E.6. *Suppose that none of the failure conditions holds. Then for any tuple* $(\mathsf{pk}||\mathsf{m}||\mathsf{r},\tilde{\mathsf{r}})$ *in* $T_{\mathcal{H}}$, $(*,\widetilde{\mathsf{pk}},\tilde{\mathsf{r}},\mathsf{c_k},*)$ *and* $(\mathsf{sk},\mathsf{pk},\triangle,\mathsf{c_k},\mathsf{k}||\mathsf{n})$ *in* T_{KEM}, $(\mathsf{k},\mathsf{n},\mathsf{m},\mathsf{c_m})$ *in* $T_{\mathcal{AE}}$, *and* $(\mathsf{c_k}||\mathsf{c_m},\mathsf{c})$ *in* $T_{\mathcal{P}}$, *it holds that*

$$\mathsf{c} = \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk},\mathsf{m},\mathsf{r}).$$

Proof. The proof is identical to the case analyzed in Proposition E.5, where $(*,\mathsf{pk},\tilde{\mathsf{r}},\mathsf{c_k},\mathsf{k}||\mathsf{n})$ is inserted through the encapsulation interface in Case 3. □ □

Proposition E.7. *Suppose that none of the failure conditions holds. Then for any tuple* $(\mathsf{pk}||\mathsf{m}||\mathsf{r},\tilde{\mathsf{r}})$ *in* $T_{\mathcal{H}}$, $(*,*,\tilde{\mathsf{r}},\mathsf{c_k},*)$ *in* T_{KEM}, *and* $(\mathsf{c_k}'||\mathsf{c_m},\mathsf{c})$ *in* $T_{\mathcal{P}}$ *such that* $\mathsf{c} = \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk},\mathsf{m},\mathsf{r})$, *it holds that* $\mathsf{c_k} = \mathsf{c_k}'$.

Proof. Provided that the uniformly sampled $\tilde{\mathsf{r}}$ from $\mathcal{H}$ does not already appear in T_{KEM}, we have

$$(\mathsf{pk}||\mathsf{m}||\mathsf{r},\tilde{\mathsf{r}}) \prec (*,*,\tilde{\mathsf{r}},\mathsf{c_k},*).$$

Without loss of generality, suppose that $(*,\mathsf{pk},\tilde{\mathsf{r}},\mathsf{c_k},*)$ is the first tuple inserted into T_{KEM} with $\mathsf{c_k}$ corresponding to $\tilde{\mathsf{r}}$. If $(*,\mathsf{pk},\tilde{\mathsf{r}},\mathsf{c_k},*)$ is inserted through the encapsulation interface in Case 5 or 6, then $(\mathsf{c_k}'||\mathsf{c_m},\mathsf{c})$ must already exist with $\mathsf{c_k} = \mathsf{c_k}'$.

If $(*,\mathsf{pk},\tilde{\mathsf{r}},\mathsf{c_k},*)$ is inserted through the encapsulation interface in Case 7, then

$$(\mathsf{pk}||\mathsf{m}||\mathsf{r},\tilde{\mathsf{r}}) \prec (*,*,\tilde{\mathsf{r}},\mathsf{c_k},*) \prec (\mathsf{c_k}'||\mathsf{c_m},\mathsf{c}).$$

Provided that no r appears simultaneously in both $T_{\mathcal{H}}$ and T^*_{r}, the pair $(\mathsf{c_k}'||\mathsf{c_m},\mathsf{c})$ may be inserted by $\mathcal{P}$ in Case 3 or 5, or by $\mathcal{P}^{-1}$. In all such cases, the simulator ensures that $\mathsf{c_k} = \mathsf{c_k}'$.

□ □

Proposition E.8. *Suppose that none of the failure conditions holds. Then for any tuple* $(\mathsf{pk}||\mathsf{m}||\mathsf{r},\tilde{\mathsf{r}})$ *in* $T_{\mathcal{H}}$, $(*,\mathsf{pk},\tilde{\mathsf{r}},\mathsf{c_k},\mathsf{k}||\mathsf{n})$ *in* T_{KEM}, $(\mathsf{k},\mathsf{n},\mathsf{m},\mathsf{c_m})$ *in* $T_{\mathcal{AE}}$, *and* $(\mathsf{c_k}||\mathsf{c_m}',\mathsf{c})$ *in* $T_{\mathcal{P}}$ *such that* $\mathsf{c} = \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk},\mathsf{m},\mathsf{r})$, *it holds that* $\mathsf{c_m} = \mathsf{c_m}'$.

Proof. Provided that the uniformly sampled $\tilde{\mathsf{r}}$ from $\mathcal{H}$ does not already appear in T_{KEM}, we have

$$(\mathsf{pk}||\mathsf{m}||\mathsf{r},\tilde{\mathsf{r}}) \prec (*,\mathsf{pk},\tilde{\mathsf{r}},\mathsf{c_k},\mathsf{k}||\mathsf{n}).$$

If $(*,\mathsf{pk},\tilde{\mathsf{r}},\mathsf{c_k},\mathsf{k}||\mathsf{n})$ is inserted through the encapsulation interface in Case 3, then $\mathsf{c_k}$ must already exist in T_{KEM} corresponding to $\tilde{\mathsf{r}}$; that is, $(*,\mathsf{pk}',\tilde{\mathsf{r}},\mathsf{c_k},*) \in T_{\mathsf{KEM}}$ with $\mathsf{c_k}$ sampled uniformly at random. If $(*,\mathsf{pk}',\tilde{\mathsf{r}},\mathsf{c_k},*)$ is inserted through the encapsulation interface in Case 5 or 6, then, provided that no failure conditions occur, we have

$$(\mathsf{c_k}||\mathsf{c_m}',\mathsf{c}) \prec (\mathsf{sk},\mathsf{pk},\triangle,\mathsf{c_k},\mathsf{k}||\mathsf{n}) \prec (\mathsf{k},\mathsf{n},\mathsf{m},\mathsf{c_m}).$$

Here, the pair $(\mathsf{k},\mathsf{n},\mathsf{m},\mathsf{c_m})$ must be inserted by $\mathcal{AE}$ in Case 2 or 3, or by $\mathcal{AD}$ in Case 2, and the simulator ensures that $\mathsf{c_m} = \mathsf{c_m}'$. Else if $(*,\mathsf{pk}',\tilde{\mathsf{r}},\mathsf{c_k},*)$ is inserted through the encapsulation interface in Case 7, then, provided that no failure conditions occur, we have

$$(\mathsf{pk}||\mathsf{m}||\mathsf{r},\tilde{\mathsf{r}}) \prec (*,\mathsf{pk}',\tilde{\mathsf{r}},\mathsf{c_k},*) \prec (\mathsf{sk},\mathsf{pk},\triangle,\mathsf{c_k},\mathsf{k}||\mathsf{n}) \prec (\mathsf{k},\mathsf{n},\mathsf{m},\mathsf{c_m}),$$

and

$$(\mathsf{pk}||\mathsf{m}||\mathsf{r},\tilde{\mathsf{r}}) \prec (*,\mathsf{pk}',\tilde{\mathsf{r}},\mathsf{c_k},*) \prec (\mathsf{c_k}||\mathsf{c_m}',\mathsf{c}).$$

In this case, the pair $(\mathsf{k},\mathsf{n},\mathsf{m},\mathsf{c_m})$ must be inserted by $\mathcal{AE}$ in Case 2, 3, or 4, and the simulator ensures that $\mathsf{c_m} = \mathsf{c_m}'$.

If $(*,\mathsf{pk},\tilde{\mathsf{r}},\mathsf{c_k},\mathsf{k}||\mathsf{n})$ is inserted through the encapsulation interface in Case 4, then $(,\mathsf{pk}',\tilde{\mathsf{r}},\mathsf{c_k},*) \in T_{\mathsf{KEM}}$ already exists with $\mathsf{c_k}$ sampled uniformly at random. If $(*,\mathsf{pk}',\tilde{\mathsf{r}},\mathsf{c_k},*)$ is inserted through the encapsulation interface in Case 5 or 6, then, provided that no failure conditions occur, we have

$$(\mathsf{pk}||\mathsf{m}||\mathsf{r},\tilde{\mathsf{r}}),(\mathsf{c_k}||\mathsf{c_m}',\mathsf{c}) \prec (*,\mathsf{pk},\tilde{\mathsf{r}},\mathsf{c_k},\mathsf{k}||\mathsf{n}) \prec (\mathsf{k},\mathsf{n},\mathsf{m},\mathsf{c_m}).$$

Here, $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ must be inserted by $\mathcal{AE}$ or $\mathcal{AD}$ in Case 2, with $\mathsf{c_m} = \mathsf{c_m}'$. If instead $(*, \mathsf{pk}', \tilde{\mathsf{r}}, \mathsf{c_k}, *)$ is inserted through the encapsulation interface in Case 7, then, provided that no failure conditions occur, we have

$$(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}}) \prec (*, \mathsf{pk}', \tilde{\mathsf{r}}, \mathsf{c_k}, *) \prec (\triangle, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \prec (\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}),$$

and

$$(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}}) \prec (*, \mathsf{pk}', \tilde{\mathsf{r}}, \mathsf{c_k}, *) \prec (\mathsf{c_k}||\mathsf{c_m}', \mathsf{c}).$$

In this case, the pair $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ must be inserted by $\mathcal{AE}$ in Case 2, 3, or 4, and the simulator again enforces $\mathsf{c_m} = \mathsf{c_m}'$.

If $(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$ is inserted through the encapsulation interface in Case 5, 6, or 7, then the reasoning is analogous to the cases above.

Thus, by considering all possibilities, we conclude that as long as no failure conditions occur, the simulator always ensures $\mathsf{c_m} = \mathsf{c_m}'$, as stated in the proposition. □

□

Proposition E.9. *Suppose that none of the failure conditions holds. Then for any tuple* $(\mathsf{sk}, \mathsf{pk}, *, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$ *in* T_{KEM}, $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ *in* $T_{\mathcal{AE}}$, *and* $(\mathsf{c_k}||\mathsf{c_m}', \mathsf{c})$ *in* $T_{\mathcal{P}}$ *such that* $\mathsf{m} = \mathbf{PKE}.\mathsf{Dec}(\mathsf{sk}, \mathsf{c})$, *it holds that* $\mathsf{c_m} = \mathsf{c_m}'$.

Proof. Since $\mathbf{PKE}.\mathsf{Enc}$ is injective from the public key, message, and randomness to the ciphertext, each c can be decrypted by exactly one sk. Therefore, if $(\mathsf{c_k}||\mathsf{c_m}', \mathsf{c})$ is inserted by $\mathcal{P}$ in Case 3, 4, 5, or 6, or by $\mathcal{P}^{-1}$ in Case 11, it must already hold that $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}')$ exists. By Proposition E.2, it follows that $\mathsf{c_m} = \mathsf{c_m}'$.

If $(\mathsf{c_k}||\mathsf{c_m}', \mathsf{c})$ is inserted by $\mathcal{P}$ in Case 8, the ciphertext c is generated by sampling sk, m, and r uniformly at random, with sk recorded in T^*_{sk}. In this situation, sk appears simultaneously in both T_{KEM} and T^*_{sk}, thereby triggering condition $\mathsf{C}_{2\text{-}\mathsf{Decaps}}$ or $\mathsf{C}_{7\text{-}\mathcal{P}}$.

If $(\mathsf{c_k}||\mathsf{c_m}', \mathsf{c})$ is inserted by $\mathcal{P}^{-1}$ in Case 12, then $(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}})$ must already exist in $T_{\mathcal{H}}$ with $\mathsf{c} = \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$, and $\mathsf{c_k}$ in T_{KEM} corresponding to $\tilde{\mathsf{r}}$. Provided that no failure conditions occur, we have

$$(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}}) \prec (*, \mathsf{pk}', \tilde{\mathsf{r}}, \mathsf{c_k}, *) \prec (\mathsf{c_k}||\mathsf{c_m}', \mathsf{c}) \prec (\mathsf{sk}, \mathsf{pk}, *, \mathsf{c_k}, \mathsf{k}||\mathsf{n}).$$

The tuple $(\mathsf{sk}, \mathsf{pk}, *, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$ may be inserted through the decapsulation interface in Case 2 or 5, yielding

$$(\mathsf{c_k}||\mathsf{c_m}', \mathsf{c}) \prec (*, \mathsf{pk}, *, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \prec (\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}).$$

By Proposition E.2, the pair $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ must then be inserted by $\mathcal{AE}$ or $\mathcal{AD}$ in Case 2 or 3, and the simulator enforces $\mathsf{c_m} = \mathsf{c_m}'$.

Thus, by considering all cases above, we conclude that, as long as none of the failure conditions occur, the simulator indeed assigns $\mathsf{c_m} = \mathsf{c_m}'$ as stated in the proposition. □

□

Proposition E.10. *Suppose that none of the failure conditions holds. Then for any tuple* $(\mathsf{sk}, \mathsf{pk}, *, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$ *in* T_{KEM}, $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ *in* $T_{\mathcal{AE}}$, *and* $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ *in* $T_{\mathcal{P}}$, *it holds that*

$$\mathsf{m} = \mathbf{PKE}.\mathsf{Dec}(\mathsf{sk}, \mathsf{c}).$$

Proof. Provided that no failure conditions occur, if $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ is inserted by $\mathcal{AE}$ or $\mathcal{AD}$ in Case 2 or 3, then clearly $\mathsf{m} = \mathbf{PKE}.\mathsf{Dec}(\mathsf{sk}, \mathsf{c})$.

If $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ is inserted by $\mathcal{AE}$ in Case 4, where $\mathsf{c_m}$ is sampled uniformly at random, we have

$$(*, \mathsf{pk}, *, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \prec (\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}) \prec (\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}).$$

If $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ is inserted by $\mathcal{P}$, the simulator sets c as the encryption of m under pk, implying $\mathsf{m} = \mathbf{PKE}.\mathsf{Dec}(\mathsf{sk}, \mathsf{c})$. If $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ is inserted by $\mathcal{P}^{-1}$ with $\mathsf{m} \neq \mathbf{PKE}.\mathsf{Dec}(\mathsf{sk}, \mathsf{c})$, this would amount to the existence of an c-collision in $T_{\mathcal{P}}$, which is ruled out by Proposition E.4.

If $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c}_\mathsf{m})$ is inserted by $\mathcal{AD}$ in Case 4 with $\mathsf{m} \neq \mathbf{PKE}.\mathsf{Dec}(\mathsf{sk}, \mathsf{c})$, then $\mathsf{m} = \perp$ even though c is a valid ciphertext under pk. In this case, $(\mathsf{c}_\mathsf{k} || \mathsf{c}_\mathsf{m}, \mathsf{c})$ must have been inserted by $\mathcal{P}$ in Case 8 or by $\mathcal{P}^{-1}$ in Case 12 or 13, leading to either a collision of sk between T_{KEM} and T^*_{sk}, or a collision on m.

Thus, by considering all cases above, we conclude that, as long as none of the failure conditions occur, the simulator enforces

$$\mathsf{m} = \mathbf{PKE}.\mathsf{Dec}(\mathsf{sk}, \mathsf{c})$$

as stated in the proposition. □
□

Proposition E.11. *Suppose that none of the failure conditions holds. Then for any tuple* $(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c}_\mathsf{k}, \mathsf{k} || \mathsf{n})$ *in* T_{KEM}, $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c}_\mathsf{m})$ *in* $T_{\mathcal{AE}}$, *and* $(\mathsf{c}_\mathsf{k} || \mathsf{c}_\mathsf{m}, \mathsf{c})$ *in* $T_{\mathcal{P}}$, *it holds that*

$$\mathsf{m} = \mathbf{PKE}.\mathsf{Dec}(\mathsf{sk}, \mathsf{c}),$$

where $\mathsf{pk} = \mathbf{PKE}.\mathsf{KG}(\mathsf{sk})$.

The proof is identical to that of Proposition E.10.

E.4 Transition from $\mathbf{G}_2$ to $\mathbf{G}_3$

Game 3. In this game, the simulator $\mathcal{S}_{\mathsf{fail}}$ remains unchanged. Instead, the relay algorithm $\mathcal{F} = (\mathcal{F}.\mathsf{KG}, \mathcal{F}.\mathsf{Enc}, \mathcal{F}.\mathsf{Dec})$ is replaced with the construction $\Pi_{\mathsf{IES\text{-}PKE}} = (\Pi_{\mathsf{IES\text{-}PKE}}.\mathsf{KG}, \Pi_{\mathsf{IES\text{-}PKE}}.\mathsf{Enc}, \Pi_{\mathsf{IES\text{-}PKE}}.\mathsf{Dec})$. Notably, $\Pi_{\mathsf{IES\text{-}PKE}}$ does not query the ideal PKE directly; rather, it interacts with $\mathcal{S}_{\mathsf{fail}}$, thereby ensuring that the honest interfaces are consistent with the adversarial interfaces. Consequently, the differentiator can distinguish between Game 2 and Game 3 only if it detects an inconsistency between $\mathcal{F}$ and $\mathcal{S}_{\mathsf{fail}}$ in Game 2.

Let $\mathbf{G}_3$ denote Game 3, which outputs 1 if and only if $\mathcal{D}$ outputs 1 after interacting with $\Pi_{\mathsf{IES\text{-}PKE}}$ and $\mathcal{S}_{\mathsf{fail}}$. Formally,

$$\Pr[\mathbf{G}_3(\mathcal{D}) = 1] = \Pr[\mathcal{D}^{\Pi^{\mathcal{S}_{\mathsf{fail}}}_{\mathsf{IES\text{-}PKE}}, \mathcal{S}^{\mathbf{PKE}}_{\mathsf{fail}}}(1^\lambda) = 1].$$

Lemma E.2. *Let q denote the number of queries made by the differentiator to both the honest and adversarial interfaces. Then,*

$$|\Pr[\mathbf{G}_3(\mathcal{D}) = 1] - \Pr[\mathbf{G}_2(\mathcal{D}) = 1]| \leq O\left(\frac{(\lambda \cdot q)^2}{N} + \frac{(\lambda \cdot q)^2}{2^\lambda}\right).$$

Proof. To prove that the views in Game 2 and Game 3 are statistically close, we argue that the differentiator can only exploit an inconsistency if the simulator $\mathcal{S}_{\mathsf{fail}}$ fails in at least one of the two games.

For Game 3, recall that the honest interfaces produce their responses by querying the adversarial interfaces, namely the simulator $\mathcal{S}_{\mathsf{fail}}$. Hence, as long as the simulator provides consistent answers to identical queries across all invocations, the responses of the honest interface necessarily coincide with those of the adversarial interface. For random oracle and KEM interfaces, this consistency holds trivially. Proposition E.1 and E.2 guarantee that a AE/AD query cannot produce more than one distinct response, which Proposition E.3 and E.4 ensure the consistency of permutation.

For Game 2, any inconsistency between the honest interface $\mathcal{F}$ and the adversarial interface $\mathcal{S}_{\mathsf{fail}}$ can occur only when the response generated by $\mathcal{F}$ deviates from that of the simulator implementing the Π_{IES} construction. We now establish that, provided $\mathcal{S}_{\mathsf{fail}}$ does not fail, such an inconsistency cannot arise.

Inconsistency with respect to $\mathcal{F}$.KG. For any public-key generation query on input sk, both the relay algorithm $\mathcal{F}$ and the simulator $\mathcal{S}_{\mathsf{fail}}.\mathbf{IES\text{-}KEM}.\mathsf{KG}$ return $\mathsf{pk} \leftarrow \mathbf{PKE}.\mathsf{KG}(\mathsf{sk})$. Therefore, consistency holds directly.

Inconsistency with respect to $\mathcal{F}$.Enc. For any encryption query on input pk, m and r, the relay algorithm $\mathcal{F}$ always returns $\mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$. Consistency therefore requires that the following two hold:

1. If pk is invalid with respect to the ideal PKE, then $\mathcal{S}_{\mathsf{fail}}.\mathbf{IES\text{-}KEM}.\mathsf{Encaps}$ on input pk must return $\perp_{\mathsf{pk}}$.

2. Otherwise, let $\mathsf{c} \leftarrow \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$, the simulator $\mathcal{S}_{\mathsf{fail}}$ must return the same ciphertext c.

Based on the description of the simulator, when processing an encapsulation query with input pk, the simulator determines its validity by querying $\mathbf{PKE}.\mathsf{Enc}$. Therefore, for any public key pk that is invalid with respect to the ideal PKE, we must have

$$\mathcal{S}_{\mathsf{fail}}.\mathbf{IES\text{-}KEM}.\mathsf{Encaps}(\mathsf{pk}, *) = \perp_{\mathsf{pk}}.$$

Now consider any private key $\mathsf{sk} \in \mathbb{Z}_N$, with $\mathsf{pk} \leftarrow \mathbf{PKE}.\mathsf{KG}(\mathsf{sk})$, and any message m and randomness r. When the differentiator obtains the ciphertext c corresponding to $(\mathsf{pk}, \mathsf{m}, \mathsf{r})$ through $\mathcal{S}_{\mathsf{fail}}$, it must query $\mathcal{P}$ on input $\mathsf{c_k}||\mathsf{c_m}$, where $\mathsf{c_m}$ is the authenticated-encryption ciphertext of m under the key $\mathsf{k}||\mathsf{n}$ and $\mathsf{c_k}$ associated with $(\mathsf{pk}, \mathsf{m}, \mathsf{r})$. In particular, the relationship between $(\mathsf{pk}, \mathsf{m}, \mathsf{r})$ and $\mathsf{c_k}||\mathsf{c_m}$ must fall into one of the following cases:

- Form 1: There exist tuple $(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}})$ in $T_{\mathcal{H}}$, tuple $(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$ in T_{KEM}, and tuple $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})$ in $T_{\mathcal{AE}}$.
- Form 2: There exist tuple $(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}})$ in $T_{\mathcal{H}}$, an element $\mathsf{c_k}$ corresponding to $\tilde{\mathsf{r}}$ with $(\mathsf{sk}, \mathsf{pk}, \triangle, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \in T_{\mathsf{KEM}}$, and a tuple $(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}) \in T_{\mathcal{AE}}$.

For Form 1, Proposition E.5 implies that

$$\mathcal{P}(\mathsf{c_k}||\mathsf{c_m}) = \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r}).$$

For Form 2, the existence of an element $\mathsf{c_k}$ in T_{KEM} corresponding to $\tilde{\mathsf{r}}$ means that $(*, \mathsf{pk}', \tilde{\mathsf{r}}, \mathsf{c_k}, *) \in T_{\mathsf{KEM}}$. In this case, Proposition E.6 ensures that $\mathcal{P}(\mathsf{c_k}||\mathsf{c_m})$ is consistent.

Furthermore, the differentiator may also issue a query $\mathsf{c_k}||\mathsf{c_m} \leftarrow \mathcal{P}^{-1}(\mathsf{c})$ with $\mathsf{c} = \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$. In this case, the simulator $\mathcal{S}_{\mathsf{fail}}$ must likewise return $\mathsf{c_k}$ and $\mathsf{c_m}$ whenever the relevant associations hold.

For the consistency of $\mathsf{c_k}$, the simulator queries $\mathcal{H}$ on input $(\mathsf{pk}, \mathsf{m}, \mathsf{r})$ to obtain $\tilde{\mathsf{r}}$, and $\mathsf{c_k}$ must correspond to $\tilde{\mathsf{r}}$ in T_{KEM}. Thus, there exist $(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}}) \in T_{\mathcal{H}}$, $(*, \mathsf{pk}', \tilde{\mathsf{r}}, \mathsf{c_k}, *) \in T_{\mathsf{KEM}}$, and $(\mathsf{c_k}'||\mathsf{c_m}, \mathsf{c}) \in T_{\mathcal{P}}$, and Proposition E.7 ensures that $\mathsf{c_k} = \mathsf{c_k}'$.

For the consistency of $\mathsf{c_m}$, the simulator must query $\mathcal{AE}$ on $(\mathsf{k}, \mathsf{n}, \mathsf{m})$ to obtain $\mathsf{c_m}'$, where $\mathsf{k}||\mathsf{n}$ either directly corresponds to $(\mathsf{pk}, \mathsf{m}, \mathsf{r})$ or arises from the decapsulation of $\mathsf{c_k}$ under sk. Specifically, the relationship between $(\mathsf{pk}, \mathsf{m}, \mathsf{r})$ and $\mathsf{k}||\mathsf{n}$ must fall into one of the following cases:

- Form 1: There exists tuple $(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}})$ in $T_{\mathcal{H}}$ and tuple $(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$ in T_{KEM}.
- Form 2: There exists tuple $(\mathsf{sk}, \mathsf{pk}, *, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$ in T_{KEM}.

Proposition E.8 and Proposition E.9 respectively guarantee that in both forms, $\mathsf{c_m} = \mathsf{c_m}'$.

Inconsistency with respect to $\mathcal{F}.\mathsf{Dec}$. For any decryption query on input $(\mathsf{sk}, \mathsf{c})$, the relay algorithm always returns $\mathbf{PKE}.\mathsf{Dec}(\mathsf{sk}, \mathsf{c})$. Consistency therefore requires that the following conditions hold:

1. If c is invalid with respect to the ideal PKE, and there exists $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ in $T_{\mathcal{P}}$, then $\mathcal{S}_{\mathsf{fail}}.\mathbf{IES\text{-}KEM}.\mathsf{Decaps}$ on input $\mathsf{c_k}$ must return $\perp_{\mathsf{c}}$.
2. Otherwise, let $\mathsf{m} \leftarrow \mathbf{PKE}.\mathsf{Dec}(\mathsf{sk}, \mathsf{c})$, the simulator $\mathcal{S}_{\mathsf{fail}}$ must return the same ciphertext c.

By the construction of the simulator, whenever $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ is inserted into $T_{\mathcal{P}}$ with c invalid, the corresponding $\mathsf{c_k}$ must be recorded in $T_{\perp}$. If $\mathcal{S}_{\mathsf{fail}}$ does not return $\perp_{\mathsf{c}}$, then $\mathsf{c_k}$ simultaneously appears in both T_{KEM} and $T_{\perp}$, thereby triggering conditions $\mathsf{C}_{2\text{-}\mathsf{Encaps}}$, $\mathsf{C}_{2\text{-}\mathcal{P}^{-1}}$, or $\mathsf{C}_{4\text{-}\mathcal{P}^{-1}}$. Thus we have

$$\mathcal{S}_{\mathsf{fail}}.\mathbf{IES\text{-}KEM}.\mathsf{Decaps}(*, \mathsf{c_k}) = \perp_{\mathsf{c}}.$$

Now consider any private key sk and ciphertext c with $\mathsf{m} \leftarrow \mathbf{PKE}.\mathsf{Dec}(\mathsf{sk}, \mathsf{c})$, where $\mathsf{c} = \mathbf{PKE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$ and $\mathsf{pk} = \mathbf{PKE}.\mathsf{KG}(\mathsf{sk})$. When the differentiator decrypts c under sk through $\mathcal{S}_{\mathsf{fail}}$, it must first query $\mathsf{c_k}||\mathsf{c_m} \leftarrow \mathcal{P}^{-1}(\mathsf{c})$ and then obtain $\mathsf{m}' \leftarrow \mathcal{AD}(\mathsf{k}, \mathsf{n}, \mathsf{c_k})$, where $\mathsf{k}||\mathsf{n}$ corresponds to c. The relationship between $(\mathsf{sk}, \mathsf{c})$ and $\mathsf{k}||\mathsf{n}$ must fall into one of the following forms:

- Form 1: There exists tuple $(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$ in T_{KEM}.
- Form 2: There exists tuple $(\mathsf{sk}, \mathsf{pk}, *, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$ in T_{KEM}.

Proposition E.10 and Proposition E.11 together guarantee that in both forms, $\mathsf{m} = \mathbf{PKE}.\mathsf{Dec}(\mathsf{sk}, \mathsf{c})$.

Hence, as long as the simulator $\mathcal{S}_{\mathsf{fail}}$ does not fail in either Game 2 or Game 3, the differentiator's view is statistically indistinguishable between the two games. Let q_{hon} and q_{adv} denote the number of queries made to the honest and adversarial interfaces, with $q_{\text{hon}} + q_{\text{adv}} \leq q$. Furthermore, each query to the honest interface in Game 3 requires at most λ queries to the simulator. Then,

$$\begin{aligned}|\Pr[\mathbf{G}_3(\mathcal{D}) = 1] - \Pr[\mathbf{G}_2(\mathcal{D}) = 1]| &\leq 2 \cdot \Pr[\mathcal{S}_{\mathsf{fail}} \text{ fails}] \\ &\leq O\left(\frac{(\lambda \cdot q)^2}{N} + \frac{(\lambda \cdot q)^2}{2^\lambda}\right).\end{aligned}$$

□
□

E.5 Transition from $\mathbf{G}_3$ to $\mathbf{G}_4$

Game 4. In this game, we modify the simulator such that its responses no longer depend on the ideal PKE. Specifically, the new simulator $\mathcal{S}_{\mathsf{samp}}$ implements each interface using lazy sampling, and all failure conditions are removed from $\mathcal{S}_{\mathsf{samp}}$.

We use the notation $\mathsf{Coin} \xleftarrow{\$} \mathrm{Ber}(p)$ to denote that a single bit Coin is sampled from a Bernoulli distribution with success probability p (i.e., $\Pr[\mathsf{Coin} = 1] = p$). We write $\mathsf{Ele\text{-}pk}(T)$ for the set of public keys $\mathsf{pk} \in \mathcal{PK}$ that appear in T, and $\mathsf{Ele\text{-}c_k}(T)$ for the set of ciphertexts $\mathsf{c_k} \in \mathcal{C_k}$ that appear in T.

Simulator $\mathcal{S}_{\mathsf{samp}}$

$\mathcal{S}_{\mathsf{samp}}.\mathcal{H}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$:

if $\exists(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}}) \in T_{\mathcal{H}}$, then **return** $\tilde{\mathsf{r}}$;

$\tilde{\mathsf{r}} \xleftarrow{\$} \mathcal{R}$; $T_{\mathcal{H}} \leftarrow T_{\mathcal{H}} \cup \{(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}})\}$;

return $\tilde{\mathsf{r}}$;

$\mathcal{S}_{\mathsf{samp}}.\mathbf{IES\text{-}KEM}.\mathsf{KG}(\mathsf{sk})$:

if $\exists(\mathsf{sk}, \mathsf{pk}) \in T_{\mathsf{KG}}$, then **return** pk;

$\mathsf{pk} \xleftarrow{\$} \mathcal{PK}$; $T_{\mathsf{KG}} \leftarrow T_{\mathsf{KG}} \cup \{(\mathsf{sk}, \mathsf{pk})\}$;

return pk;

$\mathcal{S}_{\mathsf{samp}}.\mathbf{IES\text{-}KEM}.\mathsf{Encaps}(\mathsf{pk}, \tilde{\mathsf{r}})$:

if $\exists(*, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \in T_{\mathsf{KEM}}$, then **return** $(\mathsf{c_k}, \mathsf{k}||\mathsf{c})$;

if $\mathsf{pk} \in T_{\perp_{\mathsf{pk}}}$: **return** $\perp_{\mathsf{pk}}$;

if $\exists(*, \mathsf{pk}', \tilde{\mathsf{r}}, \mathsf{c_k}, *) \in T_{\mathsf{KEM}}$:

if $\exists(\mathsf{sk}, \triangle_1, \triangle_2, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \in T_{\mathsf{KEM}}$ with $(\mathsf{sk}, \mathsf{pk}) \in T_{\mathsf{KG}}$:

replace $\triangle_1$ by pk; replace $\triangle_2$ by $\tilde{\mathsf{r}}$;

return $(\mathsf{c_k}, \mathsf{k}||\mathsf{n})$;

else: $\mathsf{k}||\mathsf{n} \xleftarrow{\$} \mathcal{K} \times \mathcal{N}$; $T_{\mathsf{KEM}} \leftarrow T_{\mathsf{KEM}} \cup \{(\triangle, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n})\}$;

return $(\mathsf{c_k}, \mathsf{k}||\mathsf{n})$;

if pk does not appear in T_{KG} and T_{KEM}:

$b \xleftarrow{\$} \mathrm{Ber}\left(\frac{|\mathcal{SK}|-|\mathsf{Ele\text{-}pk}(T_{\mathsf{KG}} \cup T_{\mathsf{KEM}})|}{|\mathcal{PK}|-|\mathsf{Ele\text{-}pk}(T_{\mathsf{KG}} \cup T_{\mathsf{KEM}} \cup T_{\perp_{\mathsf{pk}}})|}\right)$;

if $b = 0$: $T_{\perp_{\mathsf{pk}}} \leftarrow T_{\perp_{\mathsf{pk}}} \cup \{\mathsf{pk}\}$; **return** $\perp_{\mathsf{pk}}$;

$(\mathsf{c_k}, \mathsf{k}||\mathsf{n}) \xleftarrow{\$} \mathcal{C}_{\mathsf{k}} \times \mathcal{K} \times \mathcal{N}$; $T_{\mathsf{KEM}} \leftarrow T_{\mathsf{KEM}} \cup \{(\triangle, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n})\}$;

return $(\mathsf{c_k}, \mathsf{k}||\mathsf{n})$.

$\underline{\mathcal{S}_{\mathsf{samp}}.\mathbf{IES\text{-}KEM}.\mathsf{Decaps}(\mathsf{sk}, \mathsf{c_k})}$:

if $\exists(\mathsf{sk}, *, *, \mathsf{c_k}, \mathsf{k}||\mathsf{n})$, then **return** $\mathsf{k}||\mathsf{n}$;

if $\exists(\mathsf{sk}, \mathsf{pk}) \in T_{\mathsf{KG}}$ with $(\triangle, \mathsf{pk}, \tilde{\mathsf{r}}, \mathsf{c_k}, \mathsf{k}||\mathsf{n}) \in T_{\mathsf{KEM}}$:

replace $\triangle$ by sk; **return** $\mathsf{k}||\mathsf{n}$;

if $\mathsf{c_k} \in T_{\perp_{\mathsf{c}}}$, then **return** $\perp_{\mathsf{c}}$;

if $\mathsf{c_k}$ does not appear in T_{KEM} and $T_{\mathcal{P}}$:

$b \xleftarrow{\$} \mathrm{Ber}(\frac{|\mathcal{R}|-|\mathsf{Ele\text{-}c_k}(T_{\mathsf{KEM}})|}{|\mathcal{C}_{\mathsf{k}}|-|\mathsf{Ele\text{-}c_k}(T_{\mathsf{KEM}} \cup T_{\perp_{\mathsf{c}}})|})$;

if $b = 0$: $T_{\perp_{\mathsf{c}}} \leftarrow T_{\perp_{\mathsf{c}}} \cup \{\mathsf{c_k}\}$; **return** $\perp_{\mathsf{c}}$;

$\mathsf{k}||\mathsf{n} \xleftarrow{\$} \mathcal{K} \times \mathcal{N}$; $T_{\mathsf{KEM}} \leftarrow T_{\mathsf{KEM}} \cup \{(\mathsf{sk}, \triangle, \triangle, \mathsf{c_k}, \mathsf{k}||\mathsf{n})\}$;

return $\mathsf{k}||\mathsf{n}$;

$\underline{\mathcal{S}_{\mathsf{samp}}.\mathcal{AE}(\mathsf{k}, \mathsf{n}, \mathsf{m})}$:

if $\exists(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}) \in T_{\mathcal{AE}}$, **then return** $\mathsf{c_m}$;

$\mathsf{c_m} \xleftarrow{\$} \mathcal{C}_{\mathsf{m}}$; $T_{\mathcal{AE}} \leftarrow T_{\mathcal{AE}} \cup \{(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m})\}$;

return $\mathsf{c_m}$;

$\underline{\mathcal{S}_{\mathsf{samp}}.\mathcal{AD}(\mathsf{k}, \mathsf{n}, \mathsf{c_m})}$:

if $\exists(\mathsf{k}, \mathsf{n}, \mathsf{m}, \mathsf{c_m}) \in T_{\mathcal{AE}}$, then **return** m;

$T_{\mathcal{AE}} \leftarrow T_{\mathcal{AE}} \cup \{(\mathsf{k}, \mathsf{n}, \perp, \mathsf{c_m})\}$;

return $\perp$;

$\underline{\mathcal{S}_{\mathsf{samp}}.\mathcal{P}(\mathsf{c_k}||\mathsf{c_m})}$:

if $\exists(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}) \in T_{\mathcal{P}}$, then **return** c;

$\mathsf{c} \xleftarrow{\$} \mathcal{C}$; $T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})\}$;

return c;

$\underline{\mathcal{S}_{\mathsf{samp}}.\mathcal{P}^{-1}(\mathsf{c})}$:

if $\exists(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c}) \in T_{\mathcal{P}}$, then **return** $\mathsf{c_k}||\mathsf{c_m}$;

$(\mathsf{c_k}, \mathsf{c_m}) \xleftarrow{\$} \mathcal{C}_{\mathsf{k}} \times \mathcal{C}_{\mathsf{m}}$; $T_{\mathcal{P}} \leftarrow T_{\mathcal{P}} \cup \{(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})\}$;

return $\mathsf{c_k}||\mathsf{c_m}$.

We now show that the views of the differentiator in Game 3 and Game 4 are statistically close. Recall that the only difference between the two games lies in how the simulator answers queries. Thus, the proof reduces to analyzing, for each interface, whether the responses generated by $\mathcal{S}_{\mathsf{samp}}$ faithfully reproduce the distribution enforced by $\mathcal{S}_{\mathsf{fail}}$, and whether any statistical deviation can only occur through rare collision events.

Response from the $\mathcal{H}$ interface. For the $\mathcal{H}$ query, the responses generated by $\mathcal{S}_{\mathsf{fail}}$ and $\mathcal{S}_{\mathsf{samp}}$ are all sampled uniformly from $\mathbb{Z}_N$. Hence, their output distributions are identical.

Response from the **IES-KEM**.KG *interface.* For a query to **IES-KEM**.KG with input sk, the simulator $\mathcal{S}_{\mathsf{samp}}$ generates the response pk by sampling uniformly from $\mathcal{PK}$, rather than invoking **PKE**.KG. Since the public-key generation interface of the ideal PKE is a random injection from $\mathcal{SK}$ to $\mathcal{PK}$, the only bad event is that the sampled pk collides with an entry in either T_{KG} or $T_{\perp_{\mathsf{pk}}}$ (the set of currently marked invalid public keys). The probability of such a collision is at most $q/|\mathcal{PK}|$.

Response from the **IES-KEM**.Encaps *interface.* For a query to **IES-KEM**.Encaps with input $(\mathsf{pk}, \tilde{\mathsf{r}})$, the simulator $\mathcal{S}_{\mathsf{samp}}$ decides whether an unknown public key is valid by sampling from a Bernoulli distribution with probability

$$\frac{|\mathcal{SK}| - |\mathsf{Ele\text{-}pk}(T_{\mathsf{KG}} \cup T_{\mathsf{KEM}})|}{|\mathcal{PK}| - |\mathsf{Ele\text{-}pk}(T_{\mathsf{KG}} \cup T_{\mathsf{KEM}} \cup T_{\perp})|}$$

of being valid, rather than querying **PKE**.Enc. These two approaches yield identical distributions. For valid pk, if there exists no $\mathsf{c_k}$ corresponding to $\tilde{\mathsf{r}}$, the simulator $\mathcal{S}_{\mathsf{samp}}$ samples a $\mathsf{c_k}$ uniformly at random. Assuming no failure condition is triggered, the only bad event is that the sampled $\mathsf{c_k}$ collides with the existing entry in T_{KEM} or $T_{\perp_{\mathsf{c}}}$ (the set of currently marked invalid ciphertexts), which occurs with probability at most $q/|\mathcal{C}_{\mathsf{k}}|$.

Response from the **IES-KEM**.Decaps *interface.* For a query to **IES-KEM**.Decaps with input $(\mathsf{sk}, \mathsf{c_k})$, both $\mathcal{S}_{\mathsf{fail}}$ and $\mathcal{S}_{\mathsf{samp}}$ decide the validity of $\mathsf{c_k}$ by sampling from a Bernoulli distribution with the same probability. If no failure condition is triggered, then for a valid $\mathsf{c_k}$ with no previously associated $\mathsf{k}||\mathsf{n}$, the simulators sample $\mathsf{k}||\mathsf{n}$ uniformly at random. Hence, their output distributions are identical.

Response from the $\mathcal{AE}$ interface. For a query to $\mathcal{AE}$ with input $(\mathsf{k}, \mathsf{n}, \mathsf{m})$, if there exists a corresponding ciphertext c with $(\mathsf{c_k}||\mathsf{c_m}, \mathsf{c})$ recorded in $T_{\mathcal{P}}$ in Game 3, then $\mathcal{S}_{\mathsf{fail}}$ returns c. Provided that no failure condition is triggered, $\mathsf{c_m}$ is sampled uniformly at random in both Game 3 and Game 4. Therefore, the only bad event is that the sampled $\mathsf{c_m}$ collides with an existing ciphertext in $T_{\mathcal{AE}}$, which occurs with probability at most $q/|\mathcal{C}_{\mathsf{m}}|$.

Response from the $\mathcal{AD}$ interface. For a query to $\mathcal{AD}$ with input $(\mathsf{k}, \mathsf{n}, \mathsf{c_m})$, if no corresponding response recorded in $T_{\mathcal{AE}}$, the simulator return $\perp$ directly. Provided that no failure condition is triggered, their output distributions are identical.

Response from the $\mathcal{P}$ interface. For a $\mathcal{P}$ query with input $\mathsf{c_k}||\mathsf{c_m}$, assuming no failure condition is triggered, the only difference is that $\mathcal{S}_{\mathsf{samp}}$ generates the response c by sampling uniformly from $\mathcal{C}$ rather than invoking **PKE**.Enc. Since the **PKE**.Enc is itself a random injection from $\mathcal{PK} \times \mathcal{M} \times \mathcal{R}$ to $\mathcal{C}$, the only bad event here is that the sample c collides with a recorded ciphertext in $T_{\mathcal{P}}$, which occurs with probability at most $q/|\mathcal{C}|$.

Response from the $\mathcal{P}^{-1}$ interface. For a $\mathcal{P}^{-1}$ query with input c, assuming no failure condition is triggered, the only bad event is that the sampled $\mathsf{c_k}||\mathsf{c_m}$ collides with an existing entry in $T_{\mathcal{P}}$, which occurs with probability at most $q/|\mathcal{C}|$.

Therefore, if the simulator $\mathcal{S}_{\mathsf{fail}}$ does not fail in Game 3 and no response generated by $\mathcal{S}_{\mathsf{samp}}$ in Game 4 triggers any of the bad events above, the distinguisher's view remains statistically close between the two games. Let $\mathbf{G}_4$ denote Game 4, which outputs 1 if and only if the differentiator outputs 1 after interacting with Π_{IES} and $\mathcal{S}_{\mathsf{samp}}$. Formally,

$$\Pr[\mathbf{G}_4(\mathcal{D})=1]=\Pr[\mathcal{D}^{\Pi_{\mathsf{IES\text{-}PKE}}^{\mathcal{S}_{\mathsf{samp}}},\mathcal{S}_{\mathsf{samp}}}(1^\lambda)=1].$$

Then,

$$\begin{aligned}|\Pr[\mathbf{G}_4(\mathcal{D})=1]-\Pr[\mathbf{G}_3(\mathcal{D})=1]| &\le O\left(\frac{(\lambda\cdot q)^2}{N}+\frac{(\lambda\cdot q)^2}{2^\lambda}\right)+q\cdot\left(\frac{2q}{2^\ell}+\frac{2q}{2^\lambda}\right)\\ &= O\left(\frac{(\lambda\cdot q)^2}{N}+\frac{(\lambda\cdot q)^2}{2^\lambda}\right).\end{aligned}$$

E.6 Transition from $\mathbf{G}_4$ to Real

Real Game. By replacing the simulator $\mathcal{S}_{\mathsf{samp}}$ in Game 4 with the real idealized oracles

$$\mathcal{O}=(\mathcal{H},\mathbf{IES\text{-}KEM},\mathcal{AE},\mathcal{AD},\mathcal{P},\mathcal{P}^{-1}),$$

we arrive at the final game of our proof, which we refer to as the real game. In this game, the differentiator $\mathcal{D}$ interacts with $\Pi_{\mathsf{IES\text{-}PKE}}^{\mathcal{O}}$ and $\mathcal{O}$. The real game outputs 1 if and only if $\mathcal{D}$ outputs 1. Formally,

$$\Pr[\mathbf{Real}^{\Pi_{\mathsf{IES\text{-}PKE}}^{\mathcal{O}}}(\mathcal{D})=1]=\Pr[\mathcal{D}^{\Pi_{\mathsf{IES\text{-}PKE}}^{\mathcal{O}},\mathcal{O}}(1^\lambda)=1].$$

We now demonstrate that the views of the differentiator in Game 4 and the Real Game are statistically close.

For the random oracle query $\mathcal{H}$, the responses in Game 4 and Real Game are uniformly distributed.

For the public-key generation query $\mathbf{IES\text{-}KEM}.\mathsf{KG}$, the differentiator may observe a difference if the uniformly sampled pk collides with the existing one in T_{KG} or $T_{\perp_{\mathsf{pk}}}$, each with probability at most $q/|\mathcal{PK}|$. For the encapsulation query $\mathbf{IES\text{-}KEM}.\mathsf{Encaps}$, the simulator in Game 4 determines the validity of public key via the Bernoulli distribution and $T_{\perp_{\mathsf{pk}}}$. Specifically, an unknown public key is deemed valid with probability

$$\frac{|\mathcal{SK}|-|\mathsf{Ele\text{-}pk}(T_{\mathsf{KG}}\cup T_{\mathsf{KEM}})|}{|\mathcal{PK}|-|\mathsf{Ele\text{-}pk}(T_{\mathsf{KG}}\cup T_{\mathsf{KEM}}\cup T_{\perp_{\mathsf{pk}}})|}.$$

Since T_{KG}, T_{KEM} and $T_{\perp_{\mathsf{pk}}}$ contain all valid and invalid public keys known to the differentiator, the views in the two games are identical. For valid input with no corresponding response determined, the simulator in Game 4 samples $\mathsf{c_k}$ and $\mathsf{k}||\mathsf{n}$ uniformly, and the differentiator may observe a difference if such $\mathsf{c_k}$ collides with the existing one in T_{KEM} or $T_{\perp_{\mathsf{c}}}$, each with probability at most $q/|\mathcal{C}_{\mathsf{k}}|$. For the decapsulation query $\mathbf{IES\text{-}KEM}.\mathsf{Decaps}$, apparently the views in the two games are identical.

For the $\mathcal{AE}$ query, the differentiator may observe a difference if the uniformly sample $\mathsf{c_m}$ by $\mathcal{S}_{\mathsf{samp}}$ collides with the existing one in $T_{\mathcal{AE}}$, each with probability at most $q/|\mathcal{C}_{\mathsf{m}}|$. For the $\mathcal{AD}$ query, note that the simulator in Game 4 always returns $\perp$ if no response recorded. The differentiator may observe a difference if it generate a pair $(\mathsf{k},\mathsf{n},\mathsf{c_m})$ in the ideal game that can be decrypted, the probability of which is at most $|\mathcal{M}|/|\mathcal{C}_{\mathsf{m}}|\le 1/2^\lambda$.

For the random permutation queries $\mathcal{P}$ and $\mathcal{P}^{-1}$, the differentiator may observe a difference if the uniformly sampled response by $\mathcal{S}_{\mathsf{samp}}$ collides with the existing one in $T_{\mathcal{P}}$, each with probability at most $q/|\mathcal{C}|$.

Therefore, we deduce that

$$\left|\Pr[\mathbf{Real}^{\Pi^{\mathcal{O}}_{\mathsf{IES}}}(\mathcal{D})=1]-\Pr[\mathbf{G}_4(\mathcal{D})=1]\right| \leq O\left(\frac{q^2}{2^\lambda}\right).$$

Combining all the games above, we conclude that:

$$\left|\Pr[\mathbf{Real}^{\Pi^{\mathcal{O}}_{\mathsf{IES\text{-}PKE}}}(\mathcal{D})=1]-\Pr[\mathbf{Ideal}^{\mathbf{PKE}}_{\mathcal{S}}(\mathcal{D})=1]\right| \leq O\left(\frac{(\lambda\cdot q)^2}{N}+\frac{(\lambda\cdot q)^2}{2^\lambda}\right).$$

□

E.7 Bounding the failure conditions

Note that the simulator $\mathcal{S}_{\mathsf{fail}}$ in Games 2 and 3 may fail under various conditions. In this section, we analyze the probability of each such failure occurring.

Lemma E.1. $\Pr$ [Failure Conditions Hold] $\leq O\left(q^2/N + q^2/2^\lambda\right)$.

Proof. As described in Appendix E.2, the simulator evaluates potential failure conditions whenever it handles a $\mathcal{H}$, **IES-KEM**.Encaps, **IES-KEM**.Decaps, $\mathcal{AE}$, $\mathcal{P}$ or $\mathcal{P}^{-1}$ query.

When processing a $\mathcal{H}$ query on input $(\mathsf{pk},\mathsf{m},\mathsf{r})$, the simulator samples $\tilde{\mathsf{r}} \in \mathcal{R}$ uniformly at random as the candidate response. Two distinct failure conditions arise:

Condition $\mathsf{C}_{1\text{-}\mathcal{H}}$. The simulator fails if the sampled $\tilde{\mathsf{r}}$ already exists in T_{KEM} or $T_{\mathcal{H}}$. Since the simulator makes at most q queries to $\mathcal{S}_{\mathsf{fail}}$, each table maintained by $\mathcal{S}_{\mathsf{fail}}$ contains at most q entries, meaning there are at most q randomness recorded. Therefore, the probability that a uniformly sampled $\tilde{\mathsf{r}}$ collides with one of these randomness is bounded by

$$\Pr[\mathsf{C}_{1\text{-}\mathcal{H}}] \leq \frac{q}{|\mathcal{R}|} = \frac{q}{N}.$$

Condition $\mathsf{C}_{2\text{-}\mathcal{H}}$. The simulator fails if the differentiator queries with some r such that r already appears in T^*_{r}. Since each randomness in T^*_{r} is sampled uniformly at random and remains hidden from the differentiator, the probability of this event is bounded by

$$\Pr[\mathsf{C}_{2\text{-}\mathcal{H}_x}] \leq \frac{q}{|\mathcal{R}|} = \frac{q}{N}.$$

When processing a **IES-KEM**.Encaps query on input $(\mathsf{pk},\tilde{\mathsf{r}})$, in some cases the simulator samples $\mathsf{k}||\mathsf{n} \in \mathcal{K}\times\mathcal{N}$ and/or $\mathsf{c}_\mathsf{k} \in \mathcal{C}_\mathsf{k}$ uniformly at random as the candidate response. Two distinct failure conditions arise:

Condition $\mathsf{C}_{1\text{-}\mathsf{Encaps}}$. The simulator fails if the sampled $\mathsf{k}||\mathsf{n}$ already appears in T_{KEM} or $T_{\mathcal{AE}}$. As noted previously, each table maintained by $\mathcal{S}_{\mathsf{fail}}$ contains at most q entries. Therefore, the probability is bounded by

$$\Pr[\mathsf{C}_{1\text{-}\mathsf{Encaps}}] \leq \frac{q}{|\mathcal{K}\times\mathcal{N}|} = \frac{q}{2^{2\lambda}}.$$

Condition $\mathsf{C}_{2\text{-}\mathsf{Encaps}}$. The simulator fails if the sampled c_k already appears in T_{KEM} or $T_\perp$. The probability is bounded by

$$\Pr[\mathsf{C}_{2\text{-}\mathsf{Encaps}}] \leq \frac{q}{|\mathcal{C}_\mathsf{k}|} = \frac{q}{2^\ell}.$$

When processing a **IES-KEM**.Decaps query on input $(\mathsf{sk},\mathsf{c}_\mathsf{k})$, in some cases the simulator samples $\mathsf{k}||\mathsf{n} \in \mathcal{K}\times\mathcal{N}$ uniformly at random as the candidate response. Two distinct failure conditions arise:

Condition $\mathsf{C}_{1\text{-}\mathsf{Decaps}}$. The simulator fails if the sampled $\mathsf{k}||\mathsf{n}$ already appears in T_{KEM} or $T_{\mathcal{AE}}$. The probability is bounded by

$$\Pr[\mathsf{C}_{1\text{-}\mathsf{Decaps}}] \leq \frac{q}{|\mathcal{K}\times\mathcal{N}|} = \frac{q}{2^{2\lambda}}.$$

Condition $\mathsf{C}_{2\text{-}\mathsf{Decaps}}$. The simulator fails if the differentiator queries with some sk such that sk already appears in T^*_{sk}. Since each private key in T^*_{sk} is sampled uniformly at random and remains hidden from the differentiator, the probability of this event is bounded by

$$\Pr[\mathsf{C}_{2\text{-}\mathsf{Decaps}}] \leq \frac{q}{|\mathcal{SK}|} = \frac{q}{N}.$$

When processing a $\mathcal{AE}$ query on input $(\mathsf{k}, \mathsf{n}, \mathsf{m})$, in some cases the simulator samples $\mathsf{c_m} \in \mathcal{C}_{\mathsf{m}}$ uniformly at random as the candidate response. One failure condition arises:

Condition $\mathsf{C}_{1\text{-}\mathcal{AE}}$. The simulator fails if the sampled $\mathsf{c_m}$ already appears in $T_{\mathcal{P}}$ or $T_{\mathcal{AE}}$. The probability is bounded by

$$\Pr[\mathsf{C}_{1\text{-}\mathsf{Decaps}}] \leq \frac{q}{|\mathcal{C}_{\mathsf{m}}|} = \frac{q}{2^{\lambda}}.$$

When processing a $\mathcal{P}$ query on input $\mathsf{c_k}||\mathsf{c_m}$, the simulator first checks the validity of $\mathsf{c_k}$. If $\mathsf{c_k}$ is regarded as invalid, the simulator samples an invalid ciphertext c as the candidate response. Two failure conditions arises:

Condition $\mathsf{C}_{1\text{-}\mathcal{P}}$. The simulator fails if the sampled c already appears in $T_{\mathcal{P}}$. The probability is bounded by

$$\Pr[\mathsf{C}_{1\text{-}\mathcal{P}}] \leq \frac{q}{|\mathcal{C}|} \leq \frac{q}{2^{\lambda}}.$$

Condition $\mathsf{C}_{2\text{-}\mathcal{P}}$. The simulator fails if, even after λ iterations, the uniformly sampled ciphertext c remains valid. The probability is bounded by

$$\Pr\left[\mathsf{C}_{2\text{-}\mathcal{P}}\right] \leq \left(\frac{|\mathcal{R}| \cdot |\mathcal{C}_{\mathsf{m}}|}{|\mathcal{C}|}\right)^{\lambda} \leq \frac{1}{2^{\lambda}}.$$

If $\mathsf{c_k}$ and $\mathsf{c_m}$ are generated through Encaps and $\mathcal{AE}$ respectively, and correspond to the same pk and m, then the response is the ciphertext under $(\mathsf{pk}, \mathsf{m})$. If no corresponding randomness r exists, the simulator generates a fresh ciphertext c as the candidate response by sampling r uniformly at random, and two failure conditions arises:

Condition $\mathsf{C}_{3\text{-}\mathcal{P}}$. The simulator fails if the sampled r already appears in $T_{\mathcal{H}}$ or T^*_{r}. The probability is bounded by

$$\Pr[\mathsf{C}_{3\text{-}\mathcal{P}}] \leq \frac{q}{|\mathcal{R}|} = \frac{q}{N}.$$

Condition $\mathsf{C}_{4\text{-}\mathcal{P}}$. The simulator fails if the generated c already appears in $T_{\mathcal{P}}$. The probability is bounded by

$$\Pr[\mathsf{C}_{4\text{-}\mathcal{P}}] \leq \frac{q}{|\mathcal{R}|} = \frac{q}{N}.$$

Otherwise, the simulator determines the validity of an unknown $\mathsf{c_k}$ by sampling from a Bernoulli distribution. If $\mathsf{c_k}$ is regarded as invalid, the simulator samples an invalid ciphertext c as the candidate response and then checks the failure conditions $\mathsf{C}_{1\text{-}\mathcal{P}}(\mathsf{c})$ and $\mathsf{C}_{2\text{-}\mathcal{P}}(\mathsf{c})$. Otherwise, for valid $\mathsf{c_k}$, the simulator generates a fresh ciphertext c as the candidate response by sampling sk, m, and r uniformly at random, and two failure conditions arises:

Condition $\mathsf{C}_{5\text{-}\mathcal{P}}$. The simulator fails if the sampled r already appears in $T_{\mathcal{H}}$ or T^*_{r}. The probability is bounded by

$$\Pr[\mathsf{C}_{5\text{-}\mathcal{P}}] \leq \frac{q}{|\mathcal{R}|} = \frac{q}{N}.$$

Condition $\mathsf{C}_{6\text{-}\mathcal{P}}$. The simulator fails if the sampled sk already appears in T_{KEM}. The probability is bounded by

$$\Pr[\mathsf{C}_{6\text{-}\mathcal{P}}] \leq \frac{q}{|\mathcal{SK}|} = \frac{q}{N}.$$

When processing a $\mathcal{P}^{-1}$ query with input c, the simulator first checks the validity of c using $\mathbf{PKE}$.Dec. If c is invalid, the simulator samples $\mathsf{c_k}$ and $\mathsf{c_m}$ uniformly at random as the candidate response, and two distinct failure conditions arises:

Condition $\mathsf{C}_{1\text{-}\mathcal{P}^{-1}}$. The simulator fails if the sampled $\mathsf{c_m}$ already appears in $T_{\mathcal{P}}$ or $T_{\mathcal{AE}}$. The probability is bounded by

$$\Pr[\mathsf{C}_{1\text{-}\mathcal{P}^{-1}}] \leq \frac{q}{|\mathcal{C}_{\mathsf{m}}|} = \frac{q}{2^{\lambda}}.$$

Condition $\mathsf{C}_{2\text{-}\mathcal{P}^{-1}}$. The simulator fails if the sampled $\mathsf{c_k}$ already appears in T_{KEM}. The probability is bounded by

$$\Pr[\mathsf{C}_{2\text{-}\mathcal{P}^{-1}}] \leq \frac{q}{|\mathcal{C}_{\mathsf{k}}|} = \frac{q}{2^{\ell}}.$$

If c is valid and there exists $(\mathsf{pk}||\mathsf{m}||\mathsf{r}, \tilde{\mathsf{r}})$ such that $(\mathsf{pk}, \mathsf{m}, \mathsf{r})$ corresponds to c and $\tilde{\mathsf{r}}$ corresponds to $\mathsf{c_k}$, the simulator samples $\mathsf{c_m}$ uniformly at random and one failure condition arises:

Condition $\mathsf{C}_{3\text{-}\mathcal{P}^{-1}}$. The simulator fails if the sampled $\mathsf{c_m}$ already appears in $T_{\perp}$. The probability is bounded by

$$\Pr[\mathsf{C}_{3\text{-}\mathcal{P}^{-1}}] \leq \frac{q}{|\mathcal{C}_{\mathsf{m}}|} = \frac{q}{2^{\lambda}}.$$

Combine together, we have

$$\begin{aligned}\Pr\left[\text{Failure Conditions Hold}\right] &\leq q \cdot \left(\frac{7q}{N} + \frac{4q+1}{2^{\lambda}} + \frac{2q}{2^{\ell}} + \frac{2q}{2^{2\lambda}}\right) \\ &= O\left(\frac{q^2}{N} + \frac{q^2}{2^{\lambda}}\right).\end{aligned}$$

□
□