Designs, Codes and Cryptography (2020) 88:2433–2452
https://doi.org/10.1007/s10623-020-00794-z

Tightly CCA-secure encryption scheme in a multi-user setting with corruptions

Youngkyung Lee[1] · Dong Hoon Lee[1] · Jong Hwan Park[2]

Received: 20 February 2019 / Revised: 25 March 2020 / Accepted: 19 August 2020 /
Published online: 2 September 2020

Abstract
The security of public-key encryption (PKE) schemes in a multi-user setting is aimed at capturing real-world scenarios in which an adversary could attack multiple users and multiple ciphertexts of its choice. However, the fact that a real-world adversary can also mount key-exposure attacks for a set of multiple public keys requires us to consider a more realistic notion of security in multi-user settings. In this study, we establish the security notion of PKE in a multi-user setting *with corruptions*, where an adversary is able to issue (adaptive) encryption, decryption, and corruption (i.e., private key) queries. We then propose the first practical PKE scheme whose security is proven in a multi-user setting with corruptions. The security of our scheme is based on the computational Diffie–Hellman (CDH) assumption and is proven to be tightly chosen-ciphertext secure in a random oracle model. Our scheme essentially follows the recently proposed modular approach of combining KEM and augmented DEM in a multi-user setting, but we show that this modular approach works well in a multi-user setting with corruptions.

Keywords Public-key encryption · Multi-user setting with corruptions · CDH · Tight security · Random oracle model

Mathematics Subject Classification 94A60

Communicated by R. Steinfeld.

✉ Jong Hwan Park
jhpark@smu.ac.kr

Youngkyung Lee
dudrudve@korea.ac.kr

Dong Hoon Lee
donghlee@korea.ac.kr

1 Korea University, Seoul, Korea

2 Sangmyung University, Seoul, Korea

1 Introduction

The security of most public-key encryption (PKE) schemes is analyzed using security models that reflect real-world attack environments as closely as possible. The standard security model for PKE schemes has been formalized as the indistinguishability against adaptive chosen-ciphertext attacks model (denoted as 'IND-CCA' security [5,12,26,27]), where a single user and a single ciphertext become targets to an adversary. However, the IND-CCA security model is still lacking for fully reflecting realistic scenarios because a real-world adversary can try to attack multiple users and multiple ciphertexts of their choice. To narrow the gap between the IND-CCA security model and real-world scenarios, Bellare et al. [6] proposed an IND-CCA security model in a *multi-user* setting (hereafter denoted as 'IND-CCA-MUC' or simply the 'MUC' model), where multiple users and multiple ciphertexts become targets to an adversary. In particular, the MUC model captures even attack scenarios in which an attacker obtains *related* messages encrypted using different public keys. Recently, many studies [2,14,15,18,20–22,24,25] have focused on designing new PKE schemes that are proven secure in the MUC model [6].

Another consideration to make when constructing PKE schemes is to provide tight security reductions, which involve proving the security of the scheme based on certain computational hardness assumptions. In general, a security reduction demonstrates that if a computational problem is hard to solve with probability $2^{-\lambda}$, then any adversary in a security model can break such a PKE scheme with a probability of at most $L \cdot 2^{-\lambda}$, where λ is a security parameter and L is a factor of security loss. If L is a constant, the hardness of a computational assumption is *tightly* transformed into the security of a PKE scheme at the same security level. On the other hand, if $L = 2^{30}$ for instance, then any adversary can break such a PKE scheme with a probability of at most $2^{30} \cdot 2^{-\lambda}$, which means that the security parameter λ should be increased to approximately $\lambda + 30$ to achieve the same level of security. Because increasing the security parameter results in a significant decrease in efficiency, it is important to achieve tight security reductions in a practical sense. Moreover, there is no doubt that computational assumptions should be believed to be standard, i.e., well-established as cryptographic complexity assumptions, such as the discrete logarithm and the computational Diffie-Hellman (CDH) assumptions [4,11].

In light of the above considerations, many PKE schemes [2,14,15,18,20–22,24,25] have been proposed to provide (almost[1]) tight security reductions in the MUC model. Though most of them provide (almost) tight security reductions without using random oracles, these schemes are generally considered inefficient for practical use. More concretely, some these PKE schemes [21,24,25] are constructed using a variant of the Naor–Yung (NY) double-encryption paradigm [26] and the Groth–Sahai [19] non-interactive zero-knowledge (NIZK) proofs, so they require quite a large number of pairing operations during decryption. In addition, for a security parameter λ, some have the drawbacks of requiring $O(\lambda)$ ciphertexts [21] or $O(\lambda)$ public keys [24,25]. Two of these PKE schemes are obtained by applying the Boneh–Canetti–Halevi–Katz transform [8] to identity-based encryption (IBE) schemes [2,22] that are almost tightly secure in a multi-challenge and multi-instance[2] setting. The resultant PKE constructions [2,22] are all based on bit-by-bit encoding methods for handling elements corresponding to $O(\lambda)$-bit identities, resulting in $O(\lambda)$-long public keys. Recently, Gay et al. [14] presented a PKE scheme with an almost tight security reduction in pairing-free

[1] "Almost" tightness means that the security loss L is determined by $O(\lambda)$ for a security parameter λ.

[2] "Multi-instance" means that there exist multiple key generation centers, which correspond to multiple users in the process of converting from IBE to PKE.

groups, but their scheme still suffers from having $O(\lambda)$ public keys. More recently, Gay et al. [15] reduced the size of the public keys to a constant size. However, their security reduction is almost tight, which means that a security loss still occurs with a factor of $O(\lambda)$.

1.1 Motivation

Although the MUC model has been considered to capture real attack scenarios in which an adversary tries to attack multiple users and multiple ciphertexts of its choice, the security guarantee of the MUC model is still insufficient to reflect attacks that can occur in reality. This insufficiency comes from not taking the corruption of private keys into account as a possible attack even though many public keys are given to the adversary. Namely, in the MUC model, a set of public keys $\{\mathsf{PK}_1, \ldots, \mathsf{PK}_n\}$ for a positive integer n are given to the adversary but, without allowing for the corruption of some private keys corresponding to the public keys, the adversary must rely on chosen-ciphertext attacks. However, given the fact that cryptographic computations (such as decryption using private keys) are sometimes performed on weakly protected devices, the adversary can successfully expose some private keys. This issue requires us to strengthen the MUC model by including so-called corruption (i.e., private key) queries on some of the public keys. We denote the strengthened MUC model as the MUC^+ model, which refers to the IND-CCA security model in a multi-user setting *with corruptions*. Such private key queries in a multi-user setting can also be justified by the recent works [3,17], which suggests a practical and tightly-secure signature scheme in a multi-user setting with corruptions. Similarly, the standard notion of IBE security [7,28] offers corruption queries for identities (used as public keys) that an adversary can choose adaptively.

The more realistic MUC^+ model raises the following question: can we construct a practical and tightly secure PKE scheme in the MUC^+ model? It is known that using random oracles makes it easy to design such a PKE scheme in the MUC model. For instance, one can obtain the ElGamal encryption scheme, which is proven to be tightly secure against chosen-plaintext attacks (denoted as 'IND-CPA') in the MUC model, based on the decisional Diffie–Hellman (DDH) problem and its random self-reducibility [6]. Next, by applying the tight variant [23] of the Fujisaki–Okamoto (FO) transform [13] to the IND-CPA-secure ElGamal scheme, one can achieve a practical PKE scheme with a tight security reduction in the MUC model. This is the reason why there has not been enough research to find practical and tightly secure PKE schemes in random-oracle models. However, the same approach does not hold for constructing a secure PKE scheme in the MUC^+ model because handling corruption queries on an arbitrary subset of public keys $\{\mathsf{PK}_1, \ldots, \mathsf{PK}_n\}$ (by guessing a subset of public keys in advance) results in a poor security reduction, even for somewhat large values of n, e.g., $n = 2^{10}$. This shows that a new technique may be required when using random oracles to construct an tightly IND-CCA-secure PKE scheme in the MUC^+ model.

1.2 Our contributions

We propose the first practical PKE scheme with tight security reduction in the MUC^+ model, following the modular approaches of the KEM (key encapsulation mechanism) and augmented DEM (data encapsulation mechanism) frameworks in a multi-user setting [16]. In terms of efficiency, our scheme offers a $O(1)$-sized public key that contains four group elements and $O(1)$-sized ciphertext that consists of two hash outputs plus one group element. In addition, KEM encryption and decryption operations each require five group exponentiations.

Table 1 Comparison between previous PKE schemes and ours

Scheme	$\lvert\mathsf{PK}\rvert$	$\lvert\mathsf{CT}\rvert$ - $\lvert\mathsf{M}\rvert$	Sec. loss	Sec. model	Sec. assump.	ROM/ STD	Pairing
CS98 [10]	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(q_e)$	MUC	DDH	STD	YES
HJ12 [21]	$\mathcal{O}(1)$	$\mathcal{O}(\lambda)$	$\mathcal{O}(1)$	MUC	2-LIN	STD	YES
LPJY15 [24,25]	$\mathcal{O}(\lambda)$	$\mathcal{O}(1)$	$\mathcal{O}(\lambda)$	MUC	2-LIN	STD	YES
GCDCT15 [18]	$\mathcal{O}(\lambda)$	$\mathcal{O}(k)$	$\mathcal{O}(\lambda)$	MUC	k-LIN	STD	YES
AHY15 [2]	$\mathcal{O}(\lambda)$	$\mathcal{O}(1)$	$\mathcal{O}(\lambda)$	MUC	2-LIN	STD	YES
HKS15 [22]	$\mathcal{O}(\lambda)$	$\mathcal{O}(1)$	$\mathcal{O}(\lambda)$	MUC	k-LIN	STD	YES
GHKW16 [14]	$\mathcal{O}(\lambda)$	$\mathcal{O}(1)$	$\mathcal{O}(\lambda)$	MUC	DDH	STD	NO
H16 [20]	$\mathcal{O}(\lambda k^2)$	$\mathcal{O}(k)$	$\mathcal{O}(\lambda)$	MUC	k-LIN	STD	YES
GHK17 [15]	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(\lambda)$	MUC	DDH	STD	NO
Ours	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	MUC^+	CDH	ROM	NO

$\circ$ λ is the security parameter. $\circ$ q_e is the maximum number of challenge queries per public key. $\circ$ MUC denotes the IND-CCA security model in a *multi-user* setting. $\circ$ MUC^+ denotes the IND-CCA security model in a *multi-user* setting with corruptions. $\circ$ ROM denotes random oracle models. $\circ$ STD denotes standard models (i.e., without random oracles). $\circ$ DDH denotes the decisional Diffie–Hellman assumption. $\circ$ CDH denotes the computational Diffie–Hellman assumption. $\circ$ DLIN denotes the decision linear assumption. $\circ$ $\lvert\mathsf{PK}\rvert$ denotes the size of the public key in groups elements. $\circ$ $\lvert\mathsf{CT}\rvert$ - $\lvert\mathsf{M}\rvert$ denotes the size of the ciphertext in groups elements, ignoring smaller components from symmetric encryption

In particular, because our scheme works with pairing-free groups, those group operations can be easily performed using elliptic-curve cryptography (ECC) algorithms. In terms of security, we first provide a formal definition of KEM with respect to the MUC^+ model where, in addition to multiple encapsulation and decapsulation queries, corruption (i.e., private key) queries are permitted given a set of public keys $\{\mathsf{PK}_1, \ldots, \mathsf{PK}_n\}$. Next, we suggest a modified version of augmented hybrid encryption by combining a KEM approach (secure in the MUC^+ model) and an augmented DEM approach (secure in a multi-instance setting) and show that the resulting PKE scheme achieves a tight security reduction in the MUC^+ model. Interestingly, augmented DEM (without allowing for corruption queries in its security) suffices for the security of the MUC^+ model because corruption queries involve key-exposure attacks only in KEM. Under the abovementioned security frameworks, we prove that our KEM scheme is tightly secure in the MUC^+ model using random oracles based on the CDH assumption. Table 1 presents a comparison between previous (almost) tightly secure PKE schemes and our scheme in terms of security and efficiency.

1.2.1 Our approach

To achieve tight security reductions in the MUC^+ model, we first need to consider a technique to handle corruption queries without causing a non-tight security reduction. To do this, we employ the idea of the NY double-encryption paradigm (even in a random oracle model), by which a simulator in our security analysis can answer any corruption queries with respect to each of the public keys $\{\mathsf{PK}_1, \ldots, \mathsf{PK}_n\}$. More precisely, each PK_i consists of two public keys $(\mathsf{pk}_{0,i}, \mathsf{pk}_{1,i})$ and thus, for a randomly chosen bit $b \in \{0, 1\}$, only $\mathsf{sk}_{b,i}$ is the corresponding private key. In this setting, the simulator knows only one of the private keys, $\mathsf{sk}_{b,i}$, which can be given as the answer to the corruption query, whereas the other private key $\mathsf{sk}_{1-b,i}$ is unknown to the simulator. Using the encryption algorithm enc of a PKE scheme under

$\mathsf{pk}_{0,i}$ and $\mathsf{pk}_{1,i}$, our KEM scheme encrypts the same message R (randomly chosen in a message space) and generates a ciphertext $\mathsf{CT} = (C_0, C_1)$, where $C_0 = \mathsf{enc}(\mathsf{pk}_{0,i}, R)$ and $C_1 = \mathsf{enc}(\mathsf{pk}_{1,i}, R)$. Given the ciphertext $\mathsf{CT} = (C_0, C_1)$ generated in the abovementioned way, the simulator can decrypt C_b using $\mathsf{sk}_{b,i}$ but cannot decrypt the other part C_{1-b}. This makes it hard for the simulator to check the well-formedness of C_{1-b} and thus deal with (adversarial) decapsulation queries. To solve this problem, the NY technique generates the NIZK proof π to prove that C_0 and C_1 encrypt the same message R and appends proof π to the ciphertext.

On the other hand, using random oracles instead of the generally inefficient NIZK proof makes it easier and simpler to solve the problem without increasing the size of ciphertexts. Let H and H' be hash functions modeled as random oracles. Our idea is to use the KEM variant of the FO transform [1] to check the well-formedness of the other part, C_{1-b}. For the same encrypted message R, the ciphertext is constructed as $\mathsf{CT} = (C_0, C_1)$, where $C_0 = \mathsf{enc}(\mathsf{pk}_{0,i}, R; H(R))$ and $C_1 = \mathsf{enc}(\mathsf{pk}_{1,i}, R; H(R))$ for the same randomness $H(R)$. In this case, the simulator can obtain R by decrypting C_b and check if C_{1-b} was generated correctly through the simple re-encryption of R under the randomness $H(R)$. That is, the randomness recovery property of the FO transform makes it very simple to check in the NY paradigm whether both C_0 and C_1 encrypt the same message under the same randomness. Under the ciphertext CT generated as explained above, the relevant KEM key is computed as $\mathsf{K} = H'(R)$. Because the adversary is asked to query R to oracle H each time it encrypts message R, the queried R allows the simulator to know beforehand what the KEM key K corresponding to ciphertext CT will be. This shows how the simulator is able to handle decapsulation queries in the above (NY+FO)-combined technique.

Given a challenge ciphertext $\mathsf{CT}^* = (C_0^*, C_1^*)$ for a challenged $\mathsf{PK}_i^* = (\mathsf{pk}_b^*, \mathsf{pk}_{1-b}^*)$ (which can be easily extended to a multi-user setting), the adversary must decrypt either C_0^* or C_1^* to recover the encrypted message R^* (and then the KEM key $\mathsf{K}^* = H'(R^*)$). We now have a simulator that can decrypt C_b^* using $\mathsf{sk}_{b,i}^*$ but not C_{1-b}^*, hoping that the adversary can decrypt C_{1-b}^*. If the adversary does not know bit b for private key $\mathsf{sk}_{b,i}^*$, which has not been corrupted, we can expect that the adversary will decrypt C_{1-b}^* with a probability of essentially 1/2. Hence, the simulator can use the ability of the adversary to break the PKE scheme under $\mathsf{pk}_{1-b,i}^*$ with the same probability of 1/2. This observation shows that it is necessary to prove that bit b for $\mathsf{sk}_{b,i}^*$ is not revealed to the adversary in our security proofs. However, this hiding is not enough to prove that the (NY+FO)-combined KEM is tightly secure in a multi-user setting, especially, *under the CDH assumption*. Given a set of challenged public keys, the simulator should be able to check which public key among the challenged set is the target for the adversary to succeed, because a distinct instance of the CDH problem is associated with each (challenged) public key, respectively. Otherwise, the simulator choose a target public key among the set of challenged public keys, which also causes $\mathcal{O}(n)$ security loss. To solve these issues, we use a PKE scheme as a building block (regarding $\mathsf{pk}_{0,i}$ and $\mathsf{pk}_{1,i}$), which is a slight variant of the twin ElGamal encryption scheme [9]. Our variant[3] can be proven to be tightly one-way-secure against chosen-plaintext attacks (denoted as '$\mathsf{OW\text{-}CPA}$') based on the twin Diffie–Hellman (TDH) assumption [9] (and thus the CDH assumption), which has an access to a decisional TDH oracle as a checking mechanism. Therefore, breaking the PKE scheme under $\mathsf{pk}_{1-b,i}^*$ with probability ε allows the simulator to solve an instance of the TDH (and thus the CDH) problem with probability $\varepsilon/2$. Finally, the random self-reducibility

[3] Roughly speaking, our variant works as follows: for a public key $(g, X_1 = g^{x_1}, X_2 = g^{x_2})$ with a corresponding secret key (x_1, x_2), encryption is done by computing (g^s, X_1^s, X_2^s) for a random s and outputting a ciphertext $(g^s, H(g^s, X_1^s, X_2^s) \oplus m)$ for a message m.

of the TDH assumption implies that the results in single-user settings can be easily extended to multi-user settings without causing any security loss.

In 2015, Bader et al. [3] showed that using the NY double encryption technique, the generic KEM based on any $\mathsf{IND\text{-}CPA\text{-}MUC}$-secure PKE scheme can be proven tightly secure in the $\mathsf{IND\text{-}CPA\text{-}MUC}^+$ model. Their construction idea is almost similar to the above mentioned one, in that each public key PK consists of two public keys $(\mathsf{pk}_0, \mathsf{pk}_1)$ and only one sk_b is generated for a randomly chosen $b \in \{0, 1\}$. In that case, one might think that it is generically easy to build a KEM (tightly) secure in the $\mathsf{IND\text{-}CCA\text{-}MUC}^+$ model from a KEM (or PKE scheme) secure in the $\mathsf{IND\text{-}CPA\text{-}MUC}^+$ model, using the FO transform as usual. Our answer is no, and the reason is from the *indistinguishability* of the underlying PKE scheme. Our argument is as follows: for a challenged public key $\mathsf{PK}^* = (\mathsf{pk}_0^*, \mathsf{pk}_1^*)$, a challenge ciphertext is constructed as $\mathsf{CT}^* = (C_0^*, C_1^*)$ on a message m^*, where C_b^* is an encryption of m^* generated by the simulator itself and C_{1-b}^* is the challenge ciphertext that the simulator is given in response to the same m^*. Thus, C_{1-b}^* could be an encryption of either m^* or a random (unknown) message R, and the goal of the simulator is to correctly guess which one of the two cases, using the ability of the adversary. When applying the FO transform, C_b^* is computed as $C_b^* = \mathsf{enc}(\mathsf{pk}_b^*, m^*; H(b, m^*))$. In case of a single challenge ciphertext, it is easy to see that the simulator can use the adversary's ability straightforwardly by checking if (b, m^*) is queried to H oracle. This is because making such a query related to m^* is the only way that the adversary distinguishes between a real game and the simulation game. However, in case of $\mathsf{CT}^* = (C_0^*, C_1^*)$, the adversary can distinguish between the two games, regardless of whether C_{1-b}^* is the encryption of m^* or R. Firstly, when $C_{1-b}^* = \mathsf{enc}(\mathsf{pk}_{1-b}^*, m^*; r^*)$ for some randomness r^*, the adversary recovers m^* from either C_b^* or C_{1-b}^* and issues H-oracle queries on inputs (b, m^*) and $(1-b, m^*)$. The point is that even though those oracle queries related to m^* are made, the simulator does not know which ciphertext among (C_0^*, C_1^*) is decrypted by the adversary. Moreover, the simulator cannot answer the H-oracle query on the input $(1 - b, m^*)$ unless r^* is known to the simulator, which allows the adversary to differentiate between the two games. Secondly, when $C_{1-b}^* = \mathsf{enc}(\mathsf{pk}_{1-b}^*, R; r^*)$, the adversary is able to recover both m^* from C_b^* and R from C_{1-b}^*, in which case the adversary immediately can see that it is under simulation. If the adversary recovers either m^* from C_b^* or R from C_{1-b}^*, the simulator cannot reply to H-oracle queries on inputs $(1-b, m^*)$, (b, R), and $(1 - b, R)$ consistently. As a result, in any case, the adversary can distinguish the two games. On the other hand, the idea behind our construction is that we rely the security of the underlying PKE scheme on the computational hardness (rather than indistinguishability) of the TDH problem, and find a correct answer with respect to a TDH instance, using H-oracle queries and the decisional TDH oracle.

2 Background

2.1 Notation

We write $\lambda \in \mathbb{N}$ for the security parameter. We say that a function $\nu : \mathbb{N} \to \mathbb{R}$ is *negligible* if, for every positive polynomial $\mathsf{poly}(\cdot)$, there exists an integer $N > 0$ such that for all $x > N$ it holds that $|\nu(x)| < \mathsf{poly}(x)^{-1}$. Given an algorithm A, we write $y \leftarrow A$ to denote that y is the output of A. If A is a probabilistic algorithm, then $y \overset{\$}{\leftarrow} A$ denotes that y is computed by A using fresh random coins. When A is a set, $a \overset{\$}{\leftarrow} A$ denotes that a is chosen uniformly over A. For $n \in \mathbb{N}$, we write $[n]$ to denote the set $\{1, \ldots, n\}$.

2.2 Computational Diffie–Hellman assumption

Let $(\mathbb{G}, p, g)$ be a group generated by a group generation function $\mathcal{G}(\lambda)$. Given (p, g, g^a, g^b), where $a, b \overset{\$}{\leftarrow} \mathbb{Z}_p$, the CDH problem consists in computing g^{ab} in $\mathbb{G}$. The advantage of $\mathcal{A}$ for solving the CDH problem is defined as

$$\epsilon(\lambda) = \Pr[Z \leftarrow \mathcal{A}(p, g, g^a, g^b) : a, b \overset{\$}{\leftarrow} \mathbb{Z}_p \wedge Z = g^{ab}].$$

We say that the (t, ϵ)-CDH assumption holds in group $(\mathbb{G}, p, g)$ if no t-time algorithm has an advantage of at least ϵ when solving the CDH problem.

2.3 Twin Diffie–Hellman assumption

Let $(\mathbb{G}, p, g)$ be a group generated by a group generation function $\mathcal{G}(\lambda)$. Given $(p, g, g^a, g^{b_1}, g^{b_2})$, where $a, b_1, b_2 \overset{\$}{\leftarrow} \mathbb{Z}_p$ and a decisional twin Diffie–Hellman oracle (for fixed g^{b_1}, g^{b_2}), that is, an oracle that for any $(R, B_1, B_2) \in \mathbb{G}^3$ given as inputs correctly answers the question "is it the case that both $B_1 = g^{b_1 r}$ and $B_2 = g^{b_2 r}$, where r is the exponent for which $R = g^r$?", the TDH problem consists in computing (g^{ab_1}, g^{ab_2}). The advantage of $\mathcal{A}$ for solving the TDH problem is defined as

$$\epsilon(\lambda) = \Pr[(Z_1, Z_2) \leftarrow \mathcal{A}^{O_{\mathsf{DTDH}}(\cdot)}(p, g, g^a, g^{b_1}, g^{b_2}) : \\ a, b_1, b_2 \overset{\$}{\leftarrow} \mathbb{Z}_p \wedge Z_1 = g^{ab_1} \wedge Z = g^{ab_2}].$$

We say that the (t, ϵ)-TDH assumption holds in group $(\mathbb{G}, p, g)$ if no t-time algorithm has an advantage of at least ϵ when solving the TDH problem.

Theorem 1 [Theorem 1 in [9]] *The CDH assumption holds if and only if the TDH assumption holds.*

3 IND-CCA secure KEM

In this section, we define the MUC^+ model of KEM and present our practical KEM scheme. Then, we prove our scheme in the MUC^+ model of KEM.

3.1 Formal model

3.1.1 Syntax

A key encapsulation mechanism $\mathsf{KEM} = (\mathsf{KEM.Param}, \mathsf{KEM.Gen}, \mathsf{KEM.Encap}, \mathsf{KEM.Decap})$ consists of four algorithms. The parameter generation algorithm $\Pi \overset{\$}{\leftarrow} \mathsf{KEM.Param}(\lambda)$ takes the security parameter λ as input and outputs parameter Π. The key generation algorithm $(\mathsf{PK}, \mathsf{SK}) \overset{\$}{\leftarrow} \mathsf{KEM.Gen}(\Pi)$ generates, after receiving input Π, a public key PK and a private key SK. The encapsulation algorithm $(\mathsf{K}, \mathsf{CT}) \overset{\$}{\leftarrow} \mathsf{KEM.Encap}(\mathsf{PK})$ generates, with PK as input, a key K and a ciphertext CT. The decapsulation algorithm $\{\mathsf{K}, \perp\} \leftarrow \mathsf{KEM.Decap}(\mathsf{PK}, \mathsf{SK}, \mathsf{CT})$ takes public key PK, private key SK, and ciphertext CT as inputs and then outputs a key K or an error symbol $\perp$. The correctness of KEM is defined as follows: For all $(\mathsf{PK}, \mathsf{SK})$

generated by $\mathsf{KEM.Gen}(\Pi)$, it is required that $\bar{\mathsf{K}} = \mathsf{K}$ where $(\mathsf{K}, \mathsf{CT}) \overset{\$}{\leftarrow} \mathsf{KEM.Encap}(\mathsf{PK})$ and $\bar{\mathsf{K}} \leftarrow \mathsf{KEM.Decap}(\mathsf{PK}, \mathsf{SK}, \mathsf{CT})$.

3.1.2 Security model in a multi-user setting with corruptions

We define the MUC^+ security of KEM by referring to the security definition of $\mathsf{IND\text{-}CCA\text{-}MUC}$ security from PKE[21]. Our security notion is a more extended concept because it allows for corruption queries, which the previous notion did not. We argue that our MUC^+ model is more realistic because the previous MUC model cannot capture situations in which an adversary obtains some of the private keys of the users, which the MUC^+ model can capture. The following security experiment, which is a game played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, is parameterized by two integers $\mu, q_e \in \mathbb{N}$.

1. The challenger runs $\Pi \overset{\$}{\leftarrow} \mathsf{KEM.Param}(\lambda)$ once and then $\mathsf{KEM.Gen}(\Pi)$ μ times to generate μ key pairs $(\mathsf{PK}^{(i)}, \mathsf{SK}^{(i)})$, $i \in [\mu]$. Then, it tosses a coin $\beta \overset{\$}{\leftarrow} \{0, 1\}$, initializes lists $\mathsf{Clist} := \emptyset$ and $\mathsf{Klist} := \emptyset$ as empty lists, and defines a counter $j_i := 0$ for each $i \in [\mu]$.
2. The adversary receives the μ public keys $\{\mathsf{PK}^{(i)}\}_{i\in[\mu]}$ as input. It may query the challenger for three types of operations.
   - **Encapsulation queries** The adversary submits an index $i \in [\mu]$. If $j_i \geq q_e$ or $i \in \mathsf{Klist}$, then the challenger returns $\perp$. Otherwise, it generates keys K_0 and K_1 and a ciphertext CT by sampling $\mathsf{K}_0 \overset{\$}{\leftarrow} \mathcal{K}$ and computing $(\mathsf{K}_1, \mathsf{CT}) \overset{\$}{\leftarrow} \mathsf{KEM.Encap}(\mathsf{PK}^{(i)})$, respectively. Then it appends (CT, i) to Clist, updates counter j_i via $j_i = j_i + 1$, and returns $(\mathsf{K}_\beta, \mathsf{CT})$.
   - **Corruption queries** The adversary submits an index $i \in [\mu]$. If $(\mathsf{CT}, i) \in \mathsf{Clist}$ for some CT, then the challenger returns $\perp$. Otherwise, it returns $\mathsf{SK}^{(i)}$ and appends i to Klist.
   - **Decapsulation queries** The adversary submits a ciphertext CT and an index $i \in [\mu]$. If $(\mathsf{CT}, i) \in \mathsf{Clist}$, then the challenger returns $\perp$. Otherwise, it returns whatever $\mathsf{KEM.Decap}(\mathsf{PK}^{(i)}, \mathsf{SK}^{(i)}, \mathsf{CT})$ returns.
3. Eventually, the adversary $\mathcal{A}$ outputs a bit β'. We say that the adversary *wins* the game if $\beta = \beta'$.

Definition 1 Let $\mathcal{A}$ be an adversary that runs in time t, that makes at most q_e encapsulation queries per user, q_c corruption queries in total, and q_d decapsulation queries per user, and that wins with probability $1/2+\epsilon$. Then, we say that $\mathcal{A}$ breaks the $(\epsilon, t, \mu, q_e, q_c, q_d)$-$\mathsf{MUC}^+$ security of KEM. We say that KEM is $(\epsilon,t,\mu,q_e,q_c,q_d)$-$\mathsf{MUC}^+$ secure if there exists no such adversary $\mathcal{A}$.

Note that the above security model with $\mu = 1$, $q_e = 1$, and $q_c = 0$ is the standard $\mathsf{IND\text{-}CCA}$ security model of KEM in a single-user setting. When $\mu \geq 2$ and $q_c = 0$, it is equal to the previous MUC model of KEM.

3.2 Construction

Our KEM scheme is parameterized by group parameters $(\mathbb{G}, p, g)$, where p is a λ-bit integer, $\mathbb{G}$ is a cyclic group of order p, and g is a group generator, and by three hash functions, namely $H_1 : \{0, 1\}^* \to \mathbb{Z}_p$, $H_2 : \{0, 1\}^* \to \{0, 1\}^{\ell_1}$, and $H_3 : \{0, 1\}^* \to \{0, 1\}^{\ell_2}$. We denote these parameters as $\Pi = (\mathbb{G}, p, g, H_1, H_2, H_3)$.

As mentioned in the Introduction section, our KEM scheme uses a variant of the twin ElGamal approach. In this variant of twin ElGamal, a public key consists of two group elements $(X_1 = g^{x_1}, X_2 = g^{x_2}) \in \mathbb{G}^2$ with a corresponding private key $(x_1, x_2) \in \mathbb{Z}_p^2$, and encryption is done by computing $(g^s, m \oplus H_2(g^s, X_1^s, X_2^s))$ for a random exponent $s \in \mathbb{Z}_p$. Using this variant of twin ElGamal as a building block, the encapsulation algorithm of our scheme generates two ciphertexts of the same message $R \in \{0, 1\}^{\ell_1}$ and the same random seed $s \in \mathbb{Z}_p$ using two different public keys $(X_{0,1}, X_{0,2})$ and $(X_{1,1}, X_{1,2})$. The algorithms of our scheme are described as follows.

KEM.Gen(Π): Given parameters Π, the Setup algorithm runs as follows:

1. Select a bit $b \in \{0, 1\}$.
2. Select random exponents $x_1, x_2 \in \mathbb{Z}_p$ and set $(X_{b,1}, X_{b,2}) = (g^{x_1}, g^{x_2}) \in \mathbb{G}^2$.
3. Select random group elements $(X_{1-b,1}, X_{1-b,2}) \in \mathbb{G}^2$.
4. Output $\mathsf{PK} = (X_{0,1}, X_{0,2}, X_{1,1}, X_{1,2}) \in \mathbb{G}^4$ and $\mathsf{SK} = (b, x_1, x_2) \in \{0, 1\} \times \mathbb{Z}_p^2$.

KEM.Encap(Π, PK): Given a public key $\mathsf{PK} = (X_{0,1}, X_{0,2}, X_{1,1}, X_{1,2})$, the Encap algorithm runs as follows:

1. Select a random string $R \in \{0, 1\}^{\ell_1}$ and set $s = H_1(R) \in \mathbb{Z}_p$.
2. Compute $(g^s, X_{0,1}^s, X_{0,2}^s, X_{1,1}^s, X_{1,2}^s) \in \mathbb{G}^5$ and set $C_2 = g^s \in \mathbb{G}$.
3. Set $C_0 = R \oplus H_2(g^s, X_{0,1}^s, X_{0,2}^s)$ and $C_1 = R \oplus H_2(g^s, X_{1,1}^s, X_{1,2}^s)$.
4. Set $\mathsf{K} = H_3(R) \in \{0, 1\}^{\ell_2}$.
5. Output K and $\mathsf{CT} = (C_0, C_1, C_2) \in \{0, 1\}^{2\ell_1} \times \mathbb{G}$.

KEM.Decap$(\Pi, \mathsf{PK}, \mathsf{SK}, \mathsf{CT})$: Given a public key $\mathsf{PK} = (X_{0,1}, X_{0,2}, X_{1,1}, X_{1,2})$, a private key $\mathsf{SK} = (b, x_1, x_2)$, and a ciphertext $\mathsf{CT} = (C_0, C_1, C_2)$, the Decap algorithm runs as follows:

1. Compute $R_b = C_b \oplus H_2(C_2, C_2^{x_1}, C_2^{x_2}) \in \{0, 1\}^{\ell_1}$.
2. Set $s = H_1(R_b) \in \mathbb{Z}_p$ and $\mathsf{K} = H_3(R_b) \in \{0, 1\}^{\ell_2}$.
3. Compute $R_{1-b} = C_{1-b} \oplus H_2(C_2, X_{1-b,1}^s, X_{1-b,2}^s) \in \{0, 1\}^{\ell_1}$.
4. If $R_b = R_{1-b}$ and $C_2 = g^s$, output K. Otherwise, abort.

3.2.1 Correctness

Given $\mathsf{PK} = (X_{0,1}, X_{0,2}, X_{1,1}, X_{1,2}) \in \mathbb{G}^4$ and $\mathsf{SK} = (b, x_1, x_2) \in \{0, 1\} \times \mathbb{Z}_p^2$, $(\mathsf{K}, \mathsf{CT})$ is calculated by KEM.Encap(Π, PK) as follows: $\mathsf{K} = H_3(R)$, $\mathsf{CT} = (C_0, C_1, C_2) = (R \oplus H_2(g^s, X_{0,1}^s, X_{0,2}^s), R \oplus H_2(g^s, X_{1,1}^s, X_{1,2}^s), g^s)$ where $s = H_1(R)$. Then, we can show that KEM.Decap$(\Pi, \mathsf{PK}, \mathsf{SK}, \mathsf{CT})$=$\mathsf{K}$ as follows: first, we have $R_b = C_b \oplus H_2(C_2, C_2^{x_1}, C_2^{x_2}) = R \oplus H_2(g^s, X_{b,1}^s, X_{b,2}^s) \oplus H_2(C_2, C_2^{x_1}, C_2^{x_2})$. Since $X_{b,1}^s = g^{x_1 s} = C_2^{x_1}$ and $X_{b,2}^s = g^{x_2 s} = C_2^{x_2}$, we have $R_b = R$. Finally, we have $H_1(R_b) = s$, $C_2 = g^s = g^{H_1(R)}$ and $R_b = R_{1-b}$ by $R_{1-b} = C_{1-b} \oplus H_2(C_2, X_{1-b,1}^s, X_{1-b,2}^s) = R$. Then, this algorithm outputs $H_3(R_b) = H_3(R) = \mathsf{K}$.

3.3 Security proof

Theorem 2 *Suppose that the $(\epsilon_{TDH}, t_{TDH})$-TDH assumption holds in $(\mathbb{G}, p, g)$, that H_1, H_2, and H_3 are random oracles, and that an adversary $\mathcal{A}$ makes at most q_{H_1} H_1-queries, q_{H_2} H_2-queries, q_{H_3} H_3-queries, q_e encapsulation queries per user, q_c corruption queries in total, and q_d decryption queries per user. Then, the above mentioned KEM scheme is $(\epsilon, t, \mu, q_e, q_c, q_d)$-*

MUC$^+$ secure, where

$$\epsilon \leq 2\epsilon_{TDH} + \frac{q_{H_3}}{2^{\ell_2}}\mu q_e,$$
$$t \approx t_{TDH} + (5\mu + 2\mu q_e + 5\mu q_d)t_e.$$

Here, t_e denotes the time required for computing exponentiation in $\mathbb{G}$ and ℓ_2 denotes the output length of random oracle H_2.

Proof The proof is presented as a sequence of hybrid games **Game 0**,..., **Game 3**. Game 0 is the actual MUC^+ security game and, in Game 3, the adversary will win with an exact probability of 1/2. Let Win_i denote the event in which $\mathcal{A}$ wins in Game i.
Game 0 This is the (μ, q_e)-MUC^+ security experiment from Definition 2. Let $b_i \in \{0, 1\}$ be the randomly chosen coin associated with the private key of user $i \in [\mu]$. In all subsequent games, the coin b_i for each user $i \in [\mu]$ is fixed. By definition, we have

$$\Pr[\mathsf{Win}_0] = \frac{1}{2} + \epsilon_{\mathsf{KEM}}^{\mathsf{MUC}^+}.$$

Game 1 In this game, the challenger simulates the random oracles, generating random answers for any new queries as shown in Table 2. This game is identical to Game 0. Thus, we have

$$\Pr[\mathsf{Win}_1] = \Pr[\mathsf{Win}_0].$$

Game 2 This game is identical to Game 1 except that now the encapsulation oracle operates differently. Given an index $i \in [\mu]$, unlike the previous game, the challenger sets $C_2 \overset{\$}{\leftarrow} \mathbb{G}$, instead of computing $C_2 = g^s$ where $s = H_1(R)$. Then, it sets $C_j = R \oplus V_j$, where V_j is randomly chosen from $\{0, 1\}^{\ell_1}$ for $j \in \{0, 1\}$. The rest of the procedure is the same as that in Game 1.

Note that Game 1 and Game 2 are indistinguishable unless at least one of the following events occurs:

Event E : $(g^{\bar{s}}, X_{1-b,1}^{\bar{s}}, X_{1-b,2}^{\bar{s}})$ is asked to the H_2-oracle, where $g^{\bar{s}} = C_2$ for some $((C_0, C_1, C_2), i)$ in Clist, $\mathsf{PK}^{(i)} = (X_{0,1}, X_{0,2}, X_{1,1}, X_{1,2})$, and $\mathsf{SK}^{(i)} = (b, x_1, x_2)$.
Event F : $(g^{\bar{s}}, X_{b,1}^{\bar{s}}, X_{b,2}^{\bar{s}})$ is asked to the H_2-oracle, where $g^{\bar{s}} = C_2$ for some $((C_0, C_1, C_2), i)$ in Clist, $\mathsf{PK}^{(i)} = (X_{0,1}, X_{0,2}, X_{1,1}, X_{1,2})$, and $\mathsf{SK}^{(i)} = (b, x_1, x_2)$.

Then, we have

$$|\Pr[\mathsf{Win}_2] - \Pr[\mathsf{Win}_1]| \leq \Pr[\mathsf{E}] + \Pr[\mathsf{F}] \leq 2\Pr[\mathsf{E}] \leq 2\epsilon_{\mathsf{TDH}}.$$

We prove that the inequality above holds after finishing the proof of theorem 2.
Game 3 In this game, the challenge ciphertexts are computed differently compared with the previous game. Given index $i \in [\mu]$, unlike the previous game, the challenger randomly selects $\mathsf{K} \in \mathcal{K}$. Finally, it outputs $(\mathsf{K}, \mathsf{CT} = (C_0, C_1, C_2))$.

Game 2 and Game 3 are completely indistinguishable unless query $R^{(i)}$ for any $i \in [\mu]$ is asked to the H_3-oracle by either the adversary or the decryption oracle. Thus, we have

$$|\Pr[\mathsf{Win}_3] - \Pr[\mathsf{Win}_2]| \leq \frac{q_{H_3}}{2^{\ell_2}}\mu q_e.$$

From the results mentioned above, we can conclude that $\epsilon \leq 2\epsilon_{\mathsf{TDH}} + \frac{q_{H_3}}{2^{\ell_2}}\mu q_e$, which completes the proof. ◻

Table 2 Simulation of a MUC$^+$ game

H_1, H_2, H_3 oracles	Query $H_1(R)$: If a tuple (R, s) appears in the H_1-list, it returns s. Otherwise, it randomly chooses $s \in \mathbb{Z}_p$, returns s, and then appends tuple (R, s) to the H_1-list Query $H_2(C_2, Y_1, Y_2)$: If a tuple (C_2, Y_1, Y_2, V) appears in the H_2-list, it returns V. Otherwise, it randomly chooses $V \in \{0, 1\}^{\ell_1}$, returns V, and then appends tuple (C_2, Y_1, Y_2, V) to the H_2-list Query $H_3(R)$: If a tuple (R, K) appears in the H_3-list, it returns K. Otherwise, it randomly chooses $\mathsf{K} \in \{0, 1\}^{\ell_2}$, returns K, and then appends tuple (R, K) to the H_3-list
User key pair generation	For each $i \in [\mu]$, a user key pair $(\mathsf{PK}^{(i)}, \mathsf{SK}^{(i)})$ is generated as follows (1) Select two random exponents x_1, x_2 and a bit b (2) Select two random group elements $X_{1-b,1}$ and $X_{1-b,2}$ (3) Compute $X_{b,1} = g^{x_1}$ and $X_{b,2} = g^{x_2}$. (4) Set $\mathsf{PK}^{(i)} = (X_{0,1}, X_{0,2}, X_{1,1}, X_{1,2})$ and $\mathsf{SK}^{(i)} = (b, x_1, x_2)$
Encapsulation oracle	Query $Encap(i)$: If $j_i \geq q_e$ or $i \in \mathsf{Klist}$, it return $\perp$. Otherwise, it sets $j_i \leftarrow j_i + 1$, appends (CT, i) to Clist, and then returns the answer $(\mathsf{K}, \mathsf{CT})$, which is defined as follows (1) Select random strings R and K_0 (2) Set $s = H_1(R)$ and $\mathsf{K}_1 = H_3(R)$ (3) Compute $(g^s, X_{0,1}^s, X_{0,2}^s, X_{1,1}^s, X_{1,2}^s)$ and set $C_2 = g^s$ (4) Set $C_0 = R \oplus H_2(g^s, X_{0,1}^s, X_{0,2}^s)$ and $C_1 = R \oplus H_2(g^s, X_{1,1}^s, X_{1,2}^s)$ (5) Set $\mathsf{K} = \mathsf{K}_\beta$ and $\mathsf{CT} = (C_0, C_1, C_2)$
Corruption oracle	Query $Corrupt(i)$: If a tuple (CT, i) appears in Clist for some CT, it returns $\perp$. Otherwise, it appends i to Klist and returns the answer $\mathsf{SK}^{(i)}$ defined in the user key pair generation phase
Decapsulation oracle	Query $Decap$ (CT, i): If a tuple (CT, i) appears in Clist, it returns $\perp$. Otherwise, it returns the answer K, which is defined as follows (1) Retrieve a private key $\mathsf{SK}^{(i)} = (b, x_1, x_2)$. (2) Compute (Y_1, Y_2)=$(C_2^{x_1}, C_2^{x_2})$ and set $R_b = C_b \oplus H_2(C_2, Y_1, Y_2)$ (3) Set $s' = H_1(R_b)$ and $\mathsf{K} = H_3(R_b)$ (4) Compute $R_{1-b} = C_{1-b} \oplus H_2(C_2, X_{1-b,1}^{s'}, X_{1-b,2}^{s'})$ (5) If $R_b \neq R_{1-b}$ or $C_2 \neq g^{s'}$, then return $\perp$. Else, return K

Claim 1 $\Pr[\mathsf{E}] = \Pr[\mathsf{F}]$.

Proof It is sufficient to show that the adversary's view is independent of the variable $b^{(i)}$ such that $i \notin \mathsf{Klist}$ for $i \in [\mu]$, because if $i \in \mathsf{Klist}$, the abovementioned events E and F never occur for i. Because the only variables that are possibly dependent on the $b^{(i)}$ such that $i \notin \mathsf{Klist}$ are the responses from the decapsulation oracle, it is sufficient to show that there exists no ciphertext CT such that its decapsulation results are different depending on $b^{(i)}$. For the sake of creating a contradiction, we assume that there exists

Table 3 Algorithms for generating random self-reducible TDH instances

Algorithm $\mathcal{R}_0$	Given a TDH instance (g^a, g^{b_1}, g^{b_2}), it chooses random exponents $r_0 \in \mathbb{Z}_p$ and computes $g^{a'}, g^{b'_1}, g^{b'_2}$ as follows: $g^{a'} = (g^a)^{r_0}, g^{b'_1} = g^{b_1}, g^{b'_2} = g^{b_2}$. Then, it outputs $(g^{a'}, g^{b'_1}, g^{b'_2}, r_0)$.
Algorithm $\mathcal{R}_1$	Given a TDH instance (g^a, g^{b_1}, g^{b_2}), it chooses random exponents $r_0, r_1, r_2 \in \mathbb{Z}_p$ and computes $g^{a'}, g^{b'_1}, g^{b'_2}$ as follows: $g^{a'} = (g^a)^{r_0}, g^{b'_1} = (g^{b_1})^{r_1}, g^{b'_2} = (g^{b_2})^{r_2}$. Then, it outputs $(g^{a'}, g^{b'_1}, g^{b'_2}, r_0, r_1, r_2)$.

$\mathsf{CT} = (C_0, C_1, C_2)$ such that $Decap(\mathsf{PK}^{(i)}, \mathsf{SK}^{(i)}_{b^{(i)}=0}, \mathsf{CT}) \neq Decap(\mathsf{PK}^{(i)}, \mathsf{SK}^{(i)}_{b^{(i)}=1}, \mathsf{CT})$, where $\mathsf{PK}^{(i)} = (X_{0,1}, X_{0,2}, X_{1,1}, X_{1,2})$, $\mathsf{SK}^{(i)}_{b^{(i)}=0} = (0, x_1 = \log_g X_{0,1}, x_2 = \log_g X_{0,1})$, and $\mathsf{SK}^{(i)}_{b^{(i)}=1} = (1, x'_1 = \log_g X_{1,1}, x'_2 = \log_g X_{1,1})$. Without loss of generality, we assume that the results of decapsulation on the left are not $\perp$. Let $s^* = \log_g C_2$; then, by the "*well-formed*" condition checked in the decapsulation procedure at step (5), it holds that $s^* = s$, where $s = H_1(R_0)$, $R_0 = C_0 \oplus V_0$, and $V_0 = H_2(C_2, C_2^{x_1}, C_2^{x_2})$ (when $b^{(i)} = 0$). Then, it holds that $V_1 = V'_1$, where V_1 comes from the decapsulation procedure at step (4) such that $V_1 = H_2(C_2, X^s_{1,1}, X^s_{1,2})$ (when $b^{(i)} = 0$), and V'_1 comes from the decapsulation procedure at step (2) such that $V'_1 = H_2(C_2, C_2^{x'_1}, C_2^{x'_2})$ (when $b^{(i)} = 1$) because $C_2^{x'_1} = X^{s^*}_{1,1}$, $C_2^{x'_2} = X^{s^*}_{1,2}$, and $s^* = s$. Therefore, it holds that $R_0 = R'_1$ because $R_0 = R_1 = C_1 \oplus V_1$ (when $b^{(i)} = 0$) and $R'_1 = C_1 \oplus V'_1$ (when $b^{(i)} = 1$). Hence, the two results of decapsulation $H_3(R_0)$ (when $b^{(i)} = 0$) and $H_3(R'_1)$ (when $b^{(i)} = 1$) are the same, which is a contradiction. This completes the proof of the claim. □

Claim 2 $|\Pr[\mathsf{Win}_2] - \Pr[\mathsf{Win}_1]| \leq 2\epsilon_{\mathsf{TDH}}$.

Proof We show that $|\Pr[\mathsf{Win}_2] - \Pr[\mathsf{Win}_1]| \leq 2\epsilon_{\mathsf{TDH}}$ by constructing an algorithm $\mathcal{B}$, which solves the TDH problem using an adversary $\mathcal{A}_{\mathsf{KEM}}$. $\mathcal{B}$ acts as a challenger for $\mathcal{A}_{\mathsf{KEM}}$.

Let $(A, B_1, B_2, O_{\mathsf{DTDH}}(\cdot))$ be a TDH instance given to $\mathcal{B}$. For the key pair generation, for each $i \in [\mu]$ and via the random self-reducibility of TDH, $\mathcal{B}$ obtains a randomized TDH instance (A', B'_1, B'_2) from the $\mathcal{R}_1$ algorithm defined in Table 3 and records the instance with the exponents in the $\mathcal{R}_1$-list. Then, $\mathcal{B}$ sets $(X_{1-b_i,1}, X_{1-b_i,2}) = (B'_1, B'_2)$ instead of randomly selecting two group points $X_{1-b_i,1}, X_{1-b_i,2} \in \mathbb{G}$.

For the encapsulation oracle, given an index $i \in [\mu]$, $\mathcal{B}$ retrieves the TDH instance (A', B'_1, B'_2) from the $\mathcal{R}_1$-list. Then, $\mathcal{B}$ obtains a randomized TDH instance (A'', B'_1, B'_2) from the $\mathcal{R}_0$ algorithm defined in Table 3 and records it with the exponents in the $\mathcal{R}_0$-list. $\mathcal{B}$ sets $C_2 = A''$, $C_j = R \oplus V_j$, where V_j is randomly chosen from $\{0, 1\}^{\ell_1}$ for $j \in \{0, 1\}$. The rest of the procedure is the same as that in Game 1.

As mentioned above, Game 1 and Game 2 are indistinguishable unless at least one of the following events occurs:

Event E : $(g^{\bar{s}}, X^{\bar{s}}_{1-b,1}, X^{\bar{s}}_{1-b,2})$ is asked to the H_2-oracle, where $g^{\bar{s}} = C_2$ for some $((C_0, C_1, C_2), i)$ in Clist, $\mathsf{PK}^{(i)} = (X_{0,1}, X_{0,2}, X_{1,1}, X_{1,2})$, and $\mathsf{SK}^{(i)} = (b, x_1, x_2)$.

Event F : $(g^{\bar{s}}, X^{\bar{s}}_{b,1}, X^{\bar{s}}_{b,2})$ is asked to the H_2-oracle, where $g^{\bar{s}} = C_2$ for some $((C_0, C_1, C_2), i)$ in Clist, $\mathsf{PK}^{(i)} = (X_{0,1}, X_{0,2}, X_{1,1}, X_{1,2})$, and $\mathsf{SK}^{(i)} = (b, x_1, x_2)$.

Assume that the Event E occurs i.e., $(\bar{C}_2, X_{1-b,1}^{\log_g \bar{C}_2}, X_{1-b,2}^{\log_g \bar{C}_2})$ is asked where $((\bar{C}_0, \bar{C}_1, \bar{C}_2), \bar{i})$ in Clist, $\mathsf{PK}^{(\bar{i})} = (X_{0,1}, X_{0,2}, X_{1,1}, X_{1,2})$, and $\mathsf{SK}^{(\bar{i})} = (b, x_1, x_2)$ for some $\bar{i} \in [\mu]$. Then, $\mathcal{B}$ can find the answer for the original TDH instance (A, B_1, B_2) from the tuple $(\bar{C}_2, X_{1-b,1}^{\log_g \bar{C}_2}, X_{1-b,2}^{\log_g \bar{C}_2})$ in the H_2-oracle, because the tuple $(\bar{C}_2, X_{1-b,1}, X_{1-b,2})$ is a randomized TDH instance using algorithms $\mathcal{R}_0$ and $\mathcal{R}_1$. More precisely, let $(A, B_1, B_2) = (g^a, g^{b_1}, g^{b_2})$ be the original TDH instance. Then, we see that $(X_{1-b,1}, X_{1-b,2}) = (B_1', B_2') = (g^{b_1\bar{r}_1}, g^{b_2\bar{r}_2})$ and $\bar{C}_2 = A'' = g^{a\bar{r}_0}$. Thus, we have $(X_{1-b,1}^{\log_g \bar{C}_2}, X_{1-b,2}^{\log_g \bar{C}_2}) = (g^{ab_1\bar{r}_0\bar{r}_1}, g^{ab_2\bar{r}_0\bar{r}_2})$, where $\mathcal{B}$ can remove the randomized exponents $(\bar{r}_0, \bar{r}_1, \bar{r}_2)$ found in the $\{\mathcal{R}_0, \mathcal{R}_1\}$-lists and recover the answer for the original TDH instance (g^{ab_1}, g^{ab_2}). Note that this reduction is tight, because $\mathcal{B}$ can find the correct answer for the TDH instance from the H_2-oracle queries, using the decisional TDH oracle $O_{\mathsf{DTDH}}(\cdot)$.

From the results of the two claims, we can conclude that

$$|\Pr[\mathsf{Win}_3] - \Pr[\mathsf{Win}_2]| \leq \Pr[\mathsf{E}] + \Pr[\mathsf{F}] \leq 2\Pr[\mathsf{E}] \leq 2\epsilon_{\mathsf{TDH}},$$

as required. □

4 IND-CCA secure PKE

In this section, we define the MUC$^+$ model of PKE and review augmented hybrid encryption using augmented DEM [16] (hereafter denoted as 'ADEM'), which is the building block of our PKE scheme. Then, we prove that hybrid encryption combining KEM (secure in the MUC$^+$ model) and ADEM (secure in the multi-instance setting) is tightly secure in the MUC$^+$ model of PKE.

4.1 Formal model

4.1.1 Syntax

A public key encryption scheme PKE = (PKE.Param, PKE.Gen, PKE.Enc, PKE.Dec) consists of four algorithms. The parameter generation algorithm $\Pi \overset{\$}{\leftarrow}$ PKE.Param(λ) takes the security parameter λ as input and outputs parameter Π. The key generation algorithm (PK, SK) $\overset{\$}{\leftarrow}$ PKE.Gen(Π) takes Π as input and generates a public key PK and a secret key SK. The encryption algorithm CT $\overset{\$}{\leftarrow}$ PKE.Enc(PK, m) takes a public key PK and a message m as inputs and then outputs a ciphertext CT. The decryption algorithm $m \leftarrow$ PKE.Dec(PK, SK, CT) takes a public key PK, a secret key SK, and a ciphertext CT as inputs and then outputs a message m. The correctness of PKE is defined as follows: for all (PK, SK) generated by PKE.Gen(Π), it is required that PKE.Dec(PK, SK, PKE.Enc(PK, m)) $= m$ for any $m \in \mathcal{M}$.

4.1.2 Security model in a multi-user setting with corruptions

We define the MUC$^+$ security of PKE by referring to the security definition of IND-CCA-MUC security from PKE [21]. As in the MUC$^+$ security notion of KEM, the abovementioned security notion is also an extended concept that allows for corruption queries. The following security experiment, which is a game played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, is parameterized by two integers $\mu, q_e \in \mathbb{N}$.

1. The challenger runs $\Pi \overset{\$}{\leftarrow} \mathsf{PKE.Param}(\lambda)$ once and then $\mathsf{PKE.Gen}(\Pi)$ μ times to generate μ key pairs $(\mathsf{PK}^{(i)}, \mathsf{SK}^{(i)})$, $i \in [\mu]$. Then, it tosses a coin $\beta \overset{\$}{\leftarrow} \{0, 1\}$, initializes lists $\mathsf{Clist} := \emptyset$ and $\mathsf{Klist} := \emptyset$ as empty lists, and defines a counter $j_i := 0$ for each $i \in [\mu]$.
2. The adversary receives the μ public keys $\{\mathsf{PK}^{(i)}\}_{i\in[\mu]}$ as input. It may query the challenger for three types of operations.
   - **Encryption queries** The adversary submits an index $i \in [\mu]$ and two messages (m_0, m_1). If $j_i \geq q_e$ or $i \in \mathsf{Klist}$, then the challenger returns $\perp$. Otherwise, it generates a ciphertext CT by computing $\mathsf{CT} \overset{\$}{\leftarrow} \mathsf{PKE.Enc}(\mathsf{PK}^{(i)}, m_\beta)$. Then, it appends (CT, i) to Clist, updates counter j_i via $j_i = j_i + 1$, and returns CT.
   - **Corruption queries** The adversary submits an index $i \in [\mu]$. If $(\mathsf{CT}, i) \in \mathsf{Clist}$ for some CT, then the challenger returns $\perp$. Otherwise, it returns $\mathsf{SK}^{(i)}$ and appends i to Klist.
   - **Decryption queries** The adversary submits a ciphertext CT and an index $i \in [\mu]$. If $(\mathsf{CT}, i) \in \mathsf{Clist}$, then the challenger returns $\perp$. Otherwise, it returns whatever $\mathsf{PKE.Dec}(\mathsf{PK}^{(i)}, \mathsf{SK}^{(i)}, \mathsf{CT})$ returns.
3. Eventually, the adversary $\mathcal{A}$ outputs a bit β'. We say that the adversary *wins* the game if $\beta = \beta'$.

As in the case of KEM, the above security model with $\mu = 1$, $q_e = 1$, and $q_c = 0$ is a standard IND-CCA security model of PKE in a single-user setting. When $\mu \geq 2$ and $q_c = 0$, it is equal to the previous MUC model of PKE.

Definition 2 Let $\mathcal{A}$ be an adversary that runs in time t, that makes at most q_e encryption queries per user, q_c corruption queries in total, and q_d decryption queries per user, and that wins with probability $1/2+\epsilon$. Then, we can say that $\mathcal{A}$ *breaks* the $(\epsilon, t, \mu, q_e, q_c, q_d)$-MUC$^+$ security of PKE. We say that PKE is $(\epsilon, t, \mu, q_e, q_c, q_d)$-MUC$^+$ secure if there exists no such adversary $\mathcal{A}$.

Because of the results by Giacon et al. [16], we note that the standard hybrid KEM+DEM encryption paradigm is not enough for the a tightly secure PKE scheme in the MUC/MUC$^+$ model. To construct a tightly secure hybrid encryption scheme in the MUC$^+$ model, we shall use their results. In other words, we apply our proposed KEM approach in the *augmented hybrid encryption* paradigm [16], which uses the *augmented data encapsulation mechanism* instead of the standard DEM.

4.2 Augmented data encapsulation mechanism

4.2.1 Syntax

An augmented data encapsulation mechanism scheme ADEM = (ADEM.Enc, ADEM.Dec) consists of two algorithms with a message space $\mathcal{M}$, a tag space $\mathcal{T}$, and a ciphertext space $\mathcal{C}$. The encapsulation algorithm $\mathsf{C} \leftarrow \mathsf{ADEM.Enc}(\mathsf{K}, \mathsf{t}, m)$ takes a key $\mathsf{K} \in \mathcal{K}$, a tag $\mathsf{t} \in \mathcal{T}$, and a message $m \in \mathcal{M}$ as inputs and then outputs a ciphertext $\mathsf{C} \in \mathcal{C}$. The decapsulation algorithm $\{m, \perp\} \leftarrow \mathsf{ADEM.Dec}(\mathsf{K}, \mathsf{t}, \mathsf{C})$ takes a key $\mathsf{K} \in \mathcal{K}$, a tag $\mathsf{t} \in \mathcal{T}$, and a ciphertext $\mathsf{C} \in \mathcal{C}$ as inputs and then outputs either a message $m \in \mathcal{M}$ or the special symbol $\perp \notin \mathcal{M}$.

The correctness of ADEM is defined as follows: for all $\mathsf{K} \in \mathcal{K}$, $\mathsf{t} \in \mathcal{T}$, and $m \in \mathcal{M}$, it is required that $\mathsf{ADEM.Dec}(\mathsf{K}, \mathsf{t}, \mathsf{ADEM.Enc}(\mathsf{K}, \mathsf{t}, m)) = m$.

4.2.2 Nonce-based tag multi-instance one-time indistinguishability [16]

The security notion of nonce-based tag multi-instance one-time indistinguishability for ADEM is based on one in which a non-repeating tag is used. The following security experiment, which is a game played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, is parameterized by a positive integer N.

1. For $i \in [N]$, the challenger samples $\mathsf{K}_i \xleftarrow{\$} \mathcal{K}$. Then, it tosses a coin $\beta \xleftarrow{\$} \{0, 1\}$ and initializes empty sets $T := \emptyset$ and $C_i := \emptyset$ for each $i \in [N]$.
2. The adversary may query the challenger for two types of operations.
   - **Encapsulation queries** The adversary submits an index $i \in [N]$, a tag $\mathsf{t} \in \mathcal{T}$, and two messages (m_0, m_1). If $C_i \neq \emptyset$ or $\mathsf{t} \in T$, then the challenger returns $\perp$. Otherwise, it sets $\mathsf{t}_i = \mathsf{t}$ and $T = T \cup \{\mathsf{t}_i\}$ and generates a ciphertext $\mathsf{C} \in \mathcal{C}$ by computing $\mathsf{C} \leftarrow \mathsf{ADEM.Enc}(\mathsf{K}_i, \mathsf{t}_i, m_\beta)$. Then, it sets $C_i = C_i \cup \{\mathsf{C}\}$ and returns C.
   - **Decapsulation queries** The adversary submits an index $i \in [N]$ and a ciphertext C. If $C_i = \emptyset$ or $\mathsf{C} \in C_i$, then the challenger returns $\perp$. Otherwise, it returns whatever $\mathsf{ADEM.Dec}(\mathsf{K}_i, \mathsf{t}_i, \mathsf{C})$ returns.
3. Eventually the adversary $\mathcal{A}$ outputs a bit β'. We say that the adversary *wins* the game if $\beta = \beta'$.

Definition 3 [N-MIOT-IND security [16]] Let $\mathcal{A}$ be an adversary that runs in time t and wins with probability $1/2 + \epsilon$. Then, we say that $\mathcal{A}$ breaks the (ϵ, t, N)-MIOT-IND security of ADEM. We say that ADEM is (ϵ, t, N)-N-MIOT-IND secure if there exists no such adversary $\mathcal{A}$.

4.3 Augmented hybrid encryption

PKE.Gen(Π): Given parameters Π, the key generation algorithm runs $(\mathsf{PK}, \mathsf{SK}) \xleftarrow{\$} \mathsf{KEM.Gen}$ and returns $(\mathsf{PK}, \mathsf{SK})$.
PKE.Enc(Π, PK, m): Given parameters Π, a public key PK, and a message m, the encryption algorithm runs $(\mathsf{K}, \mathsf{C}_1) \xleftarrow{\$} \mathsf{KEM.Encap}(\mathsf{PK})$. Then, it runs $\mathsf{C}_2 \leftarrow \mathsf{ADEM.Enc}(\mathsf{K}, \mathsf{C}_1, m)$ and returns $\mathsf{CT} = (\mathsf{C}_1, \mathsf{C}_2)$.
PKE.Dec$(\Pi, \mathsf{SK}, \mathsf{CT})$: Given a secret key SK and a ciphertext $\mathsf{CT} = (\mathsf{C}_1, \mathsf{C}_2)$, the decryption algorithm runs $\mathsf{K} \leftarrow \mathsf{KEM.Decap}(\mathsf{SK}, \mathsf{C}_1)$. Then, it returns whatever $\mathsf{ADEM.Dec}(\mathsf{K}, \mathsf{C}_1, \mathsf{C}_2)$ returns.

Correctness We can simply check that the hybrid encryption is correct if both underlying schemes KEM and ADEM have the correctness.

Security proof Consider the following lemma, which states that augmented hybrid encryption combining a secure KEM scheme in the MUC model and a secure ADEM scheme in the N-MIOT-IND model is a tightly secure PKE scheme in the MUC model.

Lemma 1 [16, Lemma 5.3] *Let PKE be the hybrid public key encryption scheme constructed from a KEM and an ADEM schemes as mentioned above. Let p be the maximum ciphertext-*

collision probability of KEM over all positive public keys. Then, for any number of users μ and any PKE adversary $\mathcal{A}$ that makes at most q_e encryption queries and q_d decryption queries per user, there exist a KEM adversary $\mathcal{B}$ and an ADEM adversary $\mathcal{C}$ such that

$$\epsilon^{MUC}_{PKE,\mathcal{A},\mu,q_e} \leq 2\epsilon^{MUC}_{KEM,\mathcal{B},\mu,q_e} + \epsilon^{N\text{-}MIOT\text{-}IND}_{ADEM,\mathcal{C},N} + 2\binom{N}{2}p,$$

where $N = \mu q_e$. $\mathcal{B}$ poses at most q_e encapsulation queries and q_d decapsulation queries per user, and $\mathcal{C}$ poses at most μq_e encapsulation queries and μq_e decapsulation queries in total.

Note that our proposed MUC^+ security notion of PKE and KEM is different from that in the above lemma in that our model allows for corruption queries. The following theorem states that augmented hybrid encryption combining a secure KEM scheme in the MUC^+ model and a secure ADEM scheme in the N-MIOT-IND model is a tightly secure PKE scheme in the MUC^+ model.

Theorem 3 *Let PKE be a hybrid public key encryption scheme constructed from a KEM and an ADEM schemes as mentioned above. Let p be the maximum ciphertext-collision probability of KEM over all positive public keys. Then, for any number of users μ and any PKE adversary $\mathcal{A}$ that makes at most q_e encryption queries per user, q_d decryption queries per user, and q_c corruption queries in total, there exist a KEM adversary $\mathcal{B}$ and an ADEM adversary $\mathcal{C}$ such that*

$$\epsilon^{MUC^+}_{PKE,\mathcal{A},\mu,q_e} \leq 2\epsilon^{MUC^+}_{KEM,\mathcal{B},\mu,q_e} + \epsilon^{N\text{-}MIOT\text{-}IND}_{ADEM,\mathcal{C},N} + 2\binom{N}{2}p,$$

where $N = \mu q_e$. $\mathcal{B}$ poses at most q_e encapsulation queries, q_d decapsulation queries per user, and q_c corruption queries in total, and $\mathcal{C}$ poses at most μq_e encapsulation queries and μq_e decapsulation queries in total.

Proof We consider a sequence of hybrid games, **Game 0**,..., **Game 5**, where Game 0 is the actual MUC^+ PKE security game with a fixed coin $\beta = 0$ and Game 5 is the same game with a fixed coin $\beta = 1$. Let Win_i denote the event that $\mathcal{A}$ wins in Game i.
Game 0 This is the MUC^+ PKE security experiment from Definition 2 executed with $\beta = 0$. Thus, the challenger returns $\mathsf{CT} = (\mathsf{C}_1, \mathsf{C}_2)$, where $(\mathsf{K}, \mathsf{C}_1) \xleftarrow{\$} \mathsf{KEM.Encap}(\mathsf{PK}_i)$ and $\mathsf{C}_2 \leftarrow \mathsf{ADEM.Enc}(\mathsf{K}, \mathsf{C}_1, m_0)$ for the encryption query (i, m_0, m_1) only when $j_i < q_e$ and $i \notin \mathsf{Klist}$. When asked a corruption query for $i \in [\mu]$, it returns SK_i from $\mathsf{KEM.Gen}$ and appends i to Klist only when $(\mathsf{CT}, i) \notin \mathsf{Clist}$ for any CT. When asked a decryption query for $(\mathsf{CT} = (\mathsf{C}_1, \mathsf{C}_2), i)$, if $(\mathsf{CT}, i) \in \mathsf{Clist}$, it returns $\perp$; otherwise, it returns $m \leftarrow \mathsf{ADEM.Dec}(\mathsf{K}, \mathsf{C}_1, \mathsf{C}_2)$, where $\mathsf{K} \leftarrow \mathsf{KEM.Decap}(\mathsf{SK}_i, \mathsf{C}_1)$.
Game 1 This game is identical to Game 0 except that we change the way in which encryption queries are executed. We replace the keys for data encapsulation with randomly chosen keys in $\mathcal{K}$. In other words, the challenger returns $\mathsf{CT} = (\mathsf{C}_1, \mathsf{C}_2)$, where $(\mathsf{K}, \mathsf{C}_1) \xleftarrow{\$} \mathsf{KEM.Encap}(\mathsf{PK}_i)$, $\mathsf{C}_2 = \mathsf{ADEM.Enc}(\mathsf{K}', \mathsf{C}_1, m_0)$, and $\mathsf{K}' \xleftarrow{\$} \mathcal{K}$ for the encryption query (i, m_0, m_1) only when $j_i < q_e$ and $i \notin \mathsf{Klist}$. We claim that there exists an adversary $\mathcal{B}$ such that

$$|\Pr[\mathsf{Win}_1] - \Pr[\mathsf{Win}_0]| = \epsilon^{\mathsf{MUC}^+}_{\mathsf{KEM},\mathcal{B},\mu,q_e}.$$

Game 2 This game is identical to Game 1, except that now the encryption oracle operates differently. In this game, the challenger aborts if $\mathsf{KEM.Encap}$ generates the same ciphertext C_1 more than once. Let p be the maximum ciphertext-collision probability of KEM. Then, the probability that the challenger aborts during the i-th encryption query is lower-bounded by $(i-1)p$. By summing up all the probabilities for up to $N = \mu q_e$ queries(i.e., $p + 2p + \cdots + (N-1)p$), we have

$$|\Pr[\mathsf{Win}_2] - \Pr[\mathsf{Win}_1]| \leq \binom{N}{2} p.$$

Game 3 This game is identical to Game 2 except that the challenge ciphertexts are now computed differently. In this game, the challenger encrypts m_β with $\beta = 1$. In other words, it returns $\mathsf{CT} = (\mathsf{C}_1, \mathsf{C}_2)$, where $(\mathsf{K}, \mathsf{C}_1) \xleftarrow{\$} \mathsf{KEM.Encap}(\mathsf{PK}_i)$, $\mathsf{C}_2 = \mathsf{ADEM.Enc}(\mathsf{K}', \mathsf{C}_1, m_1)$, and $\mathsf{K}' \xleftarrow{\$} \mathcal{K}$ for the encryption query (i, m_0, m_1) only when $j_i < q_e$ and $i \notin \mathsf{Klist}$. We claim that there exists an adversary $\mathcal{C}$ such that

$$|\Pr[\mathsf{Win}_3] - \Pr[\mathsf{Win}_2]| = \epsilon^{\mathsf{N\text{-}MIOT\text{-}IND}}_{\mathsf{ADEM},\mathcal{C},N}.$$

Game 4 This game is identical to Game 3, except that the encryption oracle now operates differently. Now the challenger does not abort even if $\mathsf{KEM.Encap}$ generates he same ciphertext C_1 more than once. Then, just as in Game 1 and Game 2, we have

$$|\Pr[\mathsf{Win}_4] - \Pr[\mathsf{Win}_3]| \leq \binom{N}{2} p.$$

Game 5 This game is identical to Game 4 except that we change the way in which the encryption queries are executed. We replace the keys randomly chosen in $\mathcal{K}$ with the keys generated by $\mathsf{KEM.Encap}$ for data encapsulation. In other words, the challenger returns $\mathsf{CT} = (\mathsf{C}_1, \mathsf{C}_2)$, where $(\mathsf{K}, \mathsf{C}_1) \xleftarrow{\$} \mathsf{KEM.Encap}(\mathsf{PK}_i)$ and $\mathsf{C}_2 = \mathsf{ADEM.Enc}(\mathsf{K}, \mathsf{C}_1, m_1)$ for the encryption query (i, m_0, m_1) only when $j_i < q_e$ and $i \notin \mathsf{Klist}$. Then, just as in Game 0 and Game 1, we have

$$|\Pr[\mathsf{Win}_5] - \Pr[\mathsf{Win}_4]| = \epsilon^{\mathsf{MUC}^+}_{\mathsf{KEM},\mathcal{B},\mu,q_e}.$$

Note that Game 5 is identical to the MUC^+ PKE security experiment from Definition 2 executed with $\beta = 1$. Then, we have

$$|\Pr[\mathsf{Win}_0] - \Pr[\mathsf{Win}_5]| = \epsilon^{\mathsf{MUC}^+}_{\mathsf{PKE},\mathcal{A},\mu,q_e} \leq 2\epsilon^{\mathsf{MUC}^+}_{\mathsf{KEM},\mathcal{B},\mu,q_e} + \epsilon^{\mathsf{N\text{-}MIOT\text{-}IND}}_{\mathsf{ADEM},\mathcal{C},N} + 2\binom{N}{2} p.$$

□

Claim 1 The advantage of adversary $\mathcal{B}$ for breaking the MUC^+ security of KEM is exactly $|\Pr[\mathsf{Win}_1] - \Pr[\mathsf{Win}_0]|$.

Proof Let $\mathcal{A}$ be a distinguisher between Game 1 and Game 0. We now show how to construct an adversary $\mathcal{B}$ for breaking the MUC^+ security of KEM.

The adversary $\mathcal{B}$ receives the μ public keys $\{\mathsf{PK}^{(i)}\}_{i\in[\mu]}$ as input, as stated in Definition 1. Then, $\mathcal{A}$ runs as distinguisher by simulating Game 0 with the following changes:

- As the challenger, $\mathcal{B}$ sends μ public keys $\{\mathsf{PK}^{(i)}\}_{i\in[\mu]}$ to $\mathcal{A}$.
- When asked to resolve an encryption query on (i, m_0, m_1) from $\mathcal{A}$, $\mathcal{B}$ makes an encapsulation query on i and receives $(\mathsf{K}, \mathsf{C}_1)$ from its MUC^+ KEM challenger. Then, $\mathcal{B}$ sends the challenge ciphertext $\mathsf{CT} = (\mathsf{C}_1, \mathsf{C}_2)$, where $\mathsf{C}_2 \leftarrow \mathsf{ADEM.Enc}(\mathsf{K}, \mathsf{C}_1, m_0)$. $\mathcal{B}$ lists up encapsulation queries by defining $(\mathsf{K}, \mathsf{C}_1, i) \in \mathsf{Chlist}$.

- When asked to resolve a corruption query on i from $\mathcal{A}$, $\mathcal{B}$ makes a corruption query on i and receives SK_i from its MUC^+ KEM challenger. Then, $\mathcal{B}$ sends a private key for user i to $\mathcal{A}$.
- When asked to resolve a decryption query on $(\mathsf{CT} = (\mathsf{C}_1, \mathsf{C}_2), i)$ from $\mathcal{A}$, $\mathcal{B}$ returns $\mathsf{ADEM.Dec}(\mathsf{K}, \mathsf{C}_1, \mathsf{C}_2)$ if there exists a tuple such that $(\mathsf{K}, \mathsf{C}_1, i) \in \mathsf{Chlist}$. Otherwise, $\mathcal{B}$ makes a decapsulation query on (C_1, i), and receives K from its challenger, and returns $\mathsf{ADEM.Dec}(\mathsf{K}, \mathsf{C}_1, \mathsf{C}_2)$.

Eventually, $\mathcal{A}$ outputs a guess. If the guess is “Game 1”, then $\mathcal{B}$ outputs 0, which means that K is randomly chosen in $\mathcal{K}$; otherwise, it outputs 1.

Note that, in the above simulation, if challenges come from a MUC^+ KEM game executed with $\beta = 0$ (resp, $\beta = 1$), then the challenges transferred to $\mathcal{A}$ are the same as in Game 1 (resp, Game 0). Moreover, it holds that $i \in \mathsf{Klist}_{\mathsf{KEM},\mathcal{B}}$ when $i \in \mathsf{Klist}_{\mathsf{PKE},\mathcal{A}}$. Hence, $\mathcal{B}$ can simulate $\mathcal{A}$ without aborting. In other words, the probability of $\mathcal{A}$ distinguishing between Game 0 and Game 1 is exactly same as the probability of $\mathcal{B}$ to win. Therefore, we have

$$|\Pr[\mathsf{Win}_1] - \Pr[\mathsf{Win}_0]| = \epsilon^{\mathsf{MUC}^+}_{\mathsf{KEM},\mathcal{B},\mu,q_e}.$$

□

Claim 2 The advantage of adversary $\mathcal{C}$ for breaking the $\mathsf{N\text{-}MIOT\text{-}IND}$ security of ADEM is exactly $|\Pr[\mathsf{Win}_3] - \Pr[\mathsf{Win}_2]|$.

Proof Let $\mathcal{A}$ be a distinguisher between Game 3 and Game 2. We now show how to construct an adversary $\mathcal{C}$ for breaking the $\mathsf{N\text{-}MIOT\text{-}IND}$ security of ADEM. The adversary $\mathcal{C}$ runs $\mathcal{A}$ as a distinguisher by simulating Game 2 with the following changes:

- As the challenger, $\mathcal{C}$ initializes an integer $j \leftarrow 0$ and empty sets Clist and Klist.
- When asked to resolve encryption query on (i, m_0, m_1) from $\mathcal{A}$, if $i \in \mathsf{Klist}$, then $\mathcal{C}$ aborts. Otherwise, it computes $(\mathsf{K}, \mathsf{C}_1) \leftarrow \mathsf{KEM.Encap}(\mathsf{PK}_i)$, makes an encapsulation query on $(i, \mathsf{C}_1, m_0, m_1)$, and receives C_2 from its $\mathsf{N\text{-}MIOT\text{-}IND}$ ADEM challenger. Then, it sets $\mathsf{index}[i, \mathsf{C}_1] \leftarrow j$ and $j \leftarrow j + 1$, appends $(\mathsf{CT} = (\mathsf{C}_1, \mathsf{C}_2), i)$ to Clist, and returns challenge ciphertext CT to $\mathcal{A}$.
- When asked to resolve a corruption query on i from $\mathcal{A}$, if $(\mathsf{CT}, i) \in \mathsf{Clist}$ for any CT, $\mathcal{C}$ aborts. Otherwise, it returns the private key for user SK_i to $\mathcal{A}$.
- When asked to resolve a decryption query on (CT, i) from $\mathcal{A}$, if $(\mathsf{CT}, i) \in \mathsf{Clist}$, $\mathcal{C}$ aborts. Otherwise, if $\mathsf{index}[i, \mathsf{C}_1] \neq \perp$, $\mathcal{C}$ makes an decapsulation query on $(\mathsf{index}[i, \mathsf{C}_1], \mathsf{C}_2)$, receives m from its ADEM challenger, and returns m to $\mathcal{A}$. Otherwise, it returns whatever $\mathsf{ADEM.Dec}(\mathsf{K}, \mathsf{C}_1, \mathsf{C}_2)$ returns, where $\mathsf{K} \leftarrow \mathsf{KEM.Decap}(\mathsf{SK}_i, \mathsf{C}_1)$.

Eventually $\mathcal{A}$ outputs a guess. If the guess is “Game 3”, then $\mathcal{C}$ outputs $\beta' = 1$; otherwise it outputs $\beta' = 0$.

Note that, in the above simulation, if challenges come from a $\mathsf{N\text{-}MIOT\text{-}IND}$ ADEM game executed with $\beta = 0$ (resp, $\beta = 1$), then the challenges transferred to $\mathcal{A}$ are the same as in Game 2 (resp, Game 3). Additionally, the adversary $\mathcal{A}$ makes at most q_e encryption queries per user, which corresponds to at most μq_e encapsulation queries. Hence, $\mathcal{C}$ is correctly simulating Game 2 or Game 3 (depending on the bit β) without any violation of the restrictions on the number of queries. In other words, the probability of $\mathcal{A}$ to distinguish between Game 2 and Game 3 is exactly the same as the probability of $\mathcal{B}$ to win. Therefore, we have

$$|\Pr[\mathsf{Win}_3] - \Pr[\mathsf{Win}_2]| = \epsilon^{\mathsf{N\text{-}MIOT\text{-}IND}}_{\mathsf{ADEM},\mathcal{C},\mu q_e}.$$

□

Note that the maximum ciphertext-collision probability of our KEM scheme over all public keys is negligible because a random coin for encryption is chosen from $\mathbb{Z}_p^2$.

Acknowledgements The study was funded by Institute for Information and communications Technology Promotion (Grant No. 2016-6-00600, A Study on Functional Encryption: Construction, Security Analysis, and Implementation).

References

1. Attrapadung N., Furukawa J., Gomi T., Hanaoka G., Imai H., Zhang R.: Efficient identity-based encryption with tight security reduction. In: Pointcheval D., Mu Y., Chen K. (eds.) CANS, vol. 4301, pp. 19–36. Lecture Notes in Computer ScienceSpringer, Berlin (2006).
2. Attrapadung N., Hanaoka G., Yamada S.: A framework for identity-based encryption with almost tight security. In: Iwata T., Cheon J.H. (eds.) ASIACRYPT, vol. 9452, pp. 521–549. Lecture Notes in Computer ScienceSpringer, Berlin (2015).
3. Bader, C., Hofheinz, D., Jager, T., Kiltz, E., Li, Y.: Tightly-secure authenticated key exchange. In: TCC'15, Springer, Lecture Notes in Computer Science, vol. 9014, pp. 629–658 (2015)
4. Bellare M., Rogaway P.: Introduction to Modern Cryptography. University of California at San Diego, San Diego (2005).
5. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among notions of security for public-key encryption schemes. In: CRYPTO'98, vol. 1462, pp. 26–45. Springer (1998)
6. Bellare M., Boldyreva A., Micali S.: Public-key encryption in a multi-user setting: security proofs and improvements. In: Preneel B. (ed.) EUROCRYPT'00, vol. 1807, pp. 259–274. Lecture Notes in Computer ScienceSpringer, Berlin (2000).
7. Boneh D., Franklin M.K.: Identity-based encryption from the weil pairing. In: Kilian J. (ed.) CRYPTO'01, vol. 2139, pp. 213–229. Lecture Notes in Computer ScienceSpringer, Berlin (2001).
8. Boneh D., Canetti R., Halevi S., Katz J.: Chosen-ciphertext security from identity-based encryption. SIAM J. Comput. **36**(5), 1301–1328 (2007).
9. Cash D., Kiltz E., Shoup V.: The twin Diffie–Hellman problem and applications. In: Smart N. (ed.) EUROCRYPT, vol. 4965, pp. 127–145. Lecture Notes in Computer ScienceSpringer, Berlin (2008).
10. Cramer R., Shoup V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. CRYPTO'98, Lecture Notes in Computer Science, vol. 1462, pp. 13–25. Springer, Berlin (1998).
11. Diffie W., Hellman M.E.: New directions in cryptography. IEEE Trans. Inf. Theory **22**(6), 644–654 (2006).
12. Dolev D., Dwork C., Naor M.: Nonmalleable cryptography. SIAM J. Comput. **30**(2), 391–437 (2000).
13. Fujisaki E., Okamoto T.: How to enhance the security of public-key encryption at minimum cost. In: Imai H., Zheng Y. (eds.) PKC, Lecture Notes in Computer Science, vol. 1560, pp. 53–68. Springer, Berlin (1999).
14. Gay R., Hofheinz D., Kiltz E., Wee H.: Tightly cca-secure encryption without pairings. In: Fischlin M., Coron J.S. (eds.) EUROCRYPT, Part I, Lecture Notes in Computer Science, vol. 9665, pp. 1–27. Springer, New York (2016).
15. Gay, R., Hofheinz, D., Kohl, L.: Kurosawa-desmedt meets tight security. In: CRYPTO'17, vol. 10403, pp. 133–160. Springer, Berlin (2017)
16. Giacon F., Kiltz E., Poettering B.: Hybrid encryption in a multi-user setting, revisited. In: Abdalla M., Dahab R. (eds.) PKC'18, Lecture Notes in Computer Science, vol. 10769, pp. 159–189. Springer, Berlin (2018).
17. Gjøsteen, K., Jager, T.: Practical and tightly-secure digital signatures and authenticated key exchange. In: CRYPTO'18, Lecture Notes in Computer Science, vol. 10992, pp. 95–125. Springer (2018)
18. Gong J., Chen J., Dong X., Cao Z., Tang S.: Extended nested dual system groups, revisited. In: Cheng C.M., Chung K.M., Persiano G., Yang B.Y. (eds.) PKC'16, Lecture Notes in Computer Science, vol. 9614, pp. 133–163. Springer, Berlin (2016).
19. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: EUROCRYPT'08, Lecture Notes in Computer Science, vol. 4965, pp. 415–432. Springer (2008)
20. Hofheinz, D.: Adaptive partitioning. In: EUROCRYPT'17, Lecture Notes in Computer Science, vol. 10212, pp. 489–518. Springer (2017)
21. Hofheinz D., Jager T.: Tightly secure signatures and public-key encryption. In: Safavi-Naini R., Canetti R. (eds.) CRYPTO, Lecture Notes in Computer Science, vol. 7417, pp. 590–607. Springer, Cham (2012).

22. Hofheinz D., Koch J., Striecks C.: Identity-based encryption with (almost) tight security in the multi-instance, multi-ciphertext setting. In: Katz J. (ed.) PKC, Lecture Notes in Computer Science, vol. 9020, pp. 799–822. Springer, Cham (2015).
23. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the fujisaki-okamoto transformation. In: TCC'17, Lecture Notes in Computer Science, vol. 10677, pp. 341–371. Springer (2017)
24. Libert, B., Joye, M., Yung, M., Peters, T.: Concise multi-challenge cca-secure encryption and signatures with almost tight security. In: ASIACRYPT'14, Lecture Notes in Computer Science, vol. 8874, pp. 1–21. Springer (2014)
25. Libert B., Peters T., Joye M., Yung M.: Compactly hiding linear spans. In: Iwata T., Cheon J.H. (eds.) ASIACRYPT, Part I, Lecture Notes in Computer Science, vol. 9452, pp. 681–707. Springer, Cham (2015).
26. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: H. Ortiz (ed.) STOC'90, pp. 427–437. ACM (1990)
27. Rackoff, C., Simon, D.R.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: CRYPTO'91, vol. 576, pp. 433–444. Springer (1991)
28. Shamir A.: Identity-based cryptosystems and signature schemes. In: Blakley G.R., Chaum D. (eds.) CRYPTO'84, Lecture Notes in Computer Science, vol. 196, pp. 47–53. Springer, Berlin (1984).