Trading Plaintext-Awareness for Simulatability to Achieve Chosen Ciphertext Security[⋆]

Takahiro Matsuda and Goichiro Hanaoka

National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan
{t-matsuda,hanaoka-goichiro}@aist.go.jp

Abstract. In PKC 2014, Dachman-Soled showed a construction of a chosen ciphertext (CCA) secure public key encryption (PKE) scheme based on a PKE scheme which simultaneously satisfies a security property called weak simulatability and (standard model) plaintext awareness (sPA1) in the presence of multiple public keys. It is not well-known if plaintext awareness for the multiple keys setting is equivalent to the more familiar notion of that in the single key setting, and it is typically considered that plaintext awareness is a strong security assumption (because to achieve it we have to rely on a "knowledge"-type assumption). In Dachman-Soled's construction, the underlying PKE scheme needs to be plaintext aware in the presence of $2k + 2$ public keys.

The main result in this work is to show that the strength of plaintext awareness required in the Dachman-Soled construction can be somehow "traded" with the strength of a "simulatability" property of other building blocks. Furthermore, we also show that we can "separate" the assumption that a single PKE scheme needs to be both weakly simulatable and plaintext aware in her construction. Specifically, in this paper we show two new constructions of CCA secure key encapsulation mechanisms (KEMs): Our first scheme is based on a KEM which is chosen plaintext (CPA) secure and plaintext aware only under the 2 keys setting, and a PKE scheme satisfying a "slightly stronger" simulatability than weak simulatability, called *trapdoor simulatability* (introduced by Choi et al. ASIACRYPT 2009). Our second scheme is based on a KEM which is 1-bounded CCA secure (Cramer et al. ASIACRYPT 2007) and plaintext aware only in the *single* key setting, and a trapdoor simulatable PKE scheme. Our results add new recipes for constructing CCA secure PKE/KEM from general assumptions (that are incomparable to those used by Dachman-Soled), and in particular show interesting trade-offs among building blocks with those used in Dachman-Soled's construction.

Keywords: public key encryption, key encapsulation mechanism, chosen ciphertext security, plaintext-awareness, trapdoor simulatability.

[⋆] An extended abstract of this paper appears in the proceedings of PKC 2016. This is the full version.

Table of Contents

1 Introduction

1.1 Background and Motivation

For public key encryption (PKE), security (indistinguishability) against chosen ciphertext attacks (CCA) [47, 50, 24] is nowadays considered as a de-facto standard security notion required in most practical situations/applications in which PKE schemes are used. CCA security is quite important in both practical and theoretical points of view. It implies security against practical attacks (e.g. Bleichenbacher's attack [9]) and it also implies very strong and useful security notions, such as non-malleability [24] and universal composability [11]. Thus, constructing and understanding CCA secure PKE schemes is one of the central research themes in the area of cryptography. In this paper, we focus on the constructions of CCA secure PKE schemes and its closely related primitive called *key encapsulation mechanism* (KEM) from general cryptographic assumptions. There have been a number of works that show that several different kinds of cryptographic primitives are sufficient to realize CCA secure PKE/KEM: These include trapdoor permutations [24] (with some enhanced property [28]), identity-based encryption [13] and a weaker primitive called tag-based encryption [33], lossy trapdoor function [49] and related primitives [52, 43, 34, 56, 14], PKE schemes with weaker-than-but-close-to-CCA security [32, 35, 41], positive results on cryptographic obfuscation [54, 39], the combination of a CPA secure PKE scheme and a strong form of hash functions [40], and very recently, the combination of a sender non-committing encryption scheme and a key-dependent-message secure symmetric key encryption (SKE) scheme [42]. (We review more works in Section 1.4.)

In PKC 2014, Dachman-Soled [19] showed a construction of a CCA secure PKE scheme based on a PKE scheme which simultaneously satisfies a security property called weak simulatability [21, 44] and (standard model) plaintext awareness (sPA1) [6] in the presence of multiple public keys [44], which is based on the earlier work by Myers, Sergi, and Shelat [44] who showed a construction of a PKE scheme that achieves security slightly weaker than CCA (the so-called cNM-CCA1 security). Plaintext awareness was first introduced by Bellare and Rogaway [8] as a useful notion for showing CCA security of a PKE scheme in the random oracle model [7], and was used in a number of random-oracle-model constructions (e.g. [8, 25, 26, 48]). Bellare and Palacio [6] defined the standard model versions of plaintext awareness.[1] The plaintext awareness notions were further studied by subsequent works (e.g. [21]). The most works on plaintext awareness studied the notions for the single key setting. The extension to the multiple keys setting was first introduced by Myers, Sergi, and Shelat [44].

We note that it is not well-known or well-studied if plaintext awareness for the multiple keys setting is equivalent to the more familiar notion of plaintext awareness in the single key setting, and it is typically considered that plaintext awareness is a strong security assumption (because to achieve it we have to rely on a "knowledge"-type assumption). In the construction of [19], the underlying PKE scheme needs to be plaintext aware in the presence of $2k + 2$ public keys. Our motivation in this work is to clarify whether we can weaken the assumption of plaintext awareness in Dachman-Soled's construction [19]. As mentioned in [19], a plaintext aware (sPA1) PKE scheme seems almost like a CCA1 secure PKE scheme [47], but it seems not possible to replace the building block PKE scheme in [19] with a CCA1 secure scheme to remove the plaintext awareness. It is currently not known if we can construct a CCA secure PKE scheme only from a CPA secure scheme or even from

[1] [6] defined several versions (PA0, PA1, and PA2, with their computational/statistical/perfect variants) for standard model plaintext awareness. As in the previous works [44, 19], we focus on the statistical PA1 notion in the multiple keys setting (denoted by "$\mathtt{sPA1}_\ell$", where ℓ denotes the number of public keys).

a CCA1 secure scheme. We believe that studying the possibility of weakening the assumption of plaintext awareness from [19] thus is expected to lead to deepening our knowledge on this topic, and generally contribute to the long line of research on clarifying the minimal general assumption that implies CCA secure PKE.

1.2 Our Contributions

Based on the motivation mentioned above, we study the possibility of weakening the requirements of plaintext awareness used in Dachman-Soled's construction [19], and come up with new results that show that the strength of plaintext awareness required in [19] can be somehow "traded" with the strength of a "simulatability" property of other building blocks. Furthermore, we also show that we can "separate" the requirement that a single PKE scheme needs to be simultaneously weakly simulatable and plaintext aware, in her construction.

Specifically, in this paper we show two new constructions of CCA secure KEMs (which are given in Section 4), based on the assumptions that are *incomparable* to those used in [19]:

- Our first construction (Section 4.1) is based on a KEM which is chosen plaintext (CPA) secure and plaintext aware only under the 2 keys setting[2], and a PKE scheme satisfying a "slightly stronger" simulatability than weak simulatability, called "trapdoor simulatability" (introduced by Choi et al. [15]). Actually, although we write that it is "slightly stronger", it is formally *incomparable* to weak simulatability. For more details, see Section 1.3.
- Our second construction (Section 4.2) is based on a KEM which is 1-bounded CCA secure [16] and plaintext aware only in the *single* key setting, and a trapdoor simulatable PKE scheme. We can in fact slightly weaken the requirement of 1-bounded CCA security to CPA security in the presence of one "plaintext-checking" query [48, 1]. We will also show that we can construct a KEM satisfying simultaneously 1-bounded CCA security and plaintext awareness under the single key setting, based on a KEM satisfying CPA security and plaintext awareness under the $2k$ keys setting, via the recent result by Dodis and Fiore [22, Appendix C].

  One may wonder the meaning of the second construction, because if we use a KEM that is plaintext aware under $O(k)$ keys setting, there is no merit compared to our first construction. We are however considering it to be still meaningful in several aspects, and we refer the reader to Section 4.2 for more discussions regarding the second construction.

Note that from CCA secure KEMs, we can immediately obtain full-fledged PKE schemes by using CCA secure SKE [17].

We emphasize that we do not require plaintext awareness and the trapdoor simulatability property to be satisfied by a single building block. This "separation" of the requirements should be contrasted with Dachman-Soled's construction [19], the building block PKE scheme of which is required to satisfy plaintext awareness and the weak simulatability property simultaneously. We also again emphasize that the assumptions on which both of our constructions are based, are *incomparable* to those used in [19]. Thus, our results add new recipes for constructing CCA secure PKE/KEM from general assumptions (and thus the assumptions that we use could be new targets that are worth pursuing), and also show interesting trade-offs regarding assumptions with Dachman-Soled's construction.

[2] Plaintext awareness for KEMs is defined analogously to that for PKE. See Section 2.1.

1.3 Technical Overview

Assumptions on the Building Blocks. Trapdoor simulatable PKE (TSPKE) [15] is the key building block for our constructions. TSPKE is a *weaker* (relaxed) version of *simulatable PKE* that was originally formalized by Damgård and Nielsen [20]. Simulatable PKE admits "oblivious sampling" of both public keys and ciphertexts (i.e. sampling them without knowing the randomness or plaintext) in such a way that honestly generated public keys and ciphertexts can be later convincingly explained that they were generated obliviously. These properties are realized by requiring that the key generation algorithm and the encryption algorithm have their own "oblivious sampling" algorithm and its corresponding "inverting" algorithm (where the inverting algorithm corresponds to the algorithm that "explains" that an honestly generated public key (or a ciphertext) is sampled obliviously). The difference between TSPKE and simulatable PKE is whether we allow the "inverting" algorithm to take the randomness (and the plaintext) used by the ordinary algorithms (key generation and encryption algorithms) as input. TSPKE allows to take these inputs, while ordinary simulatable PKE does not, which makes the security property of TSPKE weaker but easier to achieve. For our purpose, we only need even a simplified version of TSPKE than the formalization in [15]: we only require a pair (pk, c) of public key/ciphertext (or, a "transcript") can be obliviously sampled, but not each of pk and c can be so (which is the formalization in [15]). It was shown [20, 15] that we can realize TSPKE from a number of standard cryptographic assumptions, such as the computational and decisional Diffie-Hellman assumptions, RSA, Factoring, and lattice based assumptions. (For more details, see Section 2.2.)

On the other hand, a weakly simulatable PKE scheme (used in the constructions in [44, 19]) considers oblivious sampling only for the encryption algorithm. However, the definition of weakly simulatable PKE used in [44, 19] does not allow the inverting algorithms to take the randomness and the plaintext used by the ordinary encryption algorithm. Therefore, strictly speaking, the "strength" of these primitives as "general cryptographic assumptions" are actually *incomparable*. Nonetheless, the reason why we still think that weakly simulatable PKE could be viewed as a weaker primitive, is that it does not require the key generation algorithm to be obliviously samplable. In fact, this difference is very important for our work. It is this simple difference between TSPKE and weakly simulatable PKE that enables us to weaken the plaintext awareness required in [19], from plaintext awareness in the presence of $O(k)$ keys in [19] into that under only $O(1)$ keys in our constructions.

Ideas for the Constructions. Other than employing TSPKE instead of weakly simulatable PKE, the ideas for our constructions and their security analyses are similar to those in [19]. In particular, the construction of [19] and our constructions are based on the Dolev-Dwork-Naor (DDN) construction [24], but we do not require a non-interactive zero-knowledge proof to ensure the validity of a ciphertext. Instead, the approach of the "double-layered" construction of Myers and Shelat [45] (and its simplifications [32, 38, 41] and variants [39, 40, 42]) is employed, in which a ciphertext consists of the "inner"-layer and "outer"-layer, and the randomness used for generating an outer ciphertext is somehow embedded into an inner ciphertext, so that in the decryption, the validity of the outer ciphertext can be checked by "re-encryption" using the randomness recovered from the inner ciphertext. (In our constructions, the inner-layer encryption is done by a KEM.) In fact, we do a simplification to [19] by removing a one-time signature scheme in [19], by using a commitment scheme, based on the ideas employed in the recent constructions [39, 40, 42].

Recently, Matsuda and Hanaoka [40] introduced the notion of *puncturable tag-based encryption* (PTBE) which abstracts and formalizes the "core" structure of the DDN construction [24]. We define the trapdoor simulatability property for PTBE (and call the primitive *trapdoor simulatable PTBE*) in Section 3, and use this primitive as an "intermediate" building block in our constructions. (This primitive could have other applications than constructing CCA secure PKE, and may be of independent interest.) There, we also show how to construct a trapdoor simulatable PTBE scheme from a TSPKE scheme. This construction is exactly the same as the construction of a PTBE scheme from a CPA secure PKE scheme used in [40], which is in turn based on the original DDN construction.

Ideas for the Security Proofs. We briefly recall the construction and the security proof in [19], and explain the difference in our proofs and that in [19]. As mentioned above, the construction of [19] is double-layered, where the outer encryption is like the "DDN-lite" construction (i.e. the DDN construction without a non-interactive zero-knowledge proof), and the inner encryption is a multiple-encryption by two PKE schemes. Both the inner and outer encryption schemes use the same building block, with independently generated public keys: $2k$ keys for the outer-layer encryption (that does DDN-lite-encryption) and 2 keys for the inner-layer encryption (that does multiple-encryption by two encryptions). Roughly speaking, in the security proof, [19] constructs a CPA adversary (reduction algorithm) for the inner-layer encryption, from a CCA adversary $\mathcal{A}$ against the entire construction. The reduction algorithm of course has to somehow answer $\mathcal{A}$'s decryption queries, and this is the place where plaintext awareness comes into play. Plaintext awareness in the ℓ keys setting ($\mathtt{sPA1}_\ell$ security) ensures that for any algorithm $\mathcal{C}$ (called "ciphertext creator") that receives a set of public keys $(pk_i)_{i \in \{1,\dots,\ell\}}$ and a randomness $r_\mathcal{C}$ as input and makes decryption queries, there exists an extractor $\mathcal{E}$ that also receives $(pk_i)_{i \in \{1,\dots,\ell\}}$ and $r_\mathcal{C}$ as input, and can "extract" the plaintext from a ciphertext queried by $\mathcal{C}$. (In our actual security proofs, we denote the "ciphertext creator" by "$\mathcal{A}'$", but for the explanation here we continue to use $\mathcal{C}$ for clarity.) The idea in the proof in [19] is to use an extractor guaranteed by plaintext awareness to answer the CCA adversary $\mathcal{A}$'s decryption queries. The problem that arises here is: how do we design the algorithm $\mathcal{C}$ with which the extractor $\mathcal{E}$ is considered? Since the extractor $\mathcal{E}$ needs to be given the randomness $r_\mathcal{C}$ used by $\mathcal{C}$, if we naively design $\mathcal{C}$, the reduction algorithm cannot use the extractor $\mathcal{E}$ while embedding its instances (the public key and the challenge ciphertext) in the reduction algorithm's CPA security experiment into $\mathcal{A}$'s view. The approach in [19] is to consider a modified version of the CCA security experiment in which all component ciphertexts (i.e. ciphertexts for the outer-layer encryption) are generated obliviously using some randomness r (which can be performed due to the weak simulatability property of the underlying PKE scheme), and view this modified experiment as a ciphertext creator $\mathcal{C}$ that takes as input $\ell = 2k + 2$ public keys (for both inner-/outer-layer encryptions) and a randomness $r_\mathcal{C}$ consisting of the randomness $r_\mathcal{A}$ used by $\mathcal{A}$ and the randomness r used for oblivious generation of the component ciphertexts in $\mathcal{A}$'s challenge ciphertext. ($r_\mathcal{C}$ actually also contains some additional randomness used for generating the remaining parts of $\mathcal{A}$'s challenge ciphertext, but we ignore it here for simplicity.) Designing the algorithm $\mathcal{C}$ in this way, the extractor $\mathcal{E}$ corresponding to $\mathcal{C}$ can be used to answer $\mathcal{A}$'s decryption queries while the reduction algorithm (attacking the CPA security of the inner-layer encryption) can perform the reduction.

Our main idea for weakening the requirement of plaintext awareness for the building blocks, from $2k + 2$ keys in [19] to $O(1)$ keys, is due to the observation that by relying on the trapdoor simulatability property for the outer-layer encryption, we can "push" the public keys for the outer-layer encryption, into the "randomness" $r_\mathcal{C}$ for the ciphertext creator $\mathcal{C}$ (with which the extractor $\mathcal{E}$ is considered), by generating the public keys regarding the outer-layer encryption also obliviously. In order to make this idea work, we thus consider a different design strategy for the ciphertext creator

$\mathcal{C}$. This also enables us to "separate" the requirement that a single building block PKE scheme needs to be simultaneously plaintext aware and simulatable, because we need the simulatability only for the outer-layer encryption.

Actually, like the security proof of the construction in [19], we need to deal with a "bad" decryption query, which is a ciphertext such that its actual decryption result (by the normal decryption algorithm with a secret key) differs from the decryption result obtained by using the extractor $\mathcal{E}$. (Such a decryption query makes the simulation of the decryption oracle by the reduction algorithm fail.) Our first construction uses the clever trick of Dachman-Soled [19] of using two CPA secure PKE schemes (that each encrypts a "share" of 2-out-of-2 secret sharing) and their plaintext awareness under 2 keys setting. (As mentioned earlier, in fact, we use a KEM instead of a PKE scheme for the inner encryption.) Dachman-Soled's approach enables us to use the CPA security and the ability of "detecting" bad queries at the same time. Our second construction is a simplification of our first construction, where we employ a "single" KEM for the inner layer, as opposed to multiple-encryption by two KEMs in our first construction. To detect "bad" decryption queries by an adversary, we employ the ideas and techniques from [45, 32, 38, 41] of using "1-bounded CCA" security [16]. (As mentioned earlier, in fact, CPA security in the presence of one "plaintext-checking" query [48, 1] is sufficient for our purpose.) For more details on these, see Section 4.

1.4 Related Work

The notion of CCA security for PKE was formalized by Naor and Yung [47] and Rackoff and Simon [50]. Since the introduction of the notion, CCA secure PKE schemes have been studied in a number of papers, and thus we only briefly review constructions from general cryptographic assumptions. Dolev, Dwork, and Naor [24] showed the first construction of a CCA secure PKE scheme, from a CPA secure scheme and a NIZK proof system, based on the construction by Naor and Yung [47] that achieves weaker non-adaptive CCA (CCA1) security. These NIZK-based constructions were further improved in [53, 55, 36]. Canetti, Halevi, and Katz [13] showed how to transform an identity-based encryption scheme into a CCA secure PKE scheme. Kiltz [33] showed that the transform of [13] is applicable to a weaker primitive of tag-based encryption (TBE). Peikert and Waters [49] showed how to construct a CCA secure PKE scheme from a *lossy* trapdoor function (TDF). Subsequent works showed that TDFs with weaker security/functionality properties are sufficient for obtaining CCA secure PKE schemes [52, 43, 34, 56, 14]. Hemenway and Ostrovsky [30] showed how to construct a CCA secure scheme in several ways from homomorphic encryption that has some appropriate properties, and the same authors [31] showed that one can construct a CCA secure PKE scheme from a lossy encryption scheme [5] if it can encrypt a plaintext longer than the length of randomness consumed by the encryption algorithm. Myers and Shelat [45] showed that a CCA secure PKE scheme for 1-bit messages can be turned into one with an arbitrarily large plaintext space. Hohenberger, Lewko, and Waters [32] showed that CCA secure PKE can be constructed from a PKE with a weaker security notion called detectable CCA security, from which we can obtain a 1-bit-to-multi-bit transformation for CCA security in a simpler manner than [45]. The simplicity and efficiency of [45] were further improved by Matsuda and Hanaoka [38, 41]. Lin and Tessaro [35] showed how to amplify weak CCA security into strong (ordinary) CCA secure one. Matsuda and Hanaoka [39] showed how to construct a CCA secure PKE scheme by using a CPA secure PKE scheme and point obfuscation [10, 37], and the same authors [40] showed a CCA secure PKE scheme from a CPA secure PKE scheme and a family of hash functions satisfying the very strong security notion called universal computational extractors (UCE) [3]. The same authors [42] recently also showed that a CCA secure PKE scheme can be built from the combination of a sender non-committing encryption scheme and a key-dependent-message secure SKE scheme. More recently, Hajiabadi and Kapron [29] showed how to construct a CCA secure PKE scheme, from a 1-bit PKE scheme that satisfies circular security and has the structural property called reproducibility.

As has been stated several times, Dachman-Soled [19] showed how to construct a CCA secure PKE scheme from a PKE scheme which simultaneously satisfies weak simulatability [44] and the (standard model) plaintext awareness under the multiple keys setting, which is built based on the result by Myers, Sergi, and Shelat [44] who showed a PKE scheme satisfying the so-called cNM-CCA1 security, from the same building blocks as [19]. Sahai and Waters [54] showed (among other cryptographic primitives) how CCA secure PKE and KEMs can be constructed using an indistinguishability obfuscation [2, 27].

1.5 Paper Organization

In Section 2 and Appendix A, we review basic notation and definitions of cryptographic primitives that are used in this paper. In Section 3, we introduce the notion of trapdoor simulatable PTBE, which is an extension of PTBE introduced in [40], and works as one of building blocks of our proposed KEMs in the next section. Finally, in Section 4, we show our main results: two constructions of KEMs that show a trade-off between "simulatability" property and "plaintext awareness" in Dachman-Soled's construction [19].

2 Preliminaries

In this section, we review the basic notation, and the definitions for plaintext awareness ($\mathtt{sPA1}_\ell$ security) [6, 44, 19] of a KEM, trapdoor simulatability properties of a PKE scheme and a commitment scheme, and the syntax of a puncturable tag-based encryption (PTBE) scheme, which are central to the results in this paper. The definitions for standard cryptographic primitives with standard security definitions that are not reviewed in this section are given in Appendix A, which include PKE, KEMs, universal one-way hash functions (UOWHFs), and signatures.

Basic Notation. $\mathbb{N}$ denotes the set of all natural numbers, and for $n \in \mathbb{N}$, we define $[n] := \{1, \dots, n\}$. "$x \leftarrow y$" denotes that x is chosen uniformly at random from y if y is a finite set, x is output from y if y is a function or an algorithm, or y is assigned to x otherwise. If x and y are strings, then "$|x|$" denotes the bit-length of x, "$x\|y$" denotes the concatenation x and y, and "$(x \overset{?}{=} y)$" is the operation which returns 1 if $x = y$ and 0 otherwise. "(P)PTA" stands for a *(probabilistic) polynomial time algorithm.* For a finite set S, "$|S|$" denotes its size. If $\mathcal{A}$ is a probabilistic algorithm ,then "$y \leftarrow \mathcal{A}(x; r)$" denotes that $\mathcal{A}$ computes y as output by taking x as input and using r as randomness, and we just write "$y \leftarrow \mathcal{A}(x)$" if we do not need to make the randomness used by $\mathcal{A}$ explicit. If furthermore $\mathcal{O}$ is a function or an algorithm, then "$\mathcal{A}^{\mathcal{O}}$" means that $\mathcal{A}$ has oracle access to $\mathcal{O}$. A function $\epsilon(k) : \mathbb{N} \to [0, 1]$ is said to be *negligible* if for all positive polynomials $p(k)$ and all sufficiently large $k \in \mathbb{N}$, we have $\epsilon(k) < 1/p(k)$. Throughout this paper, we use the character "k" to denote a security parameter.

2.1 Plaintext Awareness for Multiple Keys Setup ($\mathtt{sPA1}_\ell$ Security)

Here, we review the definition of (statistical) plaintext awareness for multiple key setup [44, 19]. Unlike these previous works, we define it for a KEM, rather than a PKE scheme, but we can define plaintext awareness for a KEM in essentially the same way as that for a PKE scheme.

Let $\Gamma = (\mathsf{KKG}, \mathsf{Encap}, \mathsf{Decap})$ be a KEM (where we review the definition of a KEM in Appendix A.2), and $\ell = \ell(k) > 0$ be a polynomial. Let $\mathcal{A}$ be an algorithm (called a "ciphertext creator") that takes a set of public keys $(pk_i)_{i \in [\ell]}$ as input, and makes decapsulation queries of the form $(j \in [\ell], c)$ which is supposed to be answered with $K = \mathsf{Decap}(sk_j, c)$. For this $\mathcal{A}$, we consider the corresponding "(plaintext) extractor" $\mathcal{E}$: It is a stateful algorithm that initially takes a set of public keys $(pk_i)_{i \in [\ell]}$ and the randomness $r_{\mathcal{A}}$ consumed by $\mathcal{A}$, and expects to receive "decapsulation" queries of the form $q = (j \in [\ell], c)$; Upon a query, it tries to extract a session-key K corresponding to c so that $K = \mathsf{Decap}(sk_j, c)$, where sk_j is the secret key corresponding to pk_j. After $\mathcal{E}$ extracts a session-key, it may update its internal state to prepare for the next call. Informally, a KEM Γ is said to be $\mathtt{sPA1}_\ell$ secure if for all PPTA ciphertext creators $\mathcal{A}$, there exists a corresponding PPTA extractor $\mathcal{E}$ that can work as $\mathcal{A}$'s decapsulation oracle in the experiment above.

More formally, for $\mathcal{A}$ that makes $Q = Q(k)$ decapsulation queries, $\mathcal{E}$, and ℓ, consider the following experiment $\mathsf{Expt}^{\mathtt{sPA1}}_{\Gamma,\mathcal{A},\mathcal{E},\ell}(k)$:

$$\begin{aligned}\mathsf{Expt}^{\mathtt{sPA1}}_{\Gamma,\mathcal{A},\mathcal{E},\ell}(k) : [\ & \forall i \in [\ell] : (pk_i, sk_i) \leftarrow \mathsf{KKG}(1^k);\ r_{\mathcal{A}} \leftarrow \{0,1\}^*;\ \mathsf{st}_{\mathcal{E}} \leftarrow ((pk_i)_{i \in [\ell]}, r_{\mathcal{A}}); \\ & \text{Run } \mathcal{A}^{\mathcal{E}(\mathsf{st}_{\mathcal{E}}, \cdot)}((pk_i)_{i \in [\ell]}; r_{\mathcal{A}}) \text{ until it terminates;} \\ & \text{If } \exists i \in [Q] : \mathsf{Decap}(sk_{j_i}, c_i) \neq K_i \text{ then return 1 else return 0.}],\end{aligned}$$

where (j_i, c_i) represents $\mathcal{A}$'s i-th decapsulation query (which $\mathcal{A}$ expects to be decapsulated as a ciphertext under pk_{j_i}), and K_i represents the answer (i.e. "decapsulation result" of c_i) computed by the algorithm $\mathcal{E}$. In the experiment, $\mathcal{E}$ is the (possibly stateful) extractor which initially takes $\mathsf{st}_{\mathcal{E}} = ((pk_i)_{i \in [\ell]}, r_{\mathcal{A}})$ as input, and works like $\mathcal{A}$'s decapsulation oracle, as explained above.

Definition 1. *Let $\ell = \ell(k) > 0$ be a polynomial. We say that a KEM Γ is $\mathtt{sPA1}_\ell$ secure if for all PPTAs (ciphertext creator) $\mathcal{A}$, there exists a stateful PPTA (extractor) $\mathcal{E}$ such that $\mathsf{Adv}^{\mathtt{sPA1}}_{\Gamma,\mathcal{A},\mathcal{E},\ell}(k) := \Pr[\mathsf{Expt}^{\mathtt{sPA1}}_{\Gamma,\mathcal{A},\mathcal{E},\ell}(k) = 1]$ is negligible.*

If $\ell = 1$, then $\mathtt{sPA1}_\ell$ security is equivalent to statistical PA1 security defined by Bellare and Palacio [6]. By definition, trivially, $\mathtt{sPA1}_x$ implies $\mathtt{sPA1}_y$ for $x > y$. However, to the best of our knowledge, whether there is an implication (or separation) for the opposite direction, is not known.

2.2 (Simplified) Trapdoor Simulatable Public Key Encryption

A trapdoor simulatable PKE (TSPKE) [15] is a *relaxed* version of simulatable PKE [20]. Simulatable PKE admits "oblivious sampling" of both public keys and ciphertexts (i.e. sampling them without knowing the randomness or plaintext) in such a way that honestly generated public keys and ciphertexts can be later convincingly explained that they were generated obliviously.[3] These properties are realized by requiring that the key generation algorithm and the encryption algorithm have their own "oblivious sampling" algorithm and its corresponding "inverting" algorithm (where the inverting algorithm corresponds to the algorithm that explains that an honest generated public key (or a ciphertext) is sampled obviously). The difference between TSPKE and simulatable PKE, is whether we allow for the "inverting" algorithm to take the randomness (and the plaintext) used by the ordinary algorithms PKG and Enc as input. Since the "inverting" algorithm in TSPKE is allowed to see more information than that in simulatable PKE, the former primitive is strictly weaker (and easier to construct) than the latter.

[3] (Trapdoor) simulatable PKE scheme was introduced as a building block for constructing non-committing encryption [12].

```
$\mathsf{Expt}^{\mathtt{TSPKE\text{-}Real}}_{\Pi,\mathcal{A}}(k)$ :
    $(m, \mathsf{st}) \leftarrow \mathcal{A}_1(1^k)$
    $r_g, r_e \leftarrow \{0,1\}^*$
    $(pk, sk) \leftarrow \mathsf{PKG}(1^k; r_g)$
    $c \leftarrow \mathsf{Enc}(pk, m; r_e)$
    $\widehat{r} \leftarrow \mathsf{rSamp}_{\Pi}(r_g, r_e, m)$
    $b' \leftarrow \mathcal{A}_2(\mathsf{st}, pk, c, \widehat{r})$
    Return $b'$.
```

```
$\mathsf{Expt}^{\mathtt{TSPKE\text{-}Sim}}_{\mathcal{T},\mathcal{A}}(k)$ :
    $(m, \mathsf{st}) \leftarrow \mathcal{A}_1(1^k)$
    $\widehat{r} \leftarrow \{0,1\}^*$
    $(pk, c) \leftarrow \mathsf{oSamp}_{\Pi}(1^k; \widehat{r})$
    $b' \leftarrow \mathcal{A}_2(\mathsf{st}, pk, c, \widehat{r})$
    Return $b'$.
```

```
$\mathsf{Expt}^{\mathtt{TSPTBE\text{-}Real}}_{\mathcal{T},\mathcal{A}}(k)$ :
    $(\mathsf{tag}^*, m, \mathsf{st}) \leftarrow \mathcal{A}_1(1^k)$
    $r_g, r_e \leftarrow \{0,1\}^*$
    $(pk, sk) \leftarrow \mathsf{TKG}(1^k; r_g)$
    $c \leftarrow \mathsf{TEnc}(pk, \mathsf{tag}^*, m; r_e)$
    $\widehat{sk}_{\mathsf{tag}^*} \leftarrow \mathsf{Punc}(sk, \mathsf{tag}^*)$
    $\widehat{r} \leftarrow \mathsf{rSamp}_{\mathcal{T}}(r_{\mathsf{g}}, r_{\mathsf{e}}, \mathsf{tag}^*, m)$
    $b' \leftarrow \mathcal{A}_2(\mathsf{st}, pk, c, \widehat{sk}_{\mathsf{tag}^*}, \widehat{r})$
    Return $b'$.
```

```
$\mathsf{Expt}^{\mathtt{TSPTBE\text{-}Sim}}_{\mathcal{T},\mathcal{A}}(k)$ :
    $(\mathsf{tag}^*, m, \mathsf{st}) \leftarrow \mathcal{A}_1(1^k)$
    $\widehat{r} \leftarrow \{0,1\}^*$
    $(pk, c, \widehat{sk}_{\mathsf{tag}^*}) \leftarrow \mathsf{oSamp}_{\mathcal{T}}(\mathsf{tag}^*; \widehat{r})$
    $b' \leftarrow \mathcal{A}_2(\mathsf{st}, pk, c, \widehat{sk}_{\mathsf{tag}^*}, \widehat{r})$
    Return $b'$.
```

Fig. 1. Security experiments for defining security of TSPKE (upper-left and upper-right) and those for defining security of TSPTBE (bottom-left and bottom-right)

For our purpose, we only need even a simplified version of TSPKE of [15]: we only require a pair (pk, c) of public key/ciphertext (or, “transcript) can be obliviously sampled [15], but not each of pk and c can be so. A TSPKE scheme with such a simplified syntax may not be useful for constructing non-committing encryption (as done in [20, 15]), but sufficient for our purpose in this paper.

Definition 2. *We say that a PKE scheme*[4] $\Pi = (\mathsf{PKG}, \mathsf{Enc}, \mathsf{Dec})$ *is* trapdoor simulatable *(and say that Π is a trapdoor simulatable PKE (TSPKE) scheme) if Π has two additional PPTAs* $(\mathsf{oSamp}_{\Pi}, \mathsf{rSamp}_{\Pi})$ *with the following properties:*

- oSamp_{Π} *is the oblivious-sampling algorithm which takes 1^k as input, and outputs an “obliviously generated” public key/ciphertext pair (pk, c).*
- rSamp_{Π} *is the inverting algorithm (corresponding to* oSamp_{Π}*) that takes randomness r_g and r_e, and a plaintext m (which are supposed to be used as $(pk, sk) \leftarrow \mathsf{PKG}(1^k; r_g)$ and $c \leftarrow \mathsf{Enc}(pk, m; r_e)$) as input, and outputs a string $\widehat{r}$ (that looks like a randomness used by* oSamp_{Π}*).*
- ***(Trapdoor Simulatability)*** *For all PPTAs $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the advantage $\mathsf{Adv}^{\mathsf{TSPKE}}_{\Pi,\mathcal{A}}(k) := |\Pr[\mathsf{Expt}^{\mathtt{TSPKE\text{-}Real}}_{\Pi,\mathcal{A}}(k) = 1] - \Pr[\mathsf{Expt}^{\mathtt{TSPKE\text{-}Sim}}_{\Pi,\mathcal{A}}(k) = 1]|$ is negligible, where the experiments $\mathsf{Expt}^{\mathtt{TSPKE\text{-}Real}}_{\Pi,\mathcal{A}}(k)$ and $\mathsf{Expt}^{\mathtt{TSPKE\text{-}Sim}}_{\Pi,\mathcal{A}}(k)$ are defined as in Fig. 1 (upper-left and upper-right, respectively).*

Concrete Instantiations of TSPKE. Since our definition of TSPKE is a simplified (and hence weaker) version of the definition by Choi et al. [15], and TSPKE is a weaker primitive than a simulatable PKE scheme in the sense of Damgård and Nielsen [20], we can use any of (trapdoor) simulatable PKE schemes shown in these works. In particular, we can construct a TSPKE scheme from most of the standard cryptographic assumptions such as the computational and decisional Diffie-Hellman, RSA, factoring, and learning-with-errors assumptions [20, 15]. (For example, the ElGamal encryption, Damgård's ElGamal encryption, and Cramer-Shoup-Lite encryption schemes can be shown to be a TSPKE scheme if they are implemented in a simulatable group [21].) In terms of “general” cryptographic assumptions, Damgård and Nielsen [20] showed that a simulatable

[4] The syntax of PKE is reviewed in Appendix A.1.

PKE scheme can be constructed from a family of trapdoor permutations with the simulatability property, in which the key generation and the domain-sampling algorithms have the oblivious sampling property (which is defined analogously to simulatable PKE). Hence, we can also construct a TSPKE from it.

2.3 Trapdoor Simulatable Commitment Schemes

Let $\mathcal{C} = (\mathsf{CKG}, \mathsf{Com})$ be a commitment scheme. (We review the syntax of a commitment scheme and its "target-binding" property in Appendix A.3.)

We define the trapdoor simulatability property of a commitment scheme $\mathcal{C}$, which is defined in exactly the same way as the trapdoor simulatability of a PKE scheme. Namely, we require that there be the oblivious sampling algorithm $\mathsf{oSamp}_{\mathcal{C}}$ (for sampling a key/commitment pair (ck, c)) and the corresponding inverting algorithm $\mathsf{rSamp}_{\mathcal{C}}$, whose interfaces are exactly the same as oSamp_{Π} and rSamp_{Π} of a TSPKE scheme, respectively. We say that a commitment scheme $\mathcal{C}$ is *trapdoor simulatable* (and say that $\mathcal{C}$ is a trapdoor simulatable commitment scheme) if for all PPTA adversaries $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathtt{TSCom}}_{\mathcal{C},\mathcal{A}}(k) := |\Pr[\mathsf{Expt}^{\mathtt{TSCom\text{-}Real}}_{\mathcal{C},\mathcal{A}}(k) = 1] - \Pr[\mathsf{Expt}^{\mathtt{TSCom\text{-}Sim}}_{\mathcal{C},\mathcal{A}}(k) = 1]|$ is negligible, where the experiments $\mathsf{Expt}^{\mathtt{TSCom\text{-}Real}}_{\mathcal{C},\mathcal{A}}(k)$ and $\mathsf{Expt}^{\mathtt{TSCom\text{-}Sim}}_{\mathcal{C},\mathcal{A}}(k)$ are defined in exactly the same way as $\mathsf{Expt}^{\mathtt{TSPKE\text{-}Real}}_{\Pi,\mathcal{A}}(k)$ and $\mathsf{Expt}^{\mathtt{TSPKE\text{-}Sim}}_{\mathcal{C},\mathcal{A}}(k)$ for a TSPKE scheme, respectively (and thus we do not write down them).

We can achieve a commitment scheme which satisfies target-binding, trapdoor simulatability, and the requirement of the size of commitments (namely we require the size of commitments to be k-bit for k-bit security), only from a TSPKE scheme and a UOWHF, just by hashing a ciphertext of the TSPKE scheme by the UOWHF. We review this construction in Appendix B.

2.4 Puncturable Tag-Based Encryption

Here, we recall the syntax of puncturable tag-based encryption (PTBE), which was introduced by Matsuda and Hanaoka [40] as an abstraction of the "core" structure of the Dolev-Dwork-Naor (DDN) construction [24]. Similarly to [40], we use PTBE as an intermediate building block to reduce the description complexity of our proposed constructions in Section 4.

Intuitively, a PTBE scheme is a TBE scheme that has a mechanism for generating a "punctured" secret key $\widehat{sk}_{\mathsf{tag}^*}$, according to a "punctured point" tag tag^*. The punctured secret key can be used to decrypt all "honestly generated" ciphertexts that are generated under tags that are different from tag^*, while the punctured secret key is useless for decrypting ciphertexts generated under tag^*.

Formally, a PTBE scheme consists of the five PPTAs $(\mathsf{TKG}, \mathsf{TEnc}, \mathsf{TDec}, \mathsf{Punc}, \widehat{\mathsf{TDec}})$ among which the latter three algorithms are deterministic, with the following interface:

<u>**Key Generation:**</u>	<u>**Encryption:**</u>	<u>**Decryption:**</u>
$(pk, sk) \leftarrow \mathsf{TKG}(1^k)$	$c \leftarrow \mathsf{TEnc}(pk, \mathsf{tag}, m)$	m (or $\perp$) $\leftarrow \mathsf{TDec}(sk, \mathsf{tag}, c)$

<u>**Puncturing:**</u>	<u>**Punctured Decryption:**</u>
$\widehat{sk}_{\mathsf{tag}^*} \leftarrow \mathsf{Punc}(sk, \mathsf{tag}^*)$	m (or $\perp$) $\leftarrow \widehat{\mathsf{TDec}}(\widehat{sk}_{\mathsf{tag}^*}, \mathsf{tag}, c)$

where (pk, sk) is a public/secret key pair, c is a ciphertext of a plaintext m under pk and a tag $\mathsf{tag} \in \{0,1\}^k$, and $\widehat{sk}_{\mathsf{tag}^*}$ is a "punctured" secret key corresponding to a tag $\mathsf{tag}^* \in \{0,1\}^k$.

We require for all $k \in \mathbb{N}$, all tags $\mathsf{tag}^*, \mathsf{tag} \in \{0,1\}^k$ such that $\mathsf{tag}^* \neq \mathsf{tag}$, all (pk, sk) output from $\mathsf{TKG}(1^k)$, all plaintexts m, and all ciphertexts c output from $\mathsf{TEnc}(pk, \mathsf{tag}, m)$, it holds that

$\mathsf{TDec}(sk, \mathsf{tag}, c) = \widehat{\mathsf{TDec}}(\mathsf{Punc}(sk, \mathsf{tag}^*), \mathsf{tag}, c) = m$. (Note that correctness is guaranteed only for ciphertexts c generated from $\mathsf{TEnc}(pk, \mathsf{tag}, \cdot)$.)

In [40], the security notion called "extended CPA security" was defined as a security notion of PTBE. In our proposed KEMs, we need a stronger security property for PTBE, which is an analogue of TSPKE, and we will introduce it in the next section.

3 Trapdoor Simulatable PTBE

In this section, we define trapdoor simulatability of a PTBE scheme, in the same way as that of a PKE scheme and a commitment scheme. However, for the oblivious sampling algorithm, we let it take a "punctured point" tag tag^* as input, and require that it output the punctured secret key $\widehat{sk}_{\mathsf{tag}^*}$ (corresponding to tag^*) in addition to a public key/ciphertext pair (pk, c).

Formally, we define a trapdoor simulatable PTBE (TSPTBE) as follows:

Definition 3. *We say that a PTBE scheme $\mathcal{T} = (\mathsf{TKG}, \mathsf{TEnc}, \mathsf{TDec}, \mathsf{Punc}, \widehat{\mathsf{TDec}})$ is* trapdoor simulatable *(and say that $\mathcal{T}$ is a trapdoor simulatable PTBE (TSPTBE) scheme) if $\mathcal{T}$ has two additional PPTAs $(\mathsf{oSamp}_{\mathcal{T}}, \mathsf{rSamp}_{\mathcal{T}})$ with the following properties:*

- $\mathsf{oSamp}_{\mathcal{T}}$ *is the oblivious sampling algorithm which takes a "punctured point" tag tag^* as input, and outputs an "obliviously generated" public key/ciphertext pair (pk, c) and a punctured secret key $\widehat{sk}_{\mathsf{tag}^*}$.*
- $\mathsf{rSamp}_{\mathcal{T}}$ *is the inverting algorithm (corresponding to $\mathsf{oSamp}_{\mathcal{T}}$) that takes 1^k, randomness r_g and r_e, a "punctured point" tag tag^*, and a plaintext m (which are supposed to be used as $(pk, sk) \leftarrow \mathsf{TKG}(1^k; r_g)$ and $c \leftarrow \mathsf{TEnc}(pk, \mathsf{tag}^*, m; r_e)$) as input, and outputs a string $\widehat{r}$ (that looks like a randomness used by $\mathsf{oSamp}_{\mathcal{T}}$).*
- ***(Trapdoor Simulatability)*** *For all PPTAs $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the advantage $\mathsf{Adv}^{\mathtt{TSPTBE}}_{\mathcal{T},\mathcal{A}}(k) := |\Pr[\mathsf{Expt}^{\mathtt{TSPTBE\text{-}Real}}_{\mathcal{T},\mathcal{A}}(k) = 1] - \Pr[\mathsf{Expt}^{\mathtt{TSPTBE\text{-}Sim}}_{\mathcal{T},\mathcal{A}}(k) = 1]|$ is negligible, where the experiments $\mathsf{Expt}^{\mathtt{TSPTBE\text{-}Real}}_{\mathcal{T},\mathcal{A}}(k)$ and $\mathsf{Expt}^{\mathtt{TSPTBE\text{-}Sim}}_{\mathcal{T},\mathcal{A}}(k)$ are defined as in Fig. 1 (bottom-left and bottom-right, respectively).*

On the Existence of TSPTBE. Though it might look complicated, we can construct a TSPTBE scheme from a TSPKE scheme, by a Dolev-Dwork-Naor-style approach [24]. The construction is exactly the same as the construction of a PTBE scheme from any $\mathtt{CPA}$ secure PKE shown in [40], which is the "core" structure of the DDN construction, namely, the DDN construction without a NIZK proof and without its one-time signature. We show the following lemma in Appendix C.1.

Lemma 1. *If a TSPKE scheme exists, then so does a TSPTBE scheme.*

Useful fact. For the security proofs of our constructions in Section 4, we will use the fact that the straightforward concatenation of a "transcript" of a trapdoor simulatable commitment and that of a TSPTBE scheme, also admits the trapdoor simulatable property.

Specifically, for a TSPTBE scheme $\mathcal{T} = (\mathsf{TKG}, \mathsf{TEnc}, \mathsf{TDec}, \mathsf{Punc}, \widehat{\mathsf{TDec}}, \mathsf{oSamp}_{\mathcal{T}}, \mathsf{rSamp}_{\mathcal{T}})$ and a trapdoor simulatable commitment scheme $\mathcal{C} = (\mathsf{CKG}, \mathsf{Com}, \mathsf{oSamp}_{\mathcal{C}}, \mathsf{rSamp}_{\mathcal{C}})$ such that the plaintext space of $\mathcal{T}$ and that of $\mathcal{C}$ are identical, and for an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, consider the following "real" experiment $\mathsf{Expt}^{\mathtt{TS\text{-}Real}}_{[\mathcal{C},\mathcal{T}],\mathcal{A}}(k)$ and the "simulated" experiment $\mathsf{Expt}^{\mathtt{TS\text{-}Sim}}_{[\mathcal{C},\mathcal{T}],\mathcal{A}}(k)$ as described in Fig. 2 (left and right, respectively). Then, we can prove the following lemma, whose proof is almost straightforward due to the trapdoor simulatability property of $\mathcal{C}$ and $\mathcal{T}$, and is given in Appendix C.2 for self-containment.

```
$\mathsf{Expt}^{\mathtt{TS\text{-}Real}}_{[\mathcal{C},\mathcal{T}],\mathcal{A}}(k)$ :
    $(m, \mathsf{st}) \leftarrow \mathcal{A}_1(1^k)$
    $r_g, r'_g, r_c, r_t \leftarrow \{0,1\}^*$
    $ck \leftarrow \mathsf{CKG}(1^k; r_g)$
    $\mathsf{tag}^* \leftarrow \mathsf{Com}(ck, m; r_c)$
    $\widehat{r}_c \leftarrow \mathsf{rSamp}_{\mathcal{C}}(r_g, r_c, m)$
    $(pk, sk) \leftarrow \mathsf{TKG}(1^k; r'_g)$
    $c^* \leftarrow \mathsf{TEnc}(ck, \mathsf{tag}^*, m; r_e)$
    $\widehat{sk}_{\mathsf{tag}^*} \leftarrow \mathsf{Punc}(sk, \mathsf{tag}^*)$
    $\widehat{r}_t \leftarrow \mathsf{rSamp}_{\mathcal{T}}(r'_g, r_t, \mathsf{tag}^*, m)$
    $b' \leftarrow \mathcal{A}_2(\mathsf{st}, ck, \mathsf{tag}^*, pk, c^*, \widehat{sk}_{\mathsf{tag}^*}, \widehat{r}_c, \widehat{r}_t)$
    Return $b'$.
```

```
$\mathsf{Expt}^{\mathtt{TS\text{-}Sim}}_{[\mathcal{C},\mathcal{T}],\mathcal{A}}(k)$ :
    $(m, \mathsf{st}) \leftarrow \mathcal{A}_1(1^k)$
    $\widehat{r}_c, \widehat{r}_t \leftarrow \{0,1\}^*$
    $(ck, \mathsf{tag}^*) \leftarrow \mathsf{oSamp}_{\mathcal{C}}(1^k; \widehat{r}_c)$
    $(pk, c^*, \widehat{sk}_{\mathsf{tag}^*}) \leftarrow \mathsf{oSamp}_{\mathcal{T}}(\mathsf{tag}^*; \widehat{r}_t)$
    $b' \leftarrow \mathcal{A}_2(\mathsf{st}, ck, \mathsf{tag}^*, pk, c^*, \widehat{sk}_{\mathsf{tag}^*}, \widehat{r}_c, \widehat{r}_t)$
    Return $b'$.
```

Fig. 2. Security experiments for defining the trapdoor simulatability of the concatenation of a "transcript" of a trapdoor simulatable commitment scheme and that of a TSPTBE scheme.

Lemma 2. *Assume that the commitment scheme $\mathcal{C}$ and the PTBE scheme $\mathcal{T}$ are both trapdoor simulatable. Then, for all PPTAs $\mathcal{A}$, $\mathsf{Adv}^{\mathtt{TS}}_{[\mathcal{C},\mathcal{T}],\mathcal{A}}(k) := |\Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Real}}_{[\mathcal{C},\mathcal{T}],\mathcal{A}}(k) = 1] - \Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Sim}}_{[\mathcal{C},\mathcal{T}],\mathcal{A}}(k) = 1]|$ is negligible.*

4 Proposed KEMs

In this section, we show our main results: two KEMs that show the "trade-off" between the strength of (standard model) plaintext awareness and the simulatability property with those of the construction by Dachman-Soled [19].

In Section 4.1, we show our first construction, which is `CCA` secure based on a KEM satisfying `CPA` security and $\mathtt{sPA1}_2$ security, and a TSPKE scheme. In Section 4.2, we show our second construction which is `CCA` secure based on a KEM satisfying `1-CCA` security and $\mathtt{sPA1}_1$ security, and a TSPKE scheme.

4.1 First Construction

Let $\Gamma_{\mathtt{in}} = (\mathsf{KKG}_{\mathtt{in}}, \mathsf{Encap}_{\mathtt{in}}, \mathsf{Decap}_{\mathtt{in}})$ be a KEM whose ciphertext length is $n = n(k)$ and whose session-key space is $\{0,1\}^{3k}$ for k-bit security. [5] Let $\mathcal{T} = (\mathsf{TKG}, \mathsf{TEnc}, \mathsf{TDec}, \mathsf{Punc}, \widehat{\mathsf{TDec}})$ be a PTBE scheme and $\mathcal{C} = (\mathsf{CKG}, \mathsf{Com})$ be a commitment scheme. We require the plaintext space of TEnc and the message space of Com to be $\{0,1\}^{2n}$, and the randomness space of TEnc and that of Com to be $\{0,1\}^k$ for k-bit security. [6] Then, our first proposed KEM $\Gamma = (\mathsf{KKG}, \mathsf{Encap}, \mathsf{Decap})$ is constructed as in Fig. 3.

Alternative Decapsulation Algorithm. Similarly to the constructions in [38–40], to show the `CCA` security of the proposed KEM Γ, it is useful to consider the following alternative decapsulation algorithm $\mathsf{AltDecap}$. For a k-bit string $\mathsf{tag}^* \in \{0,1\}^k$ and a key pair (PK, SK) output by $\mathsf{KKG}(1^k)$, where $PK = (pk_{\mathtt{in0}}, pk_{\mathtt{in1}}, pk, ck)$ and $SK = (sk_{\mathtt{in0}}, sk_{\mathtt{in1}}, sk, PK)$, we define an "alternative" secret key $\widehat{SK}_{\mathsf{tag}^*}$ associated with $\mathsf{tag}^* \in \{0,1\}^k$ by $\widehat{SK}_{\mathsf{tag}^*} = (sk_{\mathtt{in0}}, sk_{\mathtt{in1}}, \mathsf{tag}^*, \widehat{sk}_{\mathsf{tag}^*}, PK)$, where

[5] Note that the session-key space of a KEM can be adjusted "for free" by applying a pseudorandom generator to a session-key. Such a construction preserves `CPA` and $\mathtt{sPA1}_\ell$ security.

[6] The requirements of the randomness space of TEnc and Com are without loss of generality, because we can adjust them using a pseudorandom generator. (The trapdoor simulatability property is preserved even if we use a pseudorandom generator.)

```
$\mathsf{KKG}(1^k)$ :
    $(pk_{\mathtt{in0}}, sk_{\mathtt{in0}}) \leftarrow \mathsf{KKG}_{\mathtt{in}}(1^k)$
    $(pk_{\mathtt{in1}}, sk_{\mathtt{in1}}) \leftarrow \mathsf{KKG}_{\mathtt{in}}(1^k)$
    $(pk, sk) \leftarrow \mathsf{TKG}(1^k)$
    $ck \leftarrow \mathsf{CKG}(1^k)$
    $PK \leftarrow (pk_{\mathtt{in0}}, pk_{\mathtt{in1}}, pk, ck)$
    $SK \leftarrow (sk_{\mathtt{in0}}, sk_{\mathtt{in1}}, sk, PK)$
    Return $(PK, SK)$.
$\mathsf{Encap}(PK)$ :
    $(pk_{\mathtt{in0}}, pk_{\mathtt{in1}}, pk, ck) \leftarrow PK$
    $(c_{\mathtt{in0}}, \alpha_0) \leftarrow \mathsf{Encap}_{\mathtt{in}}(pk_{\mathtt{in0}})$
    $(c_{\mathtt{in1}}, \alpha_1) \leftarrow \mathsf{Encap}_{\mathtt{in}}(pk_{\mathtt{in1}})$
    $\alpha \leftarrow \alpha_0 \oplus \alpha_1$
    Parse $\alpha$ as $(r_c, r_t, K) \in (\{0,1\}^k)^3$
    $\mathsf{tag} \leftarrow \mathsf{Com}(ck, (c_{\mathtt{in0}} \| c_{\mathtt{in1}}); r_c)$
    $c \leftarrow \mathsf{TEnc}(pk, \mathsf{tag}, (c_{\mathtt{in0}} \| c_{\mathtt{in1}}); r_t)$
    $C \leftarrow (\mathsf{tag}, c)$.
    Return $(C, K)$.
```

```
$\mathsf{Decap}(SK, C)$ :
    $(sk_{\mathtt{in0}}, sk_{\mathtt{in1}}, sk, PK) \leftarrow SK$
    $(pk_{\mathtt{in0}}, pk_{\mathtt{in1}}, pk, ck) \leftarrow PK$
    $(\mathsf{tag}, c) \leftarrow C$
    $(c_{\mathtt{in0}} \| c_{\mathtt{in1}}) \leftarrow \mathsf{TDec}(sk, \mathsf{tag}, c)$
    If $\mathsf{TDec}$ has returned $\perp$ then return $\perp$.
    $\alpha_0 \leftarrow \mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in0}}, c_{\mathtt{in0}})$
    $\alpha_1 \leftarrow \mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in1}}, c_{\mathtt{in1}})$
    If $\alpha_0 = \perp$ or $\alpha_1 = \perp$ then return $\perp$.
    $\alpha \leftarrow \alpha_0 \oplus \alpha_1$
    Parse $\alpha$ as $(r_c, r_t, K) \in (\{0,1\}^k)^3$
    If $\mathsf{Com}(ck, (c_{\mathtt{in0}} \| c_{\mathtt{in1}}); r_c) = \mathsf{tag}$
        and $\mathsf{TEnc}(pk, \mathsf{tag}, (c_{\mathtt{in0}} \| c_{\mathtt{in1}}); r_t) = c$
            then return $K$ else return $\perp$.
```

Fig. 3. The first proposed construction: the KEM Γ based on a KEM $\Gamma_{\mathtt{in}}$, a commitment scheme $\mathcal{C}$, and a PTBE scheme $\mathcal{T}$.

$\widehat{sk}_{\mathsf{tag}^*} = \mathsf{Punc}(sk, \mathsf{tag}^*)$. $\mathsf{AltDecap}$ takes an "alternative" secret key $\widehat{SK}_{\mathsf{tag}^*}$ defined as above and a ciphertext $C = (\mathsf{tag}, c)$ as input, and runs as follows:

$\mathsf{AltDecap}(\widehat{SK}_{\mathsf{tag}^*}, C)$: First check if $\mathsf{tag}^* = \mathsf{tag}$, and return $\perp$ if this is the case. Otherwise, run in exactly the same way as $\mathsf{Decap}(SK, C)$, except that "$(c_{\mathsf{in0}} \| c_{\mathsf{in1}}) \leftarrow \widehat{\mathsf{TDec}}(\widehat{sk}_{\mathsf{tag}^*}, \mathsf{tag}, c)$" is executed in the fourth step, instead of "$(c_{\mathsf{in0}} \| c_{\mathsf{in1}}) \leftarrow \mathsf{TDec}(sk, \mathsf{tag}, c)$."

Regarding $\mathsf{AltDecap}$, the following lemma is easy to see due to the correctness of the underlying PTBE scheme $\mathcal{T}$ and the validity check of c by re-encryption performed at the last step.

Lemma 3. *Let $\mathsf{tag}^* \in \{0,1\}^k$ be a string and let (PK, SK) be a key pair output by $\mathsf{KKG}(1^k)$. Furthermore, let $\widehat{SK}_{\mathsf{tag}^*}$ be an alternative secret key as defined above. Then, for any ciphertext $C = (\mathsf{tag}, c)$ (which could be outside the range of $\mathsf{Encap}(PK)$) satisfying $\mathsf{tag} \neq \mathsf{tag}^*$, it holds that $\mathsf{Decap}(SK, C) = \mathsf{AltDecap}(\widehat{SK}_{\mathsf{tag}^*}, C)$.*

Proof of Lemma 3. Let $\mathsf{tag}^* \in \{0,1\}^k$, PK, $SK = (sk_{\mathsf{in0}}, sk_{\mathsf{in1}}, sk, PK)$, and $\widehat{SK}_{\mathsf{tag}^*} = (sk_{\mathsf{in0}}, sk_{\mathsf{in1}}, \mathsf{tag}^*, \widehat{sk}_{\mathsf{tag}^*}, PK)$ be as stated in the lemma. Fix arbitrarily a ciphertext $C = (\mathsf{tag}, c)$ (which could be outside the range of $\mathsf{Encap}(PK)$) satisfying $\mathsf{tag} \neq \mathsf{tag}^*$. For notational convenience, let $(c_{\mathsf{in0}} \| c_{\mathsf{in1}}) = \mathsf{TDec}(sk, \mathsf{tag}, c)$ and $(c'_{\mathsf{in0}} \| c'_{\mathsf{in1}}) = \mathsf{TDec}(\widehat{sk}_{\mathsf{tag}^*}, \mathsf{tag}, c)$. Consider the following two cases that cover all possibilities:

Case $(c_{\mathsf{in0}} \| c_{\mathsf{in1}}) = (c'_{\mathsf{in0}} \| c'_{\mathsf{in1}})$: In this case, both Decap and $\mathsf{AltDecap}$ proceed identically after their fourth step, and thus the outputs from both algorithms agree.

Case $(c_{\mathsf{in0}} \| c_{\mathsf{in1}}) \neq (c'_{\mathsf{in0}} \| c'_{\mathsf{in1}})$: In this case, both Decap and $\mathsf{AltDecap}$ return $\perp$. To see this, recall that the last step of Decap and that of $\mathsf{AltDecap}$ both check whether the second component c of $C = (\mathsf{tag}, c)$ is an output of $\mathsf{TEnc}(pk, \mathsf{tag}, \cdot; \cdot)$. However, the correctness of the PTBE scheme $\mathcal{T}$ implies that for all tags $\mathsf{tag} \neq \mathsf{tag}^*$ and all ciphertexts c produced from $\mathsf{TEnc}(pk, \mathsf{tag}, \cdot; \cdot)$, it holds that $\mathsf{TDec}(sk, \mathsf{tag}, c) = \mathsf{TDec}(\widehat{sk}_{\mathsf{tag}^*}, \mathsf{tag}, c)$. Thus, that $(c_{\mathsf{in0}} \| c_{\mathsf{in1}}) \neq (c'_{\mathsf{in0}} \| c'_{\mathsf{in1}})$ occurs implies that c is not in the range of $\mathsf{TEnc}(pk, \mathsf{tag}, \cdot; \cdot)$. That is, there exists no randomness r such that $\mathsf{TEnc}(pk, \mathsf{tag}, (c_{\mathsf{in0}} \| c_{\mathsf{in1}}); r) = c$ or $\mathsf{TEnc}(pk, \mathsf{tag}, (c'_{\mathsf{in0}} \| c'_{\mathsf{in1}}); r) = c$, and thus both Decap

and $\mathsf{AltDecap}$ return $\perp$ at their last step at the latest. Actually, Decap could return $\perp$ earlier if $(c_{\mathtt{in0}} \| c_{\mathtt{in1}}) = \perp$, $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in0}}, c_{\mathtt{in0}}) = \perp$, or $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in1}}, c_{\mathtt{in1}}) = \perp$, and the situation is similar for $\mathsf{AltDecap}$. However, in any case the output of these algorithms is $\perp$.

We have seen that $\mathsf{Decap}(SK, C) = \mathsf{AltDecap}(\widehat{SK}_{\mathsf{tag}^*}, C)$ holds for all ciphertexts $C = (\mathsf{tag}, c)$ satisfying $\mathsf{tag} \neq \mathsf{tag}^*$, which proves the lemma. □ (**Lemma 3**)

$\mathtt{CCA}$ *Security.* The security of Γ is guaranteed by the following theorem.

Theorem 1. *Assume that the KEM $\Gamma_{\mathtt{in}}$ is $\mathtt{CPA}$ secure and $\mathtt{sPA1}_2$ secure, the commitment scheme $\mathcal{C}$ is target-binding and trapdoor simulatable, and the PTBE scheme $\mathcal{T}$ is trapdoor simulatable. Then, the KEM Γ constructed as in Fig. 3 is $\mathtt{CCA}$ secure.*

Note that as mentioned in Section 2.3, a commitment scheme with trapdoor simulatability and target-binding can be constructed from any TSPKE scheme, and thus the above theorem shows that we can indeed construct a $\mathtt{CCA}$ secure KEM (and thus $\mathtt{CCA}$ secure PKE) from the combination of a KEM satisfying $\mathtt{CPA}$ and $\mathtt{sPA1}_2$ security and a TSPKE scheme.

We have provided ideas for the security proof in Section 1.3, and thus we directly proceed to the proof.

Proof of Theorem 1. Let $\mathcal{A}$ be any PPTA adversary that attacks the $\mathtt{CCA}$ security of the KEM Γ. Our security proof is via the sequence of games argument. To describe the games, we will need an extractor $\mathcal{E}$ corresponding to some "ciphertext creator" $\mathcal{A}'$ that is guaranteed to exist by the $\mathtt{sPA1}_2$ security of $\Gamma_{\mathtt{in}}$. Specifically, consider the following $\mathcal{A}'$ (that internally runs $\mathcal{A}$) that runs in the experiment $\mathsf{Expt}^{\mathtt{sPA1}}_{\Gamma_{\mathtt{in}},\mathcal{A}',\mathcal{E},2}(k)$, with a corresponding extractor $\mathcal{E}$:

$\mathcal{A}'^{\mathcal{E}(\mathsf{st}_{\mathcal{E}},\cdot)}(pk_1, pk_2; r_{\mathcal{A}'} = (r_{\mathcal{A}}, \widehat{r}_c, \widehat{r}_t, K^*))$**:** $\mathcal{A}'$ firstly sets $pk_{\mathtt{in0}} \leftarrow pk_1$ and $pk_{\mathtt{in1}} \leftarrow pk_2$ (which implicitly sets $sk_{\mathtt{in0}} \leftarrow sk_1$ and $sk_{\mathtt{in1}} \leftarrow sk_2$, where sk_1 (resp. sk_2) is the secret key corresponding to pk_1 (resp. pk_2)), and runs $(ck, \mathsf{tag}^*) \leftarrow \mathsf{oSamp}_{\mathcal{C}}(1^k; \widehat{r}_c)$ and $(pk, c^*, \widehat{sk}_{\mathsf{tag}^*}) \leftarrow \mathsf{oSamp}_{\mathcal{T}}(\mathsf{tag}^*; \widehat{r}_t)$. Then $\mathcal{A}'$ sets $PK \leftarrow (pk_{\mathtt{in0}}, pk_{\mathtt{in1}}, pk, ck)$ and $C^* \leftarrow (\mathsf{tag}^*, c^*)$, and then runs $\mathcal{A}(PK, C^*, K^*; r_{\mathcal{A}})$.

When $\mathcal{A}$ submits a decapsulation query C, $\mathcal{A}'$ responds to it as if it runs $\mathsf{AltDecap}(\widehat{SK}_{\mathsf{tag}^*}, C)$, where the oracle calls (to the extractor $\mathcal{E}$) of the form $(1, c_{\mathtt{in0}})$ and $(2, c_{\mathtt{in1}})$ are used as substitutes for $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in0}}, c_{\mathtt{in0}})$ and $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in1}}, c_{\mathtt{in1}})$, respectively. More precisely, $\mathcal{A}'$ answers $\mathcal{A}$'s decapsulation query $C = (\mathsf{tag}, c)$ as follows:

1. If $\mathsf{tag} = \mathsf{tag}^*$, then return $\perp$ to $\mathcal{A}$.
2. Run $(c_{\mathtt{in0}} \| c_{\mathtt{in1}}) \leftarrow \widehat{\mathsf{TDec}}(\widehat{sk}_{\mathsf{tag}^*}, \mathsf{tag}, c)$, and return $\perp$ to $\mathcal{A}$ if $\widehat{\mathsf{TDec}}$ has returned $\perp$.
3. Submit queries $(1, c_{\mathtt{in0}})$ and $(2, c_{\mathtt{in1}})$ to the extractor $\mathcal{E}(\mathsf{st}_{\mathcal{E}}, \cdot)$ and receive the answers α_0 and α_1, respectively. (Here, the answers α_0 and α_1 are expected to be $\alpha_0 = \mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in0}}, c_{\mathtt{in0}})$ and $\alpha_1 = \mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in1}}, c_{\mathtt{in1}})$, respectively, and the extractor $\mathcal{E}$ may update its state upon each call.)
4. If $\alpha_0 = \perp$ or $\alpha_1 = \perp$, then return $\perp$ to $\mathcal{A}$.
5. Let $\alpha \leftarrow \alpha_0 \oplus \alpha_1$ and parse α as $(r_c, r_t, K) \in (\{0,1\}^k)^3$.
6. If $\mathsf{Com}(ck, (c_{\mathtt{in0}} \| c_{\mathtt{in1}}); r_c) = \mathsf{tag}$ and $\mathsf{TEnc}(pk, (c_{\mathtt{in0}} \| c_{\mathtt{in1}}); r_t) = c$, then return K, otherwise return $\perp$, to $\mathcal{A}$.

When $\mathcal{A}$ terminates, $\mathcal{A}'$ also terminates.

The above completes the description of the algorithm $\mathcal{A}'$. The randomness $r_{\mathcal{A}'}$ consumed by $\mathcal{A}'$ is of the form $(r_{\mathcal{A}}, \widehat{r}_c, \widehat{r}_t, K^*)$, where $r_{\mathcal{A}}$, $\widehat{r}_c$, and $\widehat{r}_t$ are the randomness used by $\mathcal{A}$, $\mathsf{oSamp}_{\mathcal{C}}$, and $\mathsf{oSamp}_{\mathcal{T}}$, respectively, and K^* is a k-bit string. The corresponding extractor $\mathcal{E}$ thus receives (pk_1, pk_2) and $r_{\mathcal{A}'}$ as its initial state $\mathsf{st}_{\mathcal{E}}$. Note that since $\Gamma_{\mathtt{in}}$ is assumed to be $\mathtt{sPA1}_2$ secure and $\mathcal{A}'$ is a PPTA, $\mathsf{Adv}^{\mathtt{sPA1}}_{\Gamma_{\mathtt{in}},\mathcal{A}',\mathcal{E},2}(k)$ is negligible for this extractor $\mathcal{E}$, which will be used later in the proof. (Looking ahead, we will design the sequence of games so that $\mathcal{A}$'s view in the case $\mathcal{A}$ is internally run by $\mathcal{A}'$ and $\mathcal{A}'$ is run in $\mathsf{Expt}^{\mathtt{sPA1}}_{\Gamma_{\mathtt{in}},\mathcal{A}',\mathcal{E},2}(k)$, is identical to $\mathcal{A}$'s view in Game 6.)

For convenience, we refer to the procedure of using the extractor $\mathcal{E}$ as substitutes for $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in0}}, \cdot)$ and $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in1}}, \cdot)$, as $\mathsf{AltDecap}'_{\mathcal{E}}$. Here, $\mathsf{AltDecap}'_{\mathcal{E}}$ is a stateful procedure that initially takes tag^*, $\widehat{sk}_{\mathsf{tag}^*}$, and an initial state $\mathsf{st}_{\mathcal{E}}$ of $\mathcal{E}$ (i.e. $\mathsf{st}_{\mathcal{E}} = ((pk_{\mathtt{in0}}, pk_{\mathtt{in1}}), r_{\mathcal{A}'})$) as input, and expects to receive a ciphertext $C = (\mathsf{tag}, c)$ as an input. If it receives a ciphertext $C = (\mathsf{tag}, c)$, it calculates the decapsulation result K (or $\bot$) as $\mathcal{A}'$ does for $\mathcal{A}$, using $\widehat{sk}_{\mathsf{tag}^*}$ and the extractor $\mathcal{E}$, where $\mathcal{E}$'s internal state could be updated upon each execution.

Now, using the adversary $\mathcal{A}$ and the extractor $\mathcal{E}$, consider the following sequence of games: (Here, the values with asterisk (*) represent those related to the challenge ciphertext for $\mathcal{A}$.)

Game 1: This is the experiment $\mathsf{Expt}^{\mathtt{CCA}}_{\Gamma,\mathcal{A}}(k)$ itself.

Game 2: Same as Game 1, except that all decapsulation queries $C = (\mathsf{tag}, c)$ satisfying $\mathsf{tag} = \mathsf{tag}^*$ are answered with $\bot$.

Game 3: Same as Game 2, except that all decapsulation queries C are answered with $\mathsf{AltDecap}(\widehat{SK}_{\mathsf{tag}^*}, C)$, where $\widehat{SK}_{\mathsf{tag}^*}$ is the alternative secret key corresponding to (PK, SK) and tag^*. Furthermore, we pick a random bit $\gamma \in \{0,1\}$ uniformly at random just before executing $\mathcal{A}$, which will be used to define the events in this game and the subsequent games. (γ does not appear in $\mathcal{A}$'s view in this and all subsequent games, and thus does not affect its behavior at all.)

Game 4: In this game, we use $\mathsf{AltDecap}'_{\mathcal{E}}$ (defined as above) as $\mathcal{A}$'s decapsulation oracle, where the initial state of $\mathcal{E}$ (used internally by $\mathsf{AltDecap}'_{\mathcal{E}}$) is prepared using the "inverting algorithms" $\mathsf{rSamp}_{\mathcal{C}}$ of $\mathcal{C}$ and $\mathsf{rSamp}_{\mathcal{T}}$ of $\mathcal{T}$. Moreover, we also change the ordering of the steps so that they do not affect $\mathcal{A}$'s view.

More precisely, this game is defined as follows:

```
Game 4:
  $(pk_{\mathtt{in0}}, sk_{\mathtt{in0}}) \leftarrow \mathsf{KKG}_{\mathtt{in}}(1^k)$;
  $(pk_{\mathtt{in1}}, sk_{\mathtt{in1}}) \leftarrow \mathsf{KKG}_{\mathtt{in}}(1^k)$;
  $(c^*_{\mathtt{in0}}, \alpha^*_0) \leftarrow \mathsf{Encap}_{\mathtt{in}}(pk_{\mathtt{in0}})$;
  $(c^*_{\mathtt{in1}}, \alpha^*_1) \leftarrow \mathsf{Encap}_{\mathtt{in}}(pk_{\mathtt{in1}})$;
  $\alpha^* \leftarrow (\alpha^*_0 \oplus \alpha^*_1)$;
  Parse $\alpha^*$ as $(r^*_c, r^*_t, K^*_1) \in (\{0,1\}^k)^3$;
  (Continue to the center column ↗)
  $r_g \leftarrow \{0,1\}^*$;
  $ck \leftarrow \mathsf{CKG}(1^k; r_g)$;
  $\mathsf{tag}^* \leftarrow \mathsf{Com}(ck, (c^*_{\mathtt{in0}} \| c^*_{\mathtt{in1}}); r^*_c)$;
  $\widehat{r}_c \leftarrow \mathsf{rSamp}_{\mathcal{C}}(r_g, r^*_c, (c^*_{\mathtt{in0}} \| c^*_{\mathtt{in1}}))$;
  $r'_g \leftarrow \{0,1\}^*$;
  $(pk, sk) \leftarrow \mathsf{TKG}(1^k; r'_g)$;
  $\widehat{sk}_{\mathsf{tag}^*} \leftarrow \mathsf{Punc}(sk, \mathsf{tag}^*)$;
  $c^* \leftarrow \mathsf{TEnc}(pk, \mathsf{tag}^*, (c^*_{\mathtt{in0}} \| c^*_{\mathtt{in1}}); r^*_t)$;
  $\widehat{r}_t \leftarrow \mathsf{rSamp}_{\mathcal{T}}(r'_g, r^*_t, \mathsf{tag}^*, (c^*_{\mathtt{in0}} \| c^*_{\mathtt{in1}}))$;
  (Continue to the right column ↗)
  $PK \leftarrow (pk_{\mathtt{in0}}, pk_{\mathtt{in1}}, pk, ck)$;
  $C^* \leftarrow (\mathsf{tag}^*, c^*)$;
  $K^*_0 \leftarrow \{0,1\}^k$;
  $b \leftarrow \{0,1\}$;
  $r_{\mathcal{A}} \leftarrow \{0,1\}^*$;
  $r_{\mathcal{A}'} \leftarrow (r_{\mathcal{A}}, \widehat{r}_c, \widehat{r}_t, K^*_1)$;
  $\mathsf{st}_{\mathcal{E}} \leftarrow ((pk_{\mathtt{in0}}, pk_{\mathtt{in1}}), r_{\mathcal{A}'})$;
  $\gamma \leftarrow \{0,1\}$;
  $b' \leftarrow \mathcal{A}^{\mathcal{O}}(PK, C^*, K^*_b; r_{\mathcal{A}})$
```

where the decapsulation oracle $\mathcal{O}$ that $\mathcal{A}$ has access in Game 4 is $\mathsf{AltDecap}'_{\mathcal{E}}$ (which initially receives $\mathsf{tag}^*, \widehat{sk}_{\mathsf{tag}^*}, \mathsf{st}_{\mathcal{E}} = (pk_{\mathtt{in0}}, pk_{\mathtt{in1}}, r_{\mathcal{A}'})$ as input). Note that the extractor $\mathcal{E}$ used internally by $\mathsf{AltDecap}'_{\mathcal{E}}$ may update its state $\mathsf{st}_{\mathcal{E}}$ upon each execution.

Game 5: Same as Game 4, except that $r^*_c, r^*_t, K^*_1 \in \{0,1\}^k$ are picked uniformly at random, independently of $\alpha^* = \alpha^*_0 \oplus \alpha^*_1$. That is, the steps "$\alpha^* \leftarrow \alpha^*_0 \oplus \alpha^*_1$; Parse α^* as $(r^*_c, r^*_t, K^*_1) \in (\{0,1\}^k)^3$" in Game 4 are replaced with the step "$r^*_c, r^*_t, K^*_1 \leftarrow \{0,1\}^k$," and we do not use α^* anymore.

Game 6: Same as Game 5, except that the key/commitment pair (ck, tag^*) and the key/ciphertext pair (pk, c^*) and a punctured secret key $\widehat{sk}_{\mathsf{tag}^*}$ are sampled obliviously, and correspondingly the randomness $\widehat{r}_c$ and $\widehat{r}_t$ used for oblivious sampling are used in $r_{\mathcal{A}'}$.

More precisely, the steps “$r_g, r_c^* \leftarrow \{0,1\}^*$; $ck \leftarrow \mathsf{CKG}(1^k; r_g)$; $\mathsf{tag}^* \leftarrow \mathsf{Com}(ck, (c^*_{\mathtt{in0}} \| c^*_{\mathtt{in1}}); r^*_c)$; $\widehat{r}_c \leftarrow \mathsf{rSamp}_{\mathcal{C}}(r_g, r^*_c, (c^*_{\mathtt{in0}} \| c^*_{\mathtt{in1}}))$” in Game 5 are replaced with the steps “$\widehat{r}_c \leftarrow \{0,1\}^*$; $(ck, \mathsf{tag}^*) \leftarrow \mathsf{oSamp}_{\mathcal{C}}(1^k; \widehat{r}_c)$”.

Furthermore, the steps “$r'_g, r^*_t \leftarrow \{0,1\}^k$; $(pk, sk) \leftarrow \mathsf{TKG}(1^k; r'_g)$; $c^* \leftarrow \mathsf{TEnc}(pk, \mathsf{tag}^*, (c^*_{\mathtt{in0}} \| c^*_{\mathtt{in1}}); r^*_t)$; $\widehat{r}_t \leftarrow \mathsf{rSamp}_{\mathcal{T}}(r'_g, r^*_t, \mathsf{tag}^*, (c^*_{\mathtt{in0}} \| c^*_{\mathtt{in1}}))$” in Game 5 are replaced with the steps “$\widehat{r}_t \leftarrow \{0,1\}^*$; $(pk, \widehat{sk}_{\mathsf{tag}^*}, c^*) \leftarrow \mathsf{oSamp}_{\mathcal{T}}(\mathsf{tag}^*; \widehat{r}_t)$”.

The above completes the description of the games.

For $i \in [5]$, let Succ_i denote the event that $\mathcal{A}$ succeeds in guessing the challenge bit (i.e. $b' = b$ occurs) in Game i. Furthermore, for $i \in \{3, \dots, 6\}$, we define the following *bad* events in Game i:

Bad_i**:** $\mathcal{A}$ submits a decapsulation query $C = (\mathsf{tag}, c)$ satisfying the following conditions simultaneously: (1) $\mathsf{tag} \neq \mathsf{tag}^*$, (2) $\widehat{\mathsf{TDec}}(\widehat{sk}_{\mathsf{tag}^*}, \mathsf{tag}, c) = (c_{\mathtt{in0}} \| c_{\mathtt{in1}}) \neq \bot$, and (3) $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in0}}, c_{\mathtt{in0}}) \neq \mathcal{E}(\mathsf{st}_{\mathcal{E}}, (1, c_{\mathtt{in0}}))$ *or* $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in1}}, c_{\mathtt{in1}}) \neq \mathcal{E}(\mathsf{st}_{\mathcal{E}}, (2, c_{\mathtt{in1}}))$.

$\mathsf{Bad}_i^{(\sigma)}$**:** (where $\sigma \in \{0,1\}$) $\mathcal{A}$ submits a decapsulation query $C = (\mathsf{tag}, c)$ that satisfies the same conditions as Bad_i, except that the condition (3) is replaced with the condition: $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in}\sigma}, c_{\mathtt{in}\sigma}) \neq \mathcal{E}(\mathsf{st}_{\mathcal{E}}, (\sigma + 1, c_{\mathtt{in}\sigma}))$.

Bad_i^***:** $\mathcal{A}$ submits a decapsulation query $C = (\mathsf{tag}, c)$ that satisfies the same conditions as Bad_i, except that the condition (3) is replaced with the condition: $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in}\gamma}, c_{\mathtt{in}\gamma}) \neq \mathcal{E}(\mathsf{st}_{\mathcal{E}}, (\gamma + 1, c_{\mathtt{in}\gamma}))$ (where γ is the random bit chosen just before executing $\mathcal{A}$).

Note that for all $i \in \{3, \dots, 6\}$, the events $\mathsf{Bad}_i^{(0)}$, $\mathsf{Bad}_i^{(1)}$, and Bad_i^* all imply the event Bad_i, and thus we have $\Pr[\mathsf{Bad}_i^{(0)}], \Pr[\mathsf{Bad}_i^{(1)}], \Pr[\mathsf{Bad}_i^*] \leq \Pr[\mathsf{Bad}_i]$.

By the definitions of the games and events, we have

$$\begin{aligned}
\mathsf{Adv}^{\mathtt{CCA}}_{\Gamma,\mathcal{A}}(k) &= 2 \cdot \left| \Pr[\mathsf{Succ}_1] - \frac{1}{2} \right| \\
&\leq 2 \cdot \Big(\sum_{i \in [4]} \Big| \Pr[\mathsf{Succ}_i] - \Pr[\mathsf{Succ}_{i+1}] \Big| + \Big| \Pr[\mathsf{Succ}_5] - \frac{1}{2} \Big| \Big). \qquad (1)
\end{aligned}$$

In the following, we will upperbound each term that appears in the right hand side of the above inequality.

Claim 1 *There exists a PPTA $\mathcal{B}_{\mathsf{b}}$ such that $\mathsf{Adv}^{\mathtt{TBind}}_{\mathcal{C},\mathcal{B}_{\mathsf{b}}}(k) \geq |\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_2]|$.*

Proof of Claim 1. For $i \in \{1, 2\}$, let NoBind_i be the event that in Game i, $\mathcal{A}$ submits at least one decapsulation query $C = (\mathsf{tag}, c)$ satisfying $\mathsf{tag} = \mathsf{tag}^*$ and $\mathsf{Decap}(SK, C) \neq \bot$. Recall that $\mathcal{A}$'s query C must satisfy $C \neq C^* = (\mathsf{tag}^*, c^*)$, and thus $\mathsf{tag} = \mathsf{tag}^*$ implies $c \neq c^*$. The difference between Game 1 and Game 2 is how $\mathcal{A}$'s decapsulation query $C = (\mathsf{tag}, c)$ satisfying $\mathsf{tag} = \mathsf{tag}^*$ is answered. Hence, these games proceed identically unless NoBind_1 or NoBind_2 occurs in the corresponding games, and thus we have

$$\Big| \Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_2] \Big| \leq \Pr[\mathsf{NoBind}_1] = \Pr[\mathsf{NoBind}_2]. \qquad (2)$$

Thus, it is sufficient to upperbound $\Pr[\mathsf{NoBind}_2]$.

Observe that for a decapsulation query $C = (\mathsf{tag}^*, c)$ satisfying the condition of NoBind_2, it is guaranteed that $\mathsf{TDec}(sk, \mathsf{tag}, c) = (c_{\mathtt{in0}} \| c_{\mathtt{in1}}) \neq (c^*_{\mathtt{in0}} \| c^*_{\mathtt{in1}})$. Indeed, if $\mathsf{TDec}(sk, \mathsf{tag}, c) = (c^*_{\mathtt{in0}} \| c^*_{\mathtt{in1}})$ and $\mathsf{Decap}(SK, C) \neq \bot$, then by the validity check of c in Decap, we have $c^* = c$, which is because c must satisfy $\mathsf{TEnc}(pk, \mathsf{tag}^*, (c^*_{\mathtt{in0}} \| c^*_{\mathtt{in1}}); r^*_t) = c$ where r^*_t is the $(k+1)$-to-$2k$-th bits of $\alpha^* = (\alpha^*_0 \oplus \alpha^*_1) = (\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in0}}, c^*_{\mathtt{in0}}) \oplus \mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in1}}, c^*_{\mathtt{in1}}))$. However, $\mathsf{TEnc}(pk, \mathsf{tag}^*, (c^*_{\mathtt{in0}} \| c^*_{\mathtt{in1}}); r^*_t) = c^*$ also holds due to how c^* is generated, and thus contradicting the condition $c \neq c^*$ implied by NoBind_2.

We use the above fact to show how to construct a PPTA adversary $\mathcal{B}_{\mathsf{b}}$ that attacks the target-binding property of the commitment scheme $\mathcal{C}$ with advantage $\mathsf{Adv}^{\mathtt{TBind}}_{\mathcal{C},\mathcal{B}_{\mathsf{b}}}(k) = \Pr[\mathsf{NoBind}_2]$. The description of $\mathcal{B}_{\mathsf{b}} = (\mathcal{B}_{\mathsf{b1}}, \mathcal{B}_{\mathsf{b2}})$ is as follows:

$\mathcal{B}_{\mathsf{b1}}(1^k)$**:** $\mathcal{B}_{\mathsf{b1}}$ first runs $(pk_{\mathtt{in0}}, sk_{\mathtt{in0}}) \leftarrow \mathsf{KKG}_{\mathtt{in}}(1^k)$, $(pk_{\mathtt{in1}}, sk_{\mathtt{in1}}) \leftarrow \mathsf{KKG}_{\mathtt{in}}(1^k)$, $(c^*_{\mathtt{in0}}, \alpha^*_0) \leftarrow \mathsf{Encap}_{\mathtt{in}}(pk_{\mathtt{in0}})$, and $(c^*_{\mathtt{in1}}, \alpha^*_1) \leftarrow \mathsf{Encap}_{\mathtt{in}}(pk_{\mathtt{in1}})$. $\mathcal{B}_{\mathsf{b1}}$ then sets $\alpha^* \leftarrow (\alpha^*_0 \oplus \alpha^*_1)$, and parses α^* as $(r^*_c, r^*_t, \alpha^*) \in (\{0,1\}^k)^3$. Finally, $\mathcal{B}_{\mathsf{b1}}$ sets $M \leftarrow (c^*_{\mathtt{in0}} \| c^*_{\mathtt{in1}})$, $R \leftarrow r^*_c$, and $\mathsf{st}_{\mathcal{B}} \leftarrow$ ($\mathcal{B}_{\mathsf{b1}}$'s entire view), and terminates with output $(M, R, \mathsf{st}_{\mathcal{B}})$.

$\mathcal{B}_{\mathsf{b2}}(\mathsf{st}_{\mathcal{B}}, ck)$**:** $\mathcal{B}_{\mathsf{b2}}$ first runs $(pk, sk) \leftarrow \mathsf{TKG}(1^k)$, and then sets $PK \leftarrow (pk_{\mathtt{in0}}, pk_{\mathtt{in1}}, pk, ck)$ and $SK \leftarrow (sk_{\mathtt{in0}}, sk_{\mathtt{in1}}, sk, PK)$. $\mathcal{B}_{\mathsf{b2}}$ next runs $\mathsf{tag}^* \leftarrow \mathsf{Com}(ck, (c^*_{\mathtt{in0}} \| c^*_{\mathtt{in1}}); r^*_c)$ and $c^* \leftarrow \mathsf{TEnc}(pk, \mathsf{tag}^*, (c^*_{\mathtt{in0}} \| c^*_{\mathtt{in1}}); r^*_t)$, sets $C^* \leftarrow (\mathsf{tag}^*, c^*)$, and also chooses $K^*_0 \in \{0,1\}^k$ and $b \in \{0,1\}$ uniformly at random. Then, $\mathcal{B}_{\mathsf{b2}}$ runs $\mathcal{A}$, where the decapsulation queries from $\mathcal{A}$ are answered as Game 2 does, which is possible because $\mathcal{B}_{\mathsf{b2}}$ possesses SK.
When $\mathcal{A}$ terminates, $\mathcal{B}_{\mathsf{b2}}$ checks if $\mathcal{A}$ has made a decapsulation query $C = (\mathsf{tag}, c)$ satisfying the conditions of NoBind_2, namely, $\mathsf{tag} = \mathsf{tag}^*$, $c \neq c^*$, $\mathsf{TDec}(sk, \mathsf{tag}, c) = (c_{\mathtt{in0}} \| c_{\mathtt{in1}}) \notin \{(c^*_{\mathtt{in0}} \| c^*_{\mathtt{in1}}), \bot\}$, $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in1}}, c_{\mathtt{in0}}) = \alpha_0 \neq \bot$, $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in1}}, c_{\mathtt{in1}}) = \alpha_1 \neq \bot$, $(\alpha_0 \oplus \alpha_1) = (r_c \| r_t \| K) \in \{0,1\}^{3k}$, and $\mathsf{Com}(ck, (c_{\mathtt{in0}} \| c_{\mathtt{in1}}); r_c) = \mathsf{tag}^*$, and $\mathsf{TEnc}(pk, \mathsf{tag}, (c_{\mathtt{in0}} \| c_{\mathtt{in1}}); r_t) = c$. (Actually, the last condition is redundant for $\mathcal{B}_{\mathsf{b2}}$'s purpose.) If such a query is found, then $\mathcal{B}_{\mathsf{b2}}$ terminates with output $M' = (c_{\mathtt{in0}} \| c_{\mathtt{in1}})$ and $R' = r_c$. Otherwise, $\mathcal{B}_{\mathsf{b2}}$ gives up and aborts.

The above completes the description of $\mathcal{B}_{\mathsf{b}}$. It is easy to see that $\mathcal{B}_{\mathsf{b}}$ does a perfect simulation of Game 2 for $\mathcal{A}$, and whenever $\mathcal{A}$ makes a query that causes the event NoBind_2, $\mathcal{B}_{\mathsf{b2}}$ can find such a query by using SK and output a pair $(M', R') = ((c_{\mathtt{in0}} \| c_{\mathtt{in1}}), r_c)$ satisfying $\mathsf{Com}(ck, M; R) = \mathsf{Com}(ck, M'; R') = \mathsf{tag}^*$ and $M \neq M'$, violating the target-binding property of the commitment scheme $\mathcal{C}$. Therefore, we have $\mathsf{Adv}^{\mathtt{TBind}}_{\mathcal{C},\mathcal{B}_{\mathsf{b}}}(k) = \Pr[\mathsf{NoBind}_2]$. Then, by Equation (2), we have $\mathsf{Adv}^{\mathtt{TBind}}_{\mathcal{C},\mathcal{B}_{\mathsf{b}}}(k) \geq |\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_2]|$, as required. □ (**Claim 1**)

Claim 2 $\Pr[\mathsf{Succ}_2] = \Pr[\mathsf{Succ}_3]$.

Proof of Claim 2. It is sufficient to show that the behavior of the oracle given to $\mathcal{A}$ in Game 2 and that in Game 3 are identical. Let $C = (\mathsf{tag}, c)$ be a decapsulation query that $\mathcal{A}$ makes. If $\mathsf{tag} = \mathsf{tag}^*$, then the query is answered with $\bot$ in Game 2 by definition, while the oracle $\mathsf{AltDecap}(\widehat{SK}_{\mathsf{tag}^*}, C)$ that is given access to $\mathcal{A}$ in Game 3 also returns $\bot$ by definition. Otherwise (i.e. $\mathsf{tag} \neq \mathsf{tag}^*$), by Lemma 3, the result of $\mathsf{Decap}(SK, C)$ and that of $\mathsf{AltDecap}(\widehat{SK}_{\mathsf{tag}^*}, C)$ always agree. This completes the proof. □ (**Claim 2**)

Claim 3 *There exist PPTAs* $\mathcal{B}_{\mathsf{g}}$ *and* $\mathcal{B}_{\mathsf{d}}$ *such that*

$$\Big|\Pr[\mathsf{Succ}_3] - \Pr[\mathsf{Succ}_4]\Big| \leq 2 \cdot \Big(\mathsf{Adv}^{\mathtt{CPA}}_{\Gamma_{\mathtt{in}},\mathcal{B}_{\mathsf{g}}}(k) + \mathsf{Adv}^{\mathtt{TS}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}}(k) + \mathsf{Adv}^{\mathtt{sPA1}}_{\Gamma_{\mathtt{in}},\mathcal{A}',\mathcal{E},2}(k)\Big).$$

We postpone the proof of this claim to the end of the proof of Theorem 1.

Claim 4 *There exists a PPTA* $\mathcal{B}'_{\mathsf{g}}$ *such that* $\mathsf{Adv}^{\mathtt{CPA}}_{\Gamma_{\mathtt{in}},\mathcal{B}'_{\mathsf{g}}}(k) = |\Pr[\mathsf{Succ}_4] - \Pr[\mathsf{Succ}_5]|$.

Proof of Claim 2. Using $\mathcal{A}$ and $\mathcal{E}$ as building blocks, we show how to construct a PPTA $\mathtt{CPA}$ adversary $\mathcal{B}'_{\mathsf{g}}$ with the claimed advantage. The description of $\mathcal{B}'_{\mathsf{g}}$ is as follows:

$\mathcal{B}'_{\mathsf{g}}(pk', c'^*, \alpha'^*_\beta)$**:** (where $\beta \in \{0,1\}$ is $\mathcal{B}'_{\mathsf{g}}$'s challenge bit in its $\mathtt{CPA}$ experiment) $\mathcal{B}'_{\mathsf{g}}$ sets $pk_{\mathtt{in0}} \leftarrow pk'$, $c^*_{\mathtt{in0}} \leftarrow c'^*$, and $\alpha^*_0 \leftarrow \alpha'^*_\beta$. Next, $\mathcal{B}'_{\mathsf{g}}$ generates $(pk_{\mathtt{in1}}, sk_{\mathtt{in1}}) \leftarrow \mathsf{KKG}_{\mathtt{in}}(1^k)$ and $(c^*_{\mathtt{in1}}, \alpha^*_1) \leftarrow \mathsf{Encap}_{\mathtt{in}}(pk_{\mathtt{in1}})$, sets $\alpha^* \leftarrow (\alpha^*_0 \oplus \alpha^*_1)$, and parses α^* as $(r^*_c, r^*_t, K^*_1) \in (\{0,1\}^k)^3$. Then, $\mathcal{B}'_{\mathsf{g}}$ picks $r_g, r'_g \leftarrow \{0,1\}^*$ uniformly at random, and runs $ck \leftarrow \mathsf{CKG}(1^k; r_g)$, $\mathsf{tag}^* \leftarrow \mathsf{Com}(ck, (c^*_{\mathtt{in0}} \| c^*_{\mathtt{in1}}); r^*_c)$, $\widehat{r}_c \leftarrow \mathsf{rSamp}_{\mathcal{C}}(r_g, r^*_c, (c^*_{\mathtt{in0}} \| c^*_{\mathtt{in1}}))$, $(pk, sk) \leftarrow \mathsf{TKG}(1^k; r'_g)$, $\widehat{sk}_{\mathsf{tag}^*} \leftarrow \mathsf{Punc}(sk, \mathsf{tag}^*)$, $c^* \leftarrow \mathsf{TEnc}(pk, \mathsf{tag}^*, (c^*_{\mathtt{in0}} \| c^*_{\mathtt{in1}}); r^*_t)$, and $\widehat{r}_t \leftarrow \mathsf{rSamp}_{\mathcal{T}}(r'_g, r^*_t, \mathsf{tag}^*, (c^*_{\mathtt{in0}} \| c^*_{\mathtt{in1}}))$. Then $\mathcal{B}'_{\mathsf{g}}$ picks $r_{\mathcal{A}} \in \{0,1\}^*$, $K^*_0 \in \{0,1\}^k$, and $b \in \{0,1\}$ all uniformly at random, and sets $PK \leftarrow (pk_{\mathtt{in0}}, pk_{\mathtt{in1}}, pk, ck)$, $C^* \leftarrow (\mathsf{tag}^*, c^*)$, $r_{\mathcal{A}'} \leftarrow (r_{\mathcal{A}}, \widehat{r}_c, \widehat{r}_t, K^*_b)$, and $\mathsf{st}_{\mathcal{E}} \leftarrow (pk_{\mathtt{in0}}, pk_{\mathtt{in1}}, r_{\mathcal{A}'})$. Finally, $\mathcal{B}'_{\mathsf{g}}$ runs $\mathcal{A}(PK, C^*, K^*_b; r_{\mathcal{A}})$.

$\mathcal{B}'_{\mathsf{g}}$ answers $\mathcal{A}$'s decapsulation queries as $\mathsf{AltDecap}'_{\mathcal{E}}$ does, where the initial state of $\mathsf{AltDecap}'_{\mathcal{E}}$ is tag^*, $\widehat{sk}_{\mathsf{tag}^*}$, and $\mathsf{st}_{\mathcal{E}}$. (Note that $\mathsf{st}_{\mathcal{E}}$ is used by $\mathcal{E}$, and may be updated upon each call of $\mathsf{AltDecap}'_{\mathcal{E}}$.)

When $\mathcal{A}$ terminates with output b', $\mathcal{B}'_{\mathsf{g}}$ sets $\beta' \leftarrow (b' \overset{?}{=} b)$, and terminates with output β'.

The above completes the description of $\mathcal{B}'_{\mathsf{g}}$. $\mathcal{B}'_{\mathsf{g}}$'s $\mathtt{CPA}$ advantage can be calculated as follows:

$$\begin{aligned}\mathsf{Adv}^{\mathtt{CPA}}_{\Gamma_{\mathtt{in}},\mathcal{B}'_{\mathsf{g}}}(k) &= 2 \cdot \left|\Pr[\beta' = \beta] - \frac{1}{2}\right| = \Big|\Pr[\beta' = 1|\beta = 1] - \Pr[\beta' = 1|\beta = 0]\Big| \\ &= \Big|\Pr[b' = b|\beta = 1] - \Pr[b' = b|\beta = 0]\Big|.\end{aligned}$$

Consider the case when $\beta = 1$. It is easy to see that in this case, $\mathcal{B}'_{\mathsf{g}}$ simulates Game 4 perfectly for $\mathcal{A}$. Specifically, the real session-key $\alpha'^*_\beta = \alpha'^*_1$ (corresponding to $c^*_{\mathtt{in0}} = c'^*$) is used as α^*_0, and thus $\alpha^* = (\alpha^*_0 \oplus \alpha^*_1) = (r^*_c \| r^*_t \| K^*_1)$ is generated exactly as that in Game 4. All other values are distributed identically to those in Game 4. Furthermore, $\mathcal{B}'_{\mathsf{g}}$ uses $\mathsf{AltDecap}'_{\mathcal{E}}$ for answering $\mathcal{A}$'s decapsulation queries, where the initial state of $\mathsf{AltDecap}'_{\mathcal{E}}$ (and thus the initial state of $\mathcal{E}$) is appropriately generated as those in Game 4. Under this situation, the probability that $\mathcal{A}$ succeeds in guessing b (i.e. $b' = b$ occurs) is exactly the same as the probability that $\mathcal{A}$ does so in Game 4, i.e. $\Pr[b' = b|\beta = 1] = \Pr[\mathsf{Succ}_4]$.

On the other hand, when $\beta = 0$, then $\mathcal{B}'_{\mathsf{g}}$ simulates Game 5 perfectly for $\mathcal{A}$. Specifically, in this case, a uniformly random value $\alpha'^*_\beta = \alpha'^*_0$ is used as α^*_0. Therefore, $\alpha^* = (\alpha^*_0 \oplus \alpha^*_1)$ is also a uniformly random $3k$-bit string, and thus each of r^*_c, r^*_t, and K^*_1 is a uniformly random k-bit string, which is exactly how these values are chosen in Game 5. Since this is the only change from the case of $\beta = 1$, with a similar argument to the above, we have $\Pr[b' = b|\beta = 0] = \Pr[\mathsf{Succ}_5]$.

In summary, we have $\mathsf{Adv}^{\mathtt{CPA}}_{\Gamma_{\mathtt{in}},\mathcal{B}'_{\mathsf{g}}}(k) = |\Pr[\mathsf{Succ}_4] - \Pr[\mathsf{Succ}_5]|$, as required. □ (**Claim 4**)

Claim 5 $\Pr[\mathsf{Succ}_5] = 1/2$.

Proof of Claim 5. This is obvious because in Game 5, the real session-key K^*_1 is made independent of the challenge ciphertext C^*. Since both K^*_1 and K^*_0 are now uniformly random, the view of $\mathcal{A}$ does not contain any information on b. This means that the probability that $\mathcal{A}$ succeeds in guessing the challenge bit is exactly $1/2$. □ (**Claim 5**)

Claims 1 to 5 and Equation (1) guarantee that there exist PPTAs $\mathcal{B}_{\mathtt{b}}$, $\mathcal{B}_{\mathtt{g}}$, $\mathcal{B}_{\mathtt{d}}$, and $\mathcal{B}'_{\mathtt{g}}$ such that

$$\begin{aligned}\mathsf{Adv}^{\mathtt{CCA}}_{\Gamma,\mathcal{A}}(k) \leq 2 \cdot \mathsf{Adv}^{\mathtt{TBind}}_{\mathcal{C},\mathcal{B}_{\mathtt{b}}}(k) + 4 \cdot \mathsf{Adv}^{\mathtt{CPA}}_{\Gamma_{\mathtt{in}},\mathcal{B}_{\mathtt{g}}}(k) + 4 \cdot \mathsf{Adv}^{\mathtt{TS}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathtt{d}}}(k) + 4 \cdot \mathsf{Adv}^{\mathtt{sPA1}}_{\Gamma_{\mathtt{in}},\mathcal{A}',\mathcal{E},2}(k)\\ + 2 \cdot \mathsf{Adv}^{\mathtt{CPA}}_{\Gamma_{\mathtt{in}},\mathcal{B}'_{\mathtt{g}}}(k),\end{aligned}$$

which, due to our assumptions on the building blocks and Lemma 2, implies that $\mathsf{Adv}^{\mathtt{CCA}}_{\Gamma,\mathcal{A}}(k)$ is negligible. Recall that the choice of the PPTA $\mathtt{CCA}$ adversary $\mathcal{A}$ was arbitrarily, and thus for any PPTA $\mathtt{CCA}$ adversary $\mathcal{A}$ we can show a negligible upperbound for $\mathsf{Adv}^{\mathtt{CCA}}_{\Gamma,\mathcal{A}}(k)$ as above.

In order to finish the proof of Theorem 1, it remains to prove Claim 3.

Proof of Claim 3. Note that the difference between Game 3 and Game 4 is how a query $C = (\mathsf{tag}, c)$ satisfying the conditions of Bad_3 (or Bad_4) is answered, and Game 3 and Game 4 proceed identically unless Bad_3 or Bad_4 occurs in the corresponding games. This means that we have

$$\Bigl|\Pr[\mathsf{Succ}_3] - \Pr[\mathsf{Succ}_4]\Bigr| \leq \Pr[\mathsf{Bad}_3] = \Pr[\mathsf{Bad}_4]. \tag{3}$$

We claim the following:

Subclaim 1 $\Pr[\mathsf{Bad}_4] \leq 2 \cdot \Pr[\mathsf{Bad}^*_4]$.

Proof of Subclaim 1. The argument here is essentially the same as the one used in the proof of Claim 4.13 in [18].

Note that the event Bad_4, $\mathsf{Bad}^{(0)}_4$, $\mathsf{Bad}^{(1)}_4$, and Bad^*_4 are triggered once $\mathcal{A}$ makes a query $C = (\mathsf{tag}, c)$ satisfying the conditions that cause these events. Moreover, by definition, if any of the latter three events occurs, then Bad_4 occurs. Furthermore, the bit γ is information-theoretically hidden from $\mathcal{A}$'s view in Game 4. This means that the probability of Bad^*_4 occurring is identical to the probability of the event (in Game 4) that is triggered when (1) $\mathcal{A}$ first makes a query satisfying the conditions of Bad_4, (2) γ is picked "on-the-fly" at this point, and then (3) $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in}\gamma}, c_{\mathtt{in}\gamma}) \neq \mathcal{E}(\mathsf{st}_{\mathcal{E}}, (\gamma + 1, c_{\mathtt{in}\gamma}))$ holds. The probability of this event occurring is $\Pr_{\gamma \leftarrow \{0,1\}}[\mathsf{Bad}_4 \wedge \mathsf{Bad}^{(\gamma)}_4] = \Pr_{\gamma \leftarrow \{0,1\}}[\mathsf{Bad}^{(\gamma)}_4]$ (where the probability is also over Game 4 except the choice of γ). This can be further estimated as follows:

$$\Pr_{\gamma \leftarrow \{0,1\}}[\mathsf{Bad}^{(\gamma)}_4] = \frac{1}{2}\Bigl(\Pr[\mathsf{Bad}^{(0)}_4] + \Pr[\mathsf{Bad}^{(1)}_4]\Bigr) \geq \frac{1}{2}\Pr[\mathsf{Bad}^{(0)}_4 \vee \mathsf{Bad}^{(1)}_4] = \frac{1}{2}\Pr[\mathsf{Bad}_4],$$

where we used $\Pr[\mathsf{Bad}^{(0)}_4 \vee \mathsf{Bad}^{(1)}_4] = \Pr[\mathsf{Bad}_4]$, which is by definition.

In summary, we have $\Pr[\mathsf{Bad}^*_4] \geq \frac{1}{2}\Pr[\mathsf{Bad}_4]$, as required. □ (**Subclaim 1**)

Using Subclaim 1, we can further estimate $\Pr[\mathsf{Bad}_4]$ as follows:

$$\begin{aligned}\Pr[\mathsf{Bad}_4] &\leq 2 \cdot \Pr[\mathsf{Bad}^*_4]\\ &\leq 2 \cdot \Bigl(\Bigl|\Pr[\mathsf{Bad}^*_4] - \Pr[\mathsf{Bad}^*_5]\Bigr| + \Pr[\mathsf{Bad}^*_5]\Bigr)\\ &\leq 2 \cdot \Bigl(\Bigl|\Pr[\mathsf{Bad}^*_4] - \Pr[\mathsf{Bad}^*_5]\Bigr| + \Pr[\mathsf{Bad}_5]\Bigr)\\ &\leq 2 \cdot \Bigl(\Bigl|\Pr[\mathsf{Bad}^*_4] - \Pr[\mathsf{Bad}^*_5]\Bigr| + \Bigl|\Pr[\mathsf{Bad}_5] - \Pr[\mathsf{Bad}_6]\Bigr| + \Pr[\mathsf{Bad}_6]\Bigr),\end{aligned} \tag{4}$$

where we used $\Pr[\mathsf{Bad}^*_5] \leq \Pr[\mathsf{Bad}_5]$ in the third inequality, which is again by definition. It remains to upperbound the right hand side of the above inequality.

Subclaim 2 *There exists a PPTA $\mathcal{B}_{\mathtt{g}}$ such that $\mathsf{Adv}^{\mathtt{CPA}}_{\Gamma_{\mathtt{in}},\mathcal{B}_{\mathtt{g}}}(k) = |\Pr[\mathsf{Bad}^*_4] - \Pr[\mathsf{Bad}^*_5]|$.*

Proof of Subclaim 2. Using $\mathcal{A}$ and $\mathcal{E}$ as building blocks, we show how to construct a PPTA CPA adversary $\mathcal{B}_{\mathtt{g}}$ with the claimed advantage. The description of $\mathcal{B}_{\mathtt{g}}$ is as follows:

$\mathcal{B}_{\mathtt{g}}(pk', c'^*, \alpha'^*_\beta)$**:** (where $\beta \in \{0,1\}$ is $\mathcal{B}_{\mathtt{g}}$'s challenge bit in its CPA experiment) $\mathcal{B}_{\mathtt{g}}$ picks $\gamma \in \{0,1\}$ uniformly at random, then sets $pk_{\mathtt{in}(1-\gamma)} \leftarrow pk'$, $c^*_{\mathtt{in}(1-\gamma)} \leftarrow c'^*$, and $\alpha^*_{1-\gamma} \leftarrow \alpha'^*_\beta$. Next, $\mathcal{B}_{\mathtt{g}}$ generates $(pk_{\mathtt{in}\gamma}, sk_{\mathtt{in}\gamma}) \leftarrow \mathsf{KKG}_{\mathtt{in}}(1^k)$ and $(c^*_{\mathtt{in}\gamma}, \alpha^*_\gamma) \leftarrow \mathsf{Encap}_{\mathtt{in}}(pk_{\mathtt{in}\gamma})$, sets $\alpha^* \leftarrow (\alpha^*_0 \oplus \alpha^*_1)$, and parses α^* as $(r^*_c, r^*_t, K^*_1) \in (\{0,1\}^k)^3$. Then, $\mathcal{B}_{\mathtt{g}}$ prepares $K^*_1, K^*_0 \in \{0,1\}^k$, $b \in \{0,1\}$, $PK = (pk_{\mathtt{in0}}, pk_{\mathtt{in1}}, pk, c)$, $C^* = (\mathsf{tag}^*, c^*)$, $\widehat{sk}_{\mathsf{tag}^*}$, and $\mathsf{st}_{\mathcal{E}} = (pk_{\mathtt{in0}}, pk_{\mathtt{in1}}, r_{\mathcal{A}'} = (r_{\mathcal{A}}, \widehat{r}_c, \widehat{r}_t, K^*_b))$, exactly as $\mathcal{B}'_{\mathtt{g}}$ in the proof of Claim 4 does. Finally, $\mathcal{B}_{\mathtt{g}}$ runs $\mathcal{A}(PK, C^*, K^*_b; r_{\mathcal{A}})$ until it terminates, where $\mathcal{B}_{\mathtt{g}}$ answers $\mathcal{A}$'s queries in exactly the same way as $\mathcal{B}'_{\mathtt{g}}$ does.
When $\mathcal{A}$ terminates, $\mathcal{B}_{\mathtt{g}}$ checks whether $\mathcal{A}$ has submitted a decapsulation query $C = (\mathsf{tag}, c)$ that satisfies the conditions of Bad^*_4 (i.e. (1) $\mathsf{tag} \neq \mathsf{tag}^*$, (2) $\widehat{\mathsf{TDec}}(\widehat{sk}_{\mathsf{tag}^*}, \mathsf{tag}, c) = (c_{\mathtt{in0}} \| c_{\mathtt{in1}}) \neq \bot$, and (3) $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in}\gamma}, c_{\mathtt{in}\gamma}) \neq \mathcal{E}(\mathsf{st}_{\mathcal{E}}, c_{\mathtt{in}\gamma})$ hold), which can be checked by using $sk_{\mathtt{in}\gamma}$. If such a query is found, the $\mathcal{B}_{\mathtt{g}}$ sets $\beta' \leftarrow 1$, otherwise sets $\beta' \leftarrow 0$, and terminates with output β'.

The above completes the description of $\mathcal{B}_{\mathtt{g}}$. Let $\mathsf{Bad}^*_{\mathcal{B}}$ be the event that $\mathcal{A}$ submits a decapsulation query that satisfies the conditions (1), (2), and (3) of Bad^*_4, in the experiment simulated by $\mathcal{B}_{\mathtt{g}}$. Note that $\mathcal{B}_{\mathtt{g}}$ outputs $\beta' = 1$ only when $\mathsf{Bad}^*_{\mathcal{B}}$ occurs. Therefore, $\mathcal{B}_{\mathtt{g}}$'s CPA advantage can be calculated as follows:

$$\begin{aligned}\mathsf{Adv}^{\mathtt{CPA}}_{\Gamma_{\mathtt{in}},\mathcal{B}_{\mathtt{g}}}(k) &= 2 \cdot \left|\Pr[\beta' = \beta] - \frac{1}{2}\right| = \Big|\Pr[\beta' = 1|\beta = 1] - \Pr[\beta' = 1|\beta = 0]\Big| \\ &= \Big|\Pr[\mathsf{Bad}^*_{\mathcal{B}}|\beta = 1] - \Pr[\mathsf{Bad}^*_{\mathcal{B}}|\beta = 0]\Big|.\end{aligned}$$

With essentially the same arguments as in the proof of Claim 4, we can see that $\mathcal{B}_{\mathtt{g}}$ does a perfect simulation of Game 4 for $\mathcal{A}$ if $\beta = 1$, and does a perfect simulation of Game 5 for $\mathcal{A}$ if $\beta = 0$. In particular, the only difference from the proof of Claim 4 is in which of the positions $(pk_{\mathtt{in0}}, c^*_{\mathtt{in0}}, \alpha^*_0)$ or $(pk_{\mathtt{in1}}, c^*_{\mathtt{in1}}, \alpha^*_1)$ $\mathcal{B}_{\mathtt{g}}$ embeds $\mathcal{B}_{\mathtt{g}}$'s instance of the CPA experiment. In the proof of Claim 4, the reduction algorithm $\mathcal{B}'_{\mathtt{g}}$ embeds its challenge into $(pk_{\mathtt{in0}}, c^*_{\mathtt{in0}}, \alpha^*_0)$, while in the current proof, the reduction algorithm $\mathcal{B}_{\mathtt{g}}$ embeds its challenge into $(pk_{\mathtt{in}(1-\gamma)}, c^*_{\mathtt{in}(1-\gamma)}, \alpha^*_{1-\gamma})$ for a random $\gamma \in \{0,1\}$. It is easy to see that even after this change, if $\beta = 1$, then the view of $\mathcal{A}$ is identical to that in Game 4, and if $\beta = 0$, then the view of $\mathcal{A}$ is identical to that in Game 5.

Under the situation, the probability that $\mathsf{Bad}^*_{\mathcal{B}}$ occurs in the experiment simulated by $\mathcal{B}_{\mathtt{g}}$ in case $\beta = 1$ (resp. $\beta = 0$) is identical to the probability that Bad^*_4 (resp. Bad^*_5) occurs in Game 4 (resp. Game 5), namely, we have $\Pr[\mathsf{Bad}^*_{\mathcal{B}}|\beta = 1] = \Pr[\mathsf{Bad}^*_4]$ and $\Pr[\mathsf{Bad}^*_{\mathcal{B}}|\beta = 0] = \Pr[\mathsf{Bad}^*_5]$.

In summary, we have $\mathsf{Adv}^{\mathtt{CPA}}_{\Gamma_{\mathtt{in}},\mathcal{B}_{\mathtt{g}}}(k) = |\Pr[\mathsf{Bad}^*_4] - \Pr[\mathsf{Bad}^*_5]|$, as required. □ (**Subclaim 2**)

Subclaim 3 *There exists a PPTA $\mathcal{B}_{\mathtt{d}}$ such that $\mathsf{Adv}^{\mathtt{TS}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathtt{d}}}(k) = |\Pr[\mathsf{Bad}_5] - \Pr[\mathsf{Bad}_6]|$.*

Proof of Subclaim 3. Using $\mathcal{A}$ and $\mathcal{E}$ as building blocks, we show how to construct a PPTA $\mathcal{B}$ that has the claimed advantage in distinguishing the distributions considered in Lemma 2. The description of $\mathcal{B}_{\mathtt{d}} = (\mathcal{B}_{\mathtt{d1}}, \mathcal{B}_{\mathtt{d2}})$ as follows:

$\mathcal{B}_{\mathtt{d1}}(1^k)$**:** $\mathcal{B}_{\mathtt{d1}}$ first runs $(pk_{\mathtt{in0}}, sk_{\mathtt{in0}}) \leftarrow \mathsf{KKG}_{\mathtt{in}}(1^k)$, $(pk_{\mathtt{in1}}, sk_{\mathtt{in1}}) \leftarrow \mathsf{KKG}_{\mathtt{in}}(1^k)$, $(c^*_{\mathtt{in0}}, \alpha^*_0) \leftarrow \mathsf{Encap}_{\mathtt{in}}(pk_{\mathtt{in0}})$, and $(c^*_{\mathtt{in1}}, \alpha^*_1) \leftarrow \mathsf{Encap}_{\mathtt{in}}(pk_{\mathtt{in1}})$. Then, $\mathcal{B}_{\mathtt{d1}}$ sets $M \leftarrow (c^*_{\mathtt{in0}} \| c^*_{\mathtt{in1}})$ and $\mathsf{st}_{\mathcal{B}} \leftarrow$ ($\mathcal{B}_{\mathtt{d1}}$'s entire view), and terminates with output $(M, \mathsf{st}_{\mathcal{B}})$.

$\mathcal{B}_{\mathsf{d2}}(\mathsf{st}_{\mathcal{B}}, ck, \mathsf{tag}^*, pk, c^*, \widehat{sk}_{\mathtt{tag}^*}, \widehat{r}_c, \widehat{r}_t)$**:** $\mathcal{B}_{\mathsf{d2}}$ sets $PK \leftarrow (pk_{\mathtt{in0}}, pk_{\mathtt{in1}}, pk, ck)$ and $C^* \leftarrow (\mathsf{tag}^*, c^*)$, picks $K^* \in \{0,1\}^*$ and $r_{\mathcal{A}} \in \{0,1\}^*$ uniformly at random, and then sets $r_{\mathcal{A}'} \leftarrow (r_{\mathcal{A}}, \widehat{r}_c, \widehat{r}_t, K^*)$ and $\mathsf{st}_{\mathcal{E}} \leftarrow (pk_{\mathtt{in0}}, pk_{\mathtt{in1}}, r_{\mathcal{A}'})$. (Recall that K_0^* and K_1^* in Games 5 and 6 are distributed identically, and thus it is sufficient to choose just a single value K^* and pretend as if K^* is K_b^*.) Then $\mathcal{B}_{\mathsf{d2}}$ runs $\mathcal{A}(PK, C^*, K^*; r_{\mathcal{A}})$.

$\mathcal{B}_{\mathsf{d2}}$ answers $\mathcal{A}$'s queries as Game 5 does, which is possible because $\mathcal{B}_{\mathsf{d2}}$ possesses $\widehat{sk}_{\mathtt{tag}^*}$ and $\mathsf{st}_{\mathcal{E}}$, and thus $\mathcal{B}_{\mathsf{d2}}$ can run $\mathsf{AltDecap}'_{\mathcal{E}}$ (which internally runs the extractor $\mathcal{E}(\mathsf{st}_{\mathcal{E}}, \cdot)$).

When $\mathcal{A}$ terminates, $\mathcal{B}_{\mathsf{d2}}$ checks whether $\mathcal{A}$ has submitted a query that satisfies the conditions of Bad_5, which can be checked by using $sk_{\mathtt{in0}}$ and $sk_{\mathtt{in1}}$ that $\mathcal{B}_{\mathsf{d2}}$ possesses. If such a query is found, then $\mathcal{B}_{\mathsf{d2}}$ outputs 1, otherwise outputs 0, and terminates.

The above completes the description of $\mathcal{B}_{\mathsf{d}}$. Let $\mathsf{Bad}_{\mathcal{B}}$ be the event that $\mathcal{A}$ submits a decapsulation query $C = (\mathsf{tag}, c)$ that satisfies the conditions of Bad_5 in the experiment simulated by $\mathcal{B}_{\mathsf{d}}$ (i.e. the query satisfying (1) $\mathsf{tag} \neq \mathsf{tag}^*$, (2) $\widehat{\mathsf{TDec}}(\widehat{sk}_{\mathtt{tag}^*}, \mathsf{tag}, c) = (c_{\mathtt{in0}} \| c_{\mathtt{in1}}) \neq \bot$, and (3) $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in0}}, c_{\mathtt{in0}}) \neq \mathcal{E}(\mathsf{st}_{\mathcal{E}}, c_{\mathtt{in0}})$ or $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in1}}, c_{\mathtt{in1}}) \neq \mathcal{E}(\mathsf{st}_{\mathcal{E}}, c_{\mathtt{in1}})$). Note that $\mathcal{B}_{\mathsf{d}}$ submits 1 only when $\mathsf{Bad}_{\mathcal{B}}$ occurs. Therefore, $\mathcal{B}_{\mathsf{d}}$'s advantage $\mathsf{Adv}^{\mathtt{TS}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}}(k)$ can be calculated as follows:

$$\begin{aligned}\mathsf{Adv}^{\mathtt{TS}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}}(k) &= \Big|\Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Real}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}}(k) = 1] - \Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Sim}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}}(k) = 1]\Big| \\ &= \Big|\Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Real}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}} : \mathsf{Bad}_{\mathcal{B}}] - \Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Sim}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}}(k) : \mathsf{Bad}_{\mathcal{B}}]\Big|.\end{aligned}$$

Consider the case when $\mathcal{B}_{\mathsf{d}}$ is run in the "real" experiment $\mathsf{Expt}^{\mathtt{TS\text{-}Real}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}}(k)$. It is easy to see that in this case, $\mathcal{B}_{\mathsf{d}}$ simulates Game 5 perfectly for $\mathcal{A}$. Specifically, ck, pk, tag^*, c^*, and $\widehat{sk}_{\mathtt{tag}^*}$ are generated from CKG, TKG, Com, TEnc, and Punc, respectively, in such a way that tag^* is a commitment of $(c^*_{\mathtt{in0}} \| c^*_{\mathtt{in1}})$ and c^* is an encryption of $(c^*_{\mathtt{in0}} \| c^*_{\mathtt{in1}})$ under the tag tag^*. Furthermore, $\widehat{r}_c$ and $\widehat{r}_t$ are generated from $\mathsf{rSamp}_{\mathcal{C}}$ and $\mathsf{rSamp}_{\mathcal{T}}$, respectively, which is how they are generated in Game 5. Under the situation, the probability that $\mathcal{A}$ submits a decapsulation query that causes the event $\mathsf{Bad}_{\mathcal{B}}$ is exactly the same as the probability that $\mathcal{A}$ does so in Game 5. That is, we have $\Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Real}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}}(k) : \mathsf{Bad}_{\mathcal{B}}] = \Pr[\mathsf{Bad}_5]$.

On the other hand, consider the case when $\mathcal{B}_{\mathsf{d}}$ is run in the "simulated" experiment $\mathsf{Expt}^{\mathtt{TS\text{-}Sim}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}}(k)$. In this case, $\mathcal{B}_{\mathsf{d}}$ simulates Game 6 perfectly for $\mathcal{A}$. Specifically, (ck, tag^*) and $(pk, c^*, \widehat{sk}_{\mathtt{tag}^*})$ are generated by $\mathsf{oSamp}_{\mathcal{C}}(1^k; \widehat{r}_c)$ and $\mathsf{oSamp}_{\mathcal{T}}(\mathsf{tag}^*; \widehat{r}_t)$ with uniformly chosen randomness $\widehat{r}_c$ and $\widehat{r}_t$, respectively, and this is exactly how these values are generated in Game 6. Since this is the only change from the above case, with a similar argument we have $\Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Sim}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}}(k) : \mathsf{Bad}_{\mathcal{B}}] = \Pr[\mathsf{Bad}_6]$.

In summary, we have $\mathsf{Adv}^{\mathtt{TS}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}}(k) = |\Pr[\mathsf{Bad}_5] - \Pr[\mathsf{Bad}_6]|$, as required. □ (**Subclaim 3**)

Subclaim 4 $\mathsf{Adv}^{\mathtt{sPA1}}_{\Gamma_{\mathtt{in}},\mathcal{A}',\mathcal{E},2}(k) = \Pr[\mathsf{Bad}_6]$.

Proof of Subclaim 4. Note that the view of $\mathcal{A}$ in Game 6 is exactly the same as the view of $\mathcal{A}$ when it is internally run by $\mathcal{A}'$ in the situation where $\mathcal{A}'$ is run in the experiment $\mathsf{Expt}^{\mathtt{sPA1}}_{\Gamma_{\mathtt{in}},\mathcal{A}',\mathcal{E},2}(k)$ with the extractor $\mathcal{E}$. Therefore, the probability that $\mathcal{A}$ submits a query that causes the event Bad_6 in Game 6, is exactly the same as the probability that $\mathcal{A}'$ submits a query to $\mathcal{E}$ that makes the experiment $\mathsf{Expt}^{\mathtt{sPA1}}_{\Gamma_{\mathtt{in}},\mathcal{A}',\mathcal{E},2}(k)$ outputs 1 (i.e. $\mathcal{A}'$ submits a query of the form $(j+1, c_{\mathtt{in}j})$ such that $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in}j}, c_{\mathtt{in}j}) \neq \mathcal{E}(\mathsf{st}_{\mathcal{E}}, (j+1, c_{\mathtt{in}j}))$ for some $j \in \{0,1\}$). □ (**Subclaim 4**)

Equations (3), (4), and Subclaims 2 to 4 imply Claim 3. □ (**Claim 3**)

This concludes the proof of Theorem 1. □ (**Theorem 1**)

```
$\overline{\mathsf{KKG}}(1^k)$ :
    $(pk_{\mathtt{in}}, sk_{\mathtt{in}}) \leftarrow \mathsf{KKG}_{\mathtt{in}}(1^k)$
    $(pk, sk) \leftarrow \mathsf{TKG}(1^k)$
    $ck \leftarrow \mathsf{CKG}(1^k)$
    $PK \leftarrow (pk_{\mathtt{in}}, pk, ck)$
    $SK \leftarrow (sk_{\mathtt{in}}, sk, PK)$
    Return $(PK, SK)$.
```

```
$\overline{\mathsf{Encap}}(PK)$ :
    $(pk_{\mathtt{in}}, pk, ck) \leftarrow PK$
    $(c_{\mathtt{in}}, \alpha) \leftarrow \mathsf{Encap}_{\mathtt{in}}(pk_{\mathtt{in}})$
    Parse $\alpha$ as $(r_c, r_t, K) \in (\{0,1\}^k)^3$
    $\mathsf{tag} \leftarrow \mathsf{Com}(ck, c_{\mathtt{in}}; r_c)$
    $c \leftarrow \mathsf{TEnc}(pk, \mathsf{tag}, c_{\mathtt{in}}; r_t)$
    $C \leftarrow (\mathsf{tag}, c)$.
    Return $(C, K)$.
```

```
$\overline{\mathsf{Decap}}(SK, C)$ :
    $(sk_{\mathtt{in}}, sk, PK) \leftarrow SK$
    $(pk_{\mathtt{in}}, pk, ck) \leftarrow PK$
    $(\mathsf{tag}, c) \leftarrow C$
    $c_{\mathtt{in}} \leftarrow \mathsf{TDec}(sk, \mathsf{tag}, c)$
    If $c_{\mathtt{in}} = \perp$ then return $\perp$.
    $\alpha \leftarrow \mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in}}, c_{\mathtt{in}})$
    If $\alpha = \perp$ then return $\perp$.
    Parse $\alpha$ as $(r_c, r_t, K) \in (\{0,1\}^k)^3$
    If $\mathsf{Com}(ck, c_{\mathtt{in}}; r_c) = \mathsf{tag}$
      and $\mathsf{TEnc}(pk, \mathsf{tag}, c_{\mathtt{in}}; r_t) = c$
        then return $K$ else return $\perp$
```

Fig. 4. The second proposed construction: the KEM $\overline{\Gamma}$ based on a KEM $\Gamma_{\mathtt{in}}$, a commitment scheme $\mathcal{C}$, and a PTBE scheme $\mathcal{T}$.

4.2 Second Construction

Let $\Gamma_{\mathtt{in}} = (\mathsf{KKG}_{\mathtt{in}}, \mathsf{Encap}_{\mathtt{in}}, \mathsf{Decap}_{\mathtt{in}})$ be a KEM whose ciphertext length is $n = n(k)$ and whose session-key space is $\{0,1\}^{3k}$ for k-bit security. Let $\mathcal{T} = (\mathsf{TKG}, \mathsf{TEnc}, \mathsf{TDec}, \mathsf{Punc}, \widehat{\mathsf{TDec}})$ be a PTBE scheme and $\mathcal{C} = (\mathsf{CKG}, \mathsf{Com})$ be a commitment scheme. We require the plaintext space of TEnc and the message space of Com to be $\{0,1\}^n$, and the randomness space of TEnc and that of Com to be $\{0,1\}^k$ for k-bit security. Then, our second proposed KEM $\overline{\Gamma} = (\overline{\mathsf{KKG}}, \overline{\mathsf{Encap}}, \overline{\mathsf{Decap}})$ is constructed as in Fig. 4.

The security of $\overline{\Gamma}$ is guaranteed by the following theorem.

Theorem 2. *Assume that the KEM $\Gamma_{\mathtt{in}}$ is* $1\text{-}\mathtt{CCA}$ *secure and* $\mathtt{sPA1}_1$ *secure, the commitment scheme $\mathcal{C}$ is target-binding and trapdoor simulatable, and the PTBE scheme $\mathcal{T}$ is trapdoor simulatable. Then, the KEM $\overline{\Gamma}$ constructed as in Fig. 4 is* $\mathtt{CCA}$ *secure.*

The proof of this theorem proceeds very similarly to the proof of Theorem 1, and thus we only explain the difference here, and will show the formal proof in Appendix C.3.

Recall that in the proof Theorem 1, the "bad" queries (for which the extractor fails to extract correct decapsulation results) are dealt with due to the property of "multiple encryption" of two instances of the KEM $\Gamma_{\mathtt{in}}$ with public keys $(pk_{\mathtt{in0}}, pk_{\mathtt{in1}})$. In particular, the reduction algorithm in the proof of Subclaim 2 that attacks the $\mathtt{CPA}$ security of the underlying KEM $\Gamma_{\mathtt{in}}$, uses one of secret keys $sk_{\mathtt{in}\gamma}$ (corresponding to $pk_{\mathtt{in}\gamma}$) to detect whether the bad event occurs, while embedding its $\mathtt{CPA}$ instance regarding $\Gamma_{\mathtt{in}}$ into the other position, i.e. into $(pk_{\mathtt{in}(1-\gamma)}, c_{\mathtt{in}(1-\gamma)})$. This strategy works thanks to the argument regarding the probabilities given in the proof of Subclaim 1 (which is in turn based on the proof of [18, Claim 4.13]). However, for this argument to work, it seems to us that we inherently have to rely on the $\mathtt{sPA1}_2$ security of $\Gamma_{\mathtt{in}}$, in order for the reduction algorithms (especially, the reduction algorithms attacking the $\mathtt{CPA}$ of $\Gamma_{\mathtt{in}}$) to simulate the decapsulation oracle for an adversary $\mathcal{A}$.

The simple idea employed in our second construction is to change the mechanism of detecting the bad queries by relying on the $1\text{-}\mathtt{CCA}$ security of $\Gamma_{\mathtt{in}}$, so that a reduction algorithm can check (by its access to the decapsulation oracle) whether $\mathcal{A}$ has submitted a bad decapsulation query. This allows us to use $\Gamma_{\mathtt{in}}$ only in the "single" key setting, leading to only requiring it to be $\mathtt{sPA1}_1$ secure. By employing this idea, a security analysis similar to the recent constructions [45, 32, 38, 41] works, and for the other parts of the security proof (other than the analysis regarding dealing with the bad decapsulation queries) are essentially the same as those in the proof of Theorem 1. For more details, see Appendix C.3.

On the Merits of the Second Construction. Since we need to use a KEM which simultaneously satisfies 1-CCA and $\mathtt{sPA1}_1$ security for our second construction, a natural question would be whether we can construct such a scheme. We note that we can achieve such a KEM from a CPA secure PKE (or a KEM) which is also $\mathtt{sPA1}_{2k}$ secure. Specifically, Dodis and Fiore [22, Appendix C] showed how to construct a 1-CCA secure PKE scheme from the combination of a CPA secure PKE scheme and a one-time secure signature scheme (in which $2k$ independently generated public keys are arranged as in the "DDN-lite" construction, but a message is encoded and encrypted in a k-out-of-k fashion, rather than encrypting the same message under k public keys as done in [24]). We note that we can slightly optimize their construction by using a CPA secure KEM instead of a PKE scheme, and provide its security proof in Appendix D.

However, if we implement a 1-CCA and $\mathtt{sPA1}_1$ secure KEM from a CPA and $\mathtt{sPA1}_{2k}$ secure KEM, there is no merit compared to our first construction (that only requires a CPA and $\mathtt{sPA1}_2$ secure KEM), both in terms of the assumptions and the efficiency. So far, we do not know a better way to construct a 1-CCA and $\mathtt{sPA1}_1$ secure scheme than the approach that relies on [22, Appendix C]. We would like to however emphasize that the point of our second construction is that it may in the future be possible to come up with a direct construction of a KEM (or a PKE scheme) satisfying the requirements for the second construction, from assumptions weaker than those required in our first construction or the combination of our second construction and the Dodis-Fiore construction. We believe that such a possibility of the existence of better constructions can be a raison d'etre of our second construction. In particular, we actually do not need the "full" power of 1-CCA security, but a (seemingly) much weaker security notion such that CPA security holds in the presence of one "plaintext-checking" query [48, 1]. More specifically, a plaintext-checking query (for a KEM it could be called a session-key-checking query, but we stick to the terminology in [48]) is a query of the form (c, K), and its reply is the one-bit $(\mathsf{Decap}(sk, c) \stackrel{?}{=} K)$. This could be a hint for the next step.

We would also like to note that even if using the result based on [22], we still achieve the property of "separating" the requirement that a single PKE scheme (or a KEM) needs to satisfy "plaintext awareness" and a "simulatability property" simultaneously in [19]. This is another merit of our second construction.

Acknowledgement. The authors would like to thank the members of the study group "Shin-Akarui-Angou-Benkyou-Kai," and the anonymous reviewers for their helpful comments and suggestions.

References

1. M. Abdalla, F. Benhamouda, and D. Pointcheval. Public-key encryption indistinguishable under plaintext-checkable attacks. In *Proc. of PKC 2015*, volume 9020 of *LNCS*, pages 332–352. Springer, 2015.
2. B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S.P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In *Proc. of CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, 2001.
3. M. Bellare, V.T. Hoang, and S. Keelveedhi. Instantiating random oracles via UCEs. In *Proc. of CRYPTO 2013(2)*, volume 8043 of *LNCS*, pages 398–415. Springer, 2013.
4. M. Bellare, D. Hofheinz, and E. Kiltz. Subtleties in the definition of IND-CCA: When and how should challenge decryption be disallowed? *J. Cryptology*, 28(1):29–48, 2015.
5. M. Bellare, D. Hofheinz, and S. Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In *Proc. of EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 1–35. Springer, 2009.

6. M. Bellare and A. Palacio. Towards plaintext-aware public-key encryption without random oracles. In *Proc. of ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 48–62. Springer, 2004.
7. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proc. of CCS 1993*, pages 62–73. ACM, 1993.
8. M. Bellare and P. Rogaway. Optimal asymmetric encryption. In *Proc. of EUROCRYPT 1994*, volume 950 of *LNCS*, pages 92–111. Springer, 1995.
9. D. Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In *Proc. of CRYPTO 1998*, volume 1462 of *LNCS*, pages 1–12. Springer, 1998.
10. R. Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In *Proc. of CRYPTO 1997*, volume 1294 of *LNCS*, pages 455–469. Springer, 1997.
11. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proc. of FOCS 2001*, pages 136–145. IEEE Computer Society, 2001.
12. R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multi-party computation. In *Proc. of STOC 1996*, pages 639–648. ACM, 1996.
13. R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In *Proc. of EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 207–222. Springer, 2004.
14. Y. Chen and Z. Zhang. Publicly evaluable pseudorandom functions and their applications. In *Proc. of SCN 2014*, volume 8642 of *LNCS*, pages 115–134. Springer, 2014.
15. S.G. Choi, D. Dachman-Soled, T. Malkin, and H. Wee. Improved non-committing encryption with applications to adaptively secure protocols. In *Proc. of ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 287–302. Springer, 2009.
16. R. Cramer, G. Hanaoka, D. Hofheinz, H. Imai, E. Kiltz, R. Pass, A. Shelat, and V. Vaikuntanathan. Bounded CCA2-secure encryption. In *Proc. of ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 502–518. Springer, 2007.
17. R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Computing*, 33(1):167–226, 2003.
18. D. Dachman-Soled. A black-box construction of a CCA2 encryption scheme from a plaintext aware encryption scheme, 2013. Full version of [19]. http://eprint.iacr.org/2013/680.
19. D. Dachman-Soled. A black-box construction of a CCA2 encryption scheme from a plaintext aware (sPA1) encryption scheme. In *Proc. of PKC 2014*, volume 8383 of *LNCS*, pages 37–55. Springer, 2014.
20. I. Damgård and J.B. Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In *Proc. of CRYPTO 2000*, volume 1880 of *LNCS*, pages 432–450. Springer, 2000.
21. A.W. Dent. The Cramer-Shoup encryption is plaintext aware in the standard model. In *Proc. of EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 289–307. Springer, 2006.
22. Y. Dodis and D. Fiore. Interactive encryption and message authentication, 2013. Full version of [23]. http://eprint.iacr.org/2013/817.
23. Y. Dodis and D. Fiore. Interactive encryption and message authentication. In *Proc. of SCN 2014*, volume 8642 of *LNCS*, pages 494–513. Springer, 2014.
24. D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. In *Proc. of STOC 1991*, pages 542–552. ACM, 1991.
25. E. Fujisaki and T. Okamoto. How to enhance the security of public-key encryption at minimum cost. In *Proc. of PKC 1999*, volume 1560 of *LNCS*, pages 53–68. Springer, 1999.
26. E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Proc. of CRYPTO 1999*, volume 1666 of *LNCS*, pages 537–554. Springer, 1999.
27. S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B Waters. Candidate indistinguishability obfuscation and functional encryption for all curcuits. In *Proc. of FOCS 2013*, pages 40–49. IEEE Computer Society, 2013.
28. O. Goldreich and R.D. Rothblum. Enhancements of trapdoor permutations. *J. of Cryptology*, 26(3):484–512, 2013.
29. M. Hajiabadi and B.M. Kapron. Reproducible circularly-secure bit encryption: Applications and realizations. In *Proc. of CRYPTO 2015(1)*, volume 9215 of *LNCS*, pages 224–243. Springer, 2015.
30. B. Hemenway and R. Ostrovsky. On homomorphic encryption and chosen-ciphertext security. In *Proc. of PKC 2012*, volume 7293 of *LNCS*, pages 52–65. Springer, 2012.
31. B. Hemenway and R. Ostrovsky. Building lossy trapdoor functions from lossy encryption. In *Proc. of ASIACRYPT 2013(2)*, volume 8270 of *LNCS*, pages 241–260. Springer, 2013.
32. S. Hohenberger, A. Lewko, and B. Waters. Detecting dangerous queries: A new approach for chosen ciphertext security. In *Proc. of EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 663–681. Springer, 2012.
33. E. Kiltz. Chosen-ciphertext security from tag-based encryption. In *Proc. of TCC 2006*, volume 3876 of *LNCS*, pages 581–600. Springer, 2006.

34. E. Kiltz, P. Mohassel, and A. O'Neill. Adaptive trapdoor functions and chosen-ciphertext security. In *Proc. of EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 673–692. Springer, 2010.
35. H. Lin and S. Tessaro. Amplification of chosen-ciphertext security. In *Proc. of EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 503–519. Springer, 2013.
36. Y. Lindell. A simpler construction of CCA2-secure public-key encryption under general assumptions. In *Proc. of EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 241–254. Springer, 2003.
37. B. Lynn, M. Prabhakaran, and A. Sahai. Positive results and techniques for obfuscation. In *Proc. of EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 20–39. Springer, 2004.
38. T. Matsuda and G. Hanaoka. Achieving chosen ciphertext security from detectable public key encryption efficiently via hybrid encryption. In *Proc. of IWSEC 2013*, volume 8231 of *LNCS*, pages 226–243. Springer, 2013.
39. T. Matsuda and G. Hanaoka. Chosen ciphertext security via point obfuscation. In *Proc. of TCC 2014*, volume 8349 of *LNCS*, pages 95–120. Springer, 2014.
40. T. Matsuda and G. Hanaoka. Chosen ciphertext security via UCE. In *Proc. of PKC 2014*, volume 8383 of *LNCS*, pages 56–76. Springer, 2014.
41. T. Matsuda and G. Hanaoka. An asymptotically optimal method for converting bit encryption to multi-bit encryption. In *Proc. of ASIACRYPT 2015(1)*, volume 9452 of *LNCS*, pages 415–442. Springer, 2015.
42. T. Matsuda and G. Hanaoka. Constructing and understanding chosen ciphertext security via puncturable key encapsulation mechanisms. In *Proc. of TCC 2015(1)*, volume 9014 of *LNCS*, pages 561–590. Springer, 2015.
43. P. Mol and S. Yilek. Chosen-ciphertext security from slightly lossy trapdoor functions. In *Proc. of PKC 2010*, volume 6056 of *LNCS*, pages 296–311. Springer, 2010.
44. S. Myers, M. Sergi, and A. Shelat. Blackbox construction of a more than non-malleable CCA1 encryption scheme from plaintext awareness. In *Proc. of SCN 2012*, volume 7485 of *LNCS*, pages 149–165. Springer, 2012.
45. S. Myers and A. Shelat. Bit encryption is complete. In *Proc. of FOCS 2009*, pages 607–616. IEEE Computer Society, 2009.
46. M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *Proc. of STOC 1989*, pages 33–43. ACM, 1989.
47. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proc. of STOC 1990*, pages 427–437. ACM, 1990.
48. T. Okamoto and D. Pointcheval. REACT: Rapid enhanced-security asymmetric cryptosystem transform. In *Proc. of CT-RSA 2001*, volume 2020 of *LNCS*, pages 159–174. Springer, 2001.
49. C. Peikert and B. Waters. Lossy trapdoor functions and their applications. In *Proc. of STOC 2008*, pages 187–196. ACM, 2008.
50. C. Rackoff and D.R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Proc. of CRYPTO 1991*, volume 576 of *LNCS*, pages 433–444. Springer, 1992.
51. J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proc. of STOC 1990*, pages 387–394. ACM, 1990.
52. A. Rosen and G. Segev. Chosen-ciphertext security via correlated products. In *Proc. of TCC 2009*, volume 5444 of *LNCS*, pages 419–436. Springer, 2009.
53. A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *Proc. of FOCS 1999*, pages 543–553. IEEE Computer Society, 1999.
54. A. Sahai and B. Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *Proc. of STOC 2014*, pages 475–484. ACM, 2014.
55. A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai. Robust non-interactive zero-knowledge. In *Proc. of CRYPTO 2001*, volume 2139 of *LNCS*, pages 566–598. Springer, 2001.
56. H. Wee. Efficient chosen-ciphertext security via extractable hash proofs. In *Proc. of CRYPTO 2010*, volume 6223 of *LNCS*, pages 314–332. Springer, 2010.

A Standard Cryptographic Primitives

A.1 Public Key Encryption

A public key encryption (PKE) scheme Π consists of the three PPTAs $(\mathsf{PKG}, \mathsf{Enc}, \mathsf{Dec})$ with the following interface:

Key Generation:	**Encryption:**	**Decryption:**
$(pk, sk) \leftarrow \mathsf{PKG}(1^k)$	$c \leftarrow \mathsf{Enc}(pk, m)$	m (or $\perp$) $\leftarrow \mathsf{Dec}(sk, c)$

where Dec is a deterministic algorithm, (pk, sk) is a public/secret key pair, and c is a ciphertext of a plaintext m under pk. We say that a PKE scheme satisfies *correctness* if for all $k \in \mathbb{N}$, all keys (pk, sk) output from $\mathsf{PKG}(1^k)$, and all plaintexts m, it holds that $\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) = m$.

Since we do not directly use the ordinary security notions for PKE in this paper, we do not introduce them. In Section 2.2, we review the (simplified version of) trapdoor simulatability property [15] of a PKE scheme.

A.2 Key Encapsulation Mechanisms

A key encapsulation mechanism (KEM) Γ consists of the three PPTAs $(\mathsf{KKG}, \mathsf{Encap}, \mathsf{Decap})$ with the following interface:

$$\begin{array}{ccc} \underline{\textbf{Key Generation:}} & \underline{\textbf{Encapsulation:}} & \underline{\textbf{Decapsulation:}} \\ (pk, sk) \leftarrow \mathsf{KKG}(1^k) & (c, K) \leftarrow \mathsf{Encap}(pk) & K \text{ (or } \bot) \leftarrow \mathsf{Decap}(sk, c) \end{array}$$

where Decap is a deterministic algorithm, (pk, sk) is a public/secret key pair that defines a session-key space $\mathcal{K}$, and c is a ciphertext of a session-key $K \in \mathcal{K}$ under pk. We say that a KEM satisfies *correctness* if for all $k \in \mathbb{N}$, all keys (pk, sk) output from $\mathsf{KKG}(1^k)$ and all ciphertext/session-key pairs (c, K) output from $\mathsf{Encap}(pk)$, it holds that $\mathsf{Decap}(sk, c) = K$.

$\mathtt{CPA}/1\text{-}\mathtt{CCA}/\mathtt{CCA}$ *Security.* For a KEM $\Gamma = (\mathsf{KKG}, \mathsf{Encap}, \mathsf{Decap})$ and an adversary $\mathcal{A}$, we define the $\mathtt{CCA}$ experiment $\mathsf{Expt}^{\mathtt{CCA}}_{\Gamma,\mathcal{A}}(k)$ as follows:

$$\begin{aligned} \mathsf{Expt}^{\mathtt{CCA}}_{\Gamma,\mathcal{A}}(k) : [\ (pk, sk) \leftarrow \mathsf{KKG}(1^k);\ (c^*, K_1^*) \leftarrow \mathsf{Encap}(pk);\ K_0^* \leftarrow \{0,1\}^k;\ b \leftarrow \{0,1\} \\ b' \leftarrow \mathcal{A}^{\mathsf{Decap}(sk,\cdot)}(pk, c^*, K_b^*);\ \text{Return } (b' \overset{?}{=} b)\ ], \end{aligned}$$

in the experiment, $\mathcal{A}$ is not allowed to submit c^* to the oracle. We define the $1\text{-}\mathtt{CCA}$ experiment $\mathsf{Expt}^{1\text{-}\mathtt{CCA}}_{\Gamma,\mathcal{A}}(k)$ in the same way as the $\mathtt{CCA}$ experiment, except that $\mathcal{A}$ is allowed to submit a query $c \neq c^*$ only once. Furthermore, we define the $\mathtt{CPA}$ experiment $\mathsf{Expt}^{\mathtt{CPA}}_{\Gamma,\mathcal{A}}(k)$ is also defined similarly to the $\mathtt{CCA}$ experiment, except that $\mathcal{A}$ is not allowed to submit any query.

Definition 4. *Let* $\mathtt{ATK} \in \{\mathtt{CPA}, 1\text{-}\mathtt{CCA}, \mathtt{CCA}\}$*. We say that a KEM* Γ *is* $\mathtt{ATK}$ *secure if for all PPTAs* $\mathcal{A}$*, the advantage* $\mathsf{Adv}^{\mathtt{ATK}}_{\Gamma,\mathcal{A}}(k) := 2 \cdot |\Pr[\mathsf{Expt}^{\mathtt{ATK}}_{\Gamma,\mathcal{A}}(k) = 1] - 1/2|$ *is negligible.*

Smoothness. For a KEM $\Gamma = (\mathsf{KKG}, \mathsf{Encap}, \mathsf{Decap})$, consider the following quantity, called *smoothness* [4] of Γ:

$$\mathsf{Smth}_\Gamma(k) := \mathop{\mathbf{E}}_{(pk,sk) \leftarrow \mathsf{KKG}(1^k)} \Big[\max_{c' \in \{0,1\}^*} \Pr_{(c,K) \leftarrow \mathsf{Encap}(pk)} [c = c'] \Big].$$

Bellare, Hofheinz, and Kiltz [4] showed the following:[7]

Lemma 4. *If a KEM* Γ *is* $\mathtt{CPA}$ *secure, then* $\mathsf{Smth}_\Gamma(k)$ *is negligible.*

[7] Precisely speaking, [4] only showed that if a KEM Γ is $\mathtt{CCA}$ secure, then $\mathsf{Smth}_\Gamma(k)$ is negligible. However, it is easy to see that their proof carries over to the $\mathtt{CPA}$ case.

A.3 Commitment

Here we review the definition of a commitment scheme. We only define a non-interactive commitment scheme that has a setup procedure, which is sufficient for our purpose in this paper.

Formally, a commitment scheme $\mathcal{C}$ consists of the following two PPTAs $(\mathsf{CKG}, \mathsf{Com})$ with the following interface:

Key Generation:	**Commitment Generation:**
$ck \leftarrow \mathsf{CKG}(1^k)$	$c \leftarrow \mathsf{Com}(ck, m)$

where ck is a commitment key, and c is a commitment of the message m under ck.

As a (non-standard) requirement, we require the size of a commitment to be k-bit for k-bit security, no matter how long a committed message is.[8] For the binding property, we require a slightly weaker variant than the ordinary notion, called *target-binding*, which was also used in [40].

Definition 5. *We say that a commitment scheme $\mathcal{C}$ is* target-binding[9] *if for all PPTAs $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the advantage $\mathsf{Adv}^{\mathtt{TBind}}_{\mathcal{C},\mathcal{A}}(k) := \Pr[\mathsf{Expt}^{\mathtt{TBind}}_{\mathcal{C},\mathcal{A}}(k) = 1]$ is negligible, where the experiment $\mathsf{Expt}^{\mathtt{TBind}}_{\mathcal{C},\mathcal{A}}(k)$ is defined as follows:*

$$\mathsf{Expt}^{\mathtt{TBind}}_{\mathcal{C},\mathcal{A}}(k) : [\ (m, r, \mathsf{st}) \leftarrow \mathcal{A}_1(1^k);\ ck \leftarrow \mathsf{CKG}(1^k);\ (m', r') \leftarrow \mathcal{A}_2(\mathsf{st}, ck);$$
$$\text{Return 1 iff } \mathsf{Com}(ck, m'; r') = \mathsf{Com}(ck, m; r) \wedge m' \neq m.\ ].$$

Since we do not directly use the hiding property, we do not introduce its formal definition. In Section 2.3, we define the trapdoor simulatability property for a commitment scheme, which is defined in essentially the same way as a TSPKE scheme.

A.4 Universal One-Way Hash Functions

Here, we recall the definition of a universal one-way hash function (UOWHF) [46].

Definition 6. *We say that a pair of PPTAs $\mathcal{H} = (\mathsf{HKG}, \mathsf{H})$ is a universal one-way hash function (UOWHF) if the following two properties are satisfied:*

(Syntax) *On input 1^k, HKG outputs a hash-key κ. For any hash-key κ output from $\mathsf{HKG}(1^k)$, H defines an (efficiently computable) function of the form $\mathsf{H}_\kappa : \{0,1\}^* \to \{0,1\}^k$.*

(Universal One-wayness) *For all PPTAs $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the advantage $\mathsf{Adv}^{\mathtt{UOW}}_{\mathcal{H},\mathcal{A}}(k) := \Pr[\mathsf{Expt}^{\mathtt{UOW}}_{\mathcal{H},\mathcal{A}}(k) = 1]$ is negligible, where the experiment $\mathsf{Expt}^{\mathtt{UOW}}_{\mathcal{H},\mathcal{A}}(k)$ is defined as follows:*

$$\mathsf{Expt}^{\mathtt{UOW}}_{\mathcal{H},\mathcal{A}}(k) : [\ (m, \mathsf{st}) \leftarrow \mathcal{A}_1(1^k);\ \kappa \leftarrow \mathsf{HKG}(1^k);\ m' \leftarrow \mathcal{A}_2(\mathsf{st}, \kappa);$$
$$\text{Return 1 iff } \mathsf{H}_\kappa(m') = \mathsf{H}_\kappa(m) \wedge m' \neq m\ ].$$

We can construct a UOWHF from any one-way function [46, 51], and thus for example it exists if a $\mathtt{CPA}$ secure PKE scheme exists.

[8] This requirement (together with the following binding property and the "trapdoor simulatability property") can be easily realized if we are given a TSPKE scheme. For more details, see Appendix B.

[9] Note that the target-binding property is slightly weaker than the ordinary binding notion in the sense that an adversary has to choose its first message before seeing a key ck. The relation between the ordinary binding and target-binding is similar to the relation between collision resistance and target collision resistance of a hash function family. The target-binding was also used in [40].

A.5 Signature

A signature scheme Σ consists of the three PPTAs $(\mathsf{SKG}, \mathsf{Sign}, \mathsf{SVer})$ with the following interface:

Key Generation:	**Signing:**	**Verification:**
$(vk, sigk) \leftarrow \mathsf{SKG}(1^k)$	$\sigma \leftarrow \mathsf{Sign}(sigk, m)$	$\top$ or $\bot \leftarrow \mathsf{SVer}(sk, c)$

where SVer is a deterministic algorithm, $(vk, sigk)$ is a verification/signing key pair, and σ is a signature on a message m under vk. $\top$ (resp. $\bot$) is the symbol indicating that σ is a valid (resp. invalid) signature. We say that a signature scheme satisfies *correctness* if for all $k \in \mathbb{N}$, all keys $(vk, sigk)$ output from $\mathsf{PKG}(1^k)$, and all messages m, it holds that $\mathsf{SVer}(vk, m, \mathsf{Sign}(sigk, m)) = \top$.

Strong One-time Unforgeability. We say that a signature scheme $\Sigma = (\mathsf{SKG}, \mathsf{Sign}, \mathsf{SVer})$ is strongly unforgeable under one-time chosen message attacks ($\mathtt{SOT}$ secure, for short), if for all PPTAs $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the advantage $\mathsf{Adv}^{\mathtt{SOT}}_{\Sigma,\mathcal{A}}(k) := \Pr[\mathsf{Expt}^{\mathtt{SOT}}_{\Sigma,\mathcal{A}}(k) = 1]$ is negligible, where the experiment $\mathsf{Expt}^{\mathtt{SOT}}_{\Sigma,\mathcal{A}}(k)$ is defined as follows:

$$\mathsf{Expt}^{\mathtt{SOT}}_{\Sigma,\mathcal{A}}(k) : [\ (vk, sigk) \leftarrow \mathsf{SKG}(1^k);\ (m, \mathsf{st}) \leftarrow \mathcal{A}_1(vk);\ \sigma \leftarrow \mathsf{Sign}(sigk, m);\ (m', \sigma') \leftarrow \mathcal{A}_2(\mathsf{st}, \sigma);$$
$$\text{Return 1 iff } \mathsf{SVer}(vk, m', \sigma') = \top \wedge (m', \sigma') \neq (m, \sigma).\ ].$$

We can construct a $\mathtt{SOT}$ secure signature scheme from any one-way function [46, 51], and thus for example it exists if a $\mathtt{CPA}$ secure PKE scheme exists.

B Concrete Constructions of Trapdoor Simulatable Commitment Schemes

Here, we show that we can construct a commitment scheme satisfying target-binding, trapdoor simulatability, and the requirement of size of commitments (k-bit for k-bit security) from a TSPKE scheme and a UOWHF. Since a UOWHF can be constructed from any one-way function [46, 51] which is in turn implied by a TSPKE scheme, we can achieve a commitment scheme satisfying all our requirements only from a TSPKE scheme.

Specifically, let $\Pi = (\mathsf{PKG}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{oSamp}_{\Pi}, \mathsf{rSamp}_{\Pi})$ be a TSPKE scheme, and $\mathcal{H} = (\mathsf{HKG}, \mathsf{H})$ be a UOWHF. Then we construct a trapdoor simulatable commitment scheme $\mathcal{C} = (\mathsf{CKG}, \mathsf{Com}, \mathsf{oSamp}_{\mathcal{C}}, \mathsf{rSamp}_{\mathcal{C}})$ as in Fig. 5. It is straightforward to see that the trapdoor simulatability property of the commitment scheme $\mathcal{C}$ follows from that of the underlying TSPKE scheme Π, because a commitment is just a hash value of an encryption of a message. (In the randomness $\widehat{r}_c$ used by $\mathsf{oSamp}_{\mathcal{C}}$, the randomness for HKG is also included.) Furthermore, the target-binding property of $\mathcal{C}$ follows from the security of the UOWHF $\mathcal{H}$. Specifically, recall that a PKE scheme can be considered as a perfectly binding commitment scheme (because there are no two distinct messages whose encryptions collide due to the correctness). Therefore, the security of the underlying UOWHF guarantees that a "target" collision pair of ciphertexts, and hence (m, m'), is hard to find.

C Postponed Proofs

C.1 Proof of Lemma 1: A Concrete TSPTBE Scheme

Let $\Pi = (\mathsf{PKG}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{oSamp}_{\Pi}, \mathsf{rSamp}_{\Pi})$ be a TSPKE scheme. Then, we construct a TSPTBE scheme $\mathcal{T} = (\mathsf{TKG}, \mathsf{TEnc}, \mathsf{TDec}, \mathsf{Punc}, \widehat{\mathsf{TDecoSamp}}_{\mathcal{T}}, \mathsf{rSamp}_{\mathcal{T}})$ as in Fig. 6.

$\mathsf{CKG}(1^k; r'_g = (r_h, r_g))$:	$\mathsf{Com}(ck, m; r_c)$:	$\mathsf{oSamp}_{\mathcal{C}}(1^k; \widehat{r}_c = (r_h, \widehat{r}_e))$:	$\mathsf{rSamp}_{\mathcal{C}}(r'_g = (r_h, r_g), r_c, m)$:
$\kappa \leftarrow \mathsf{HKG}(1^k; r_h)$ $(pk, sk) \leftarrow \mathsf{PKG}(1^k; r_g)$ Return $ck \leftarrow (\kappa, pk)$.	$(\kappa, pk) \leftarrow ck$ $c \leftarrow \mathsf{Enc}(pk, m; r_c)$ Return $c' \leftarrow \mathsf{H}_\kappa(c)$.	$\kappa \leftarrow \mathsf{HKG}(1^k; r_h)$ $(pk, c) \leftarrow \mathsf{oSamp}_{\Pi}(1^k; \widehat{r}_e)$ $ck \leftarrow (\kappa, pk)$ $c' \leftarrow \mathsf{H}_\kappa(c)$ Return (ck, c').	$\widehat{r}_e \leftarrow \mathsf{rSamp}_{\Pi}(r_g, r_c, m)$ $\widehat{r}_c \leftarrow (r_h, \widehat{r}_e)$ Return $\widehat{r}_c$.

Fig. 5. A concrete example of a trapdoor simulatable commitment scheme $\mathcal{C}$ based on a TSPKE scheme Π and a UOWHF $\mathcal{H}$.

In the punctured decryption algorithm $\widehat{\mathsf{TDec}}$, the index $\ell \in [k]$ computed at the sixth step can always be found because $\mathsf{tag} \neq \mathsf{tag}^*$ is guaranteed at the point. Under this index $\ell \in [k]$, it holds that $t_\ell = 1 - t^*_\ell$. Since each component c_i in a ciphertext encrypts the same plaintext, the correctness of $\mathcal{T}$ is straightforward to see.

Now, we show that for any PPTA adversary $\mathcal{A}$ that attacks the trapdoor simulatability property of the PTBE scheme $\mathcal{T}$ (in the sense of Definition 3), there exists a PPTA adversary $\mathcal{B}_{\mathsf{p}}$ that attacks the trapdoor simulatability property of the underlying PKE scheme Π such that

$$\mathsf{Adv}^{\mathtt{TSPTBE}}_{\mathcal{T},\mathcal{A}}(k) = k \cdot \mathsf{Adv}^{\mathtt{TSPKE}}_{\Pi,\mathcal{B}_{\mathsf{p}}}(k), \tag{5}$$

which by the trapdoor simulatability property of Π, implies that $\mathcal{T}$ is trapdoor simulatable as well.

Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be any PPTA adversary that attacks the trapdoor simulatability property of $\mathcal{T}$. Consider the following sequence of games:

Game 0: This is the real experiment $\mathsf{Expt}^{\mathtt{TSPTBE\text{-}Real}}_{\mathcal{T},\mathcal{A}}(k)$. To define the subsequent games, we change the ordering of the steps in such a way that the view of $\mathcal{A}$ is not changed at all, as follows:

```
Game 0:
  $(\mathsf{tag}^*, m, \mathsf{st}) \leftarrow \mathcal{A}_1(1^k)$;
  View $\mathsf{tag}^*$ as $(t^*_1, \dots, t^*_k) \in \{0,1\}^k$;
  $\forall i \in [k]$ :
      $r'_{g,i}, r_{e,i} \leftarrow \{0,1\}^*$;
      $(pk_i^{(t^*_i)}, sk_i^{(t^*_i)}) \leftarrow \mathsf{PKG}(1^k; r'_{g,i})$;
      $c_i \leftarrow \mathsf{Enc}(pk_i^{(t^*_i)}, m; r_{e,i})$;
      $\widehat{r}_i \leftarrow \mathsf{rSamp}_{\Pi}(r'_{g,i}, r_{e,i}, m)$;
  (Continue to the right column ↗)
$\forall i \in [k]$ :
    $r_{g,i} \leftarrow \{0,1\}^*$;
    $(pk_i^{(1-t^*_i)}, sk_i^{(1-t^*_i)}) \leftarrow \mathsf{PKG}(1^k; r_{g,i})$;
$pk \leftarrow (pk_i^{(j)})_{i \in [k], j \in \{0,1\}}$;
$c \leftarrow (c_i)_{i \in [k]}$;
$\widehat{sk}_{\mathsf{tag}^*} \leftarrow (\mathsf{tag}^*, (sk_i^{(1-t^*_i)})_{i \in [k]})$;
$\widehat{R} \leftarrow ((\widehat{r}_i)_{i \in [k]}, (r_{g,i})_{i \in [k]})$;
$b' \leftarrow \mathcal{A}(pk, c, \widehat{sk}_{\mathsf{tag}^*}, \widehat{R})$
```

Game n: (where $n \in [k]$) In this game, the first-to-n-th public key/ciphertext pair $(pk_i^{(t^*_i)}, c_i)_{i \in [n]}$ are generated by the oblivious sampling algorithm $\mathsf{oSamp}_{\Pi}(1^k; \widehat{r}_i)$ where $\widehat{r}_i$ is a uniformly random value.

More precisely, this game is defined as follows:

```
Game $n$: (where $n \in [k]$)
  $(\mathsf{tag}^*, m, \mathsf{st}) \leftarrow \mathcal{A}_1(1^k)$;
  View $\mathsf{tag}^*$ as $(t^*_1, \dots, t^*_k) \in \{0,1\}^k$;
  $\forall i \in [n]$ :
      $\widehat{r}_i \leftarrow \{0,1\}^*$;
      $(pk_i^{(t^*_i)}, c_i) \leftarrow \mathsf{oSamp}_{\Pi}(1^k; \widehat{r}_i)$;
  $\forall i \in \{n+1, \dots, k\}$ :
      $r'_{g,i}, r_{e,i} \leftarrow \{0,1\}^*$;
      $(pk_i^{(t^*_i)}, sk_i^{(t^*_i)}) \leftarrow \mathsf{PKG}(1^k; r'_{g,i})$;
      $c_i \leftarrow \mathsf{Enc}(pk_i^{(t^*_i)}, m; r_{e,i})$;
      $\widehat{r}_i \leftarrow \mathsf{rSamp}_{\Pi}(r'_{g,i}, r_{e,i}, m)$;
  (Continue to the right column ↗)
$\forall i \in [k]$ :
    $r_{g,i} \leftarrow \{0,1\}^*$;
    $(pk_i^{(1-t^*_i)}, sk_i^{(1-t^*_i)}) \leftarrow \mathsf{PKG}(1^k; r_{g,i})$;
$pk \leftarrow (pk_i^{(j)})_{i \in [k], j \in \{0,1\}}$;
$c \leftarrow (c_i)_{i \in [k]}$;
$\widehat{sk}_{\mathsf{tag}^*} \leftarrow (\mathsf{tag}^*, (sk_i^{(1-t^*_i)})_{i \in [k]})$;
$\widehat{R} \leftarrow ((\widehat{r}_i)_{i \in [k]}, (r_{g,i})_{i \in [k]})$;
$b' \leftarrow \mathcal{A}(pk, c, \widehat{sk}_{\mathsf{tag}^*}, \widehat{R})$
```

```
$\mathsf{TKG}(1^k; R_g)$ :
    $(r_{g,i}^{(j)})_{i\in[k],j\in\{0,1\}} \leftarrow R_g$
    $\forall (i,j) \in [k] \times \{0,1\}$ :
        $(pk_i^{(j)}, sk_i^{(j)}) \leftarrow \mathsf{PKG}(1^k; r_{g,i}^{(j)})$
    $pk \leftarrow (pk_i^{(j)})_{i\in[k],j\in\{0,1\}}$
    $sk \leftarrow (sk_i^{(j)})_{i\in[k],j\in\{0,1\}}$
    Return $(pk, sk)$.
```

```
$\mathsf{TEnc}(pk, \mathsf{tag}, m; R_t)$ :
    $(r_{e,i})_{i\in[k]} \leftarrow R_t$
    $(pk_i^{(j)})_{i\in[k],j\in\{0,1\}} \leftarrow pk$
    View $\mathsf{tag}$ as $(t_1, \dots, t_k) \in \{0,1\}^k$.
    $\forall i \in [k] : c_i \leftarrow \mathsf{Enc}(pk_i^{(t_i)}, m; r_{e,i})$
    Return $c \leftarrow (c_i)_{i\in[k]}$.
```

```
$\mathsf{TDec}(sk, \mathsf{tag}, c)$ :
    $(sk_i^{(j)})_{i\in[k],j\in\{0,1\}} \leftarrow sk$
    $(c_i)_{i\in[k]} \leftarrow c$
    Let $t_1$ be the first bit of $\mathsf{tag}$.
    Return $m \leftarrow \mathsf{Dec}(sk_1^{(t_1)}, c_1)$.
```

```
$\mathsf{Punc}(sk, \mathsf{tag}^*)$
    $(sk_i^{(j)})_{i\in[k],j\in\{0,1\}} \leftarrow sk$
    View $\mathsf{tag}^*$ as $(t_1^*, \dots, t_k^*) \in \{0,1\}^k$.
    $\widehat{sk}_{\mathsf{tag}^*} \leftarrow (\mathsf{tag}^*, (sk_i^{(1-t_i^*)})_{i\in[k]})$
    Return $\widehat{sk}_{\mathsf{tag}^*}$.
```

```
$\widehat{\mathsf{TDec}}(\widehat{sk}_{\mathsf{tag}^*}, \mathsf{tag}, c)$ :
    $(\mathsf{tag}^*, (sk_i^{(1-t_i^*)})_{i\in[k]}) \leftarrow \widehat{sk}_{\mathsf{tag}^*}$
    If $\mathsf{tag} = \mathsf{tag}^*$ then return $\perp$.
    $(c_i)_{i\in[k]} \leftarrow c$
    View $\mathsf{tag}$ as $(t_1, \dots, t_k) \in \{0,1\}^k$.
    View $\mathsf{tag}^*$ as $(t_1^*, \dots, t_k^*) \in \{0,1\}^k$.
    $\ell \leftarrow \min\{i \in [k] \mid t_i \neq t_i^*\}$
    Return $m \leftarrow \mathsf{Dec}(sk_\ell^{(1-t_\ell^*)}, c_\ell)$.
```

```
$\mathsf{oSamp}_{\mathcal{T}}(\mathsf{tag}^*; \widehat{R})$
    $((\widehat{r}_i)_{i\in[k]}, (r_{g,i})_{i\in[k]}) \leftarrow \widehat{R}$
    View $\mathsf{tag}^*$ as $(t_1^*, \dots, t_k^*) \in \{0,1\}^k$.
    $\forall i \in [k]$ :
        $(pk_i^{(t_i^*)}, c_i) \leftarrow \mathsf{oSamp}_{\Pi}(1^k; \widehat{r}_i)$
        $(pk_i^{(1-t_i^*)}, sk_i^{(1-t_i^*)}) \leftarrow \mathsf{PKG}(1^k; r_{g,i})$
    $pk \leftarrow (pk_i^{(j)})_{i\in[k],j\in\{0,1\}}$
    $c \leftarrow (c_i)_{i\in[k]}$
    $\widehat{sk}_{\mathsf{tag}^*} \leftarrow (\mathsf{tag}^*, (sk_i^{(1-t_i^*)})_{i\in[k]})$
    Return $(pk, c, \widehat{sk}_{\mathsf{tag}^*})$.
```

```
$\mathsf{rSamp}_{\mathcal{T}}(R_g, R_t, \mathsf{tag}^*, m)$ :
    $(r_{g,i}^{(j)})_{i\in[k],j\in\{0,1\}} \leftarrow R_g$
    $(r_{e,i})_{i\in[k]} \leftarrow R_t$
    View $\mathsf{tag}^*$ as $(t_1^*, \dots, t_k^*) \in \{0,1\}^k$.
    $\forall i \in [k]$ :
        $\widehat{r}_i \leftarrow \mathsf{rSamp}_{\Pi}(r_i^{(t_i^*)}, r_{e,i}, m)$
        $r_{g,i} \leftarrow r_{g,i}^{(1-t_i^*)}$
    $\widehat{R} \leftarrow ((\widehat{r}_i)_{i\in[k]}, (r_{g,i})_{i\in[k]})$
    Return $\widehat{R}$.
```

Fig. 6. A concrete instantiation of a TSPTBE scheme $\mathcal{T}$ based on a TSPKE scheme Π.

Note that in Game k, every public key/ciphertext pair $(pk_i^{(t_i^*)}, c_i)$ is generated by the oblivious sampling algorithm oSamp_{Π}, thus Game k is exactly the simulated experiment $\mathsf{Expt}_{\mathcal{T},\mathcal{A}}^{\mathtt{TSPTBE\text{-}Sim}}(k)$.

For each $i \in \{0, \dots, k\}$, let X_i be the event that in Game i, $\mathcal{A}_2$ outputs 1 (i.e. $b' = 1$ occurs). Then, by definition of the games and events, $\mathcal{A}$'s advantage in attacking the trapdoor simulatability property can be calculated as follows:

$$\begin{aligned}\mathsf{Adv}_{\mathcal{T},\mathcal{A}}^{\mathtt{TSPTBE}}(k) &= \Big|\Pr[\mathsf{Expt}_{\mathcal{T},\mathcal{A}}^{\mathtt{TSPTBE\text{-}Real}}(k) = 1] - \Pr[\mathsf{Expt}_{\mathcal{T},\mathcal{A}}^{\mathtt{TSPTBE\text{-}Sim}}(k) = 1]\Big| \\ &= \Big|\Pr[\mathsf{X}_0] - \Pr[\mathsf{X}_k]\Big| \\ &= \Big|\sum_{n\in[k]} (\Pr[\mathsf{X}_{n-1}] - \Pr[\mathsf{X}_n])\Big|. \end{aligned} \tag{6}$$

Now, consider the following PPTA adversary $\mathcal{B}_{\mathsf{p}} = (\mathcal{B}_{\mathsf{p}1}, \mathcal{B}_{\mathsf{p}2})$ that, using $\mathcal{A}$ as a building block, attacks the trapdoor simulatability property of the underlying PKE scheme Π:

$\mathcal{B}_{\mathsf{p}1}(1^k)$: $\mathcal{B}_{\mathsf{p}1}$ first runs $(\mathsf{tag}^*, m, \mathsf{st}) \leftarrow \mathcal{A}_1(1^k)$, sets $\mathsf{st}_{\mathcal{B}} \leftarrow$ ($\mathcal{B}_{\mathsf{p}1}$'s entire view), and terminates with output $(m, \mathsf{st}_{\mathcal{B}})$.

$\mathcal{B}_{\mathsf{p}2}(\mathsf{st}_{\mathcal{B}}, pk', c', \widehat{r}')$: $\mathcal{B}_{\mathsf{p}2}$ first picks $u \in [k]$ uniformly at random. Now, if $u \geq 2$, then for each $i \in [u-1]$, $\mathcal{B}_{\mathsf{p}2}$ picks $\widehat{r}_i \in \{0,1\}^*$ uniformly at random, and runs $(pk_i, c_i) \leftarrow \mathsf{oSamp}_{\Pi}(1^k; \widehat{r}_i)$. $\mathcal{B}_{\mathsf{p}2}$

sets $pk_u^{(t_u^*)} \leftarrow pk'$, $c_u \leftarrow c'$, and $\widehat{r}_u \leftarrow \widehat{r}'$. Furthermore, if $u \le k-1$, then for each $i \in \{u+1, \dots, k\}$, $\mathcal{B}_{\mathsf{p}2}$ picks $r'_{g,i}, r_{e,i} \in \{0,1\}^*$ uniformly at random, and runs $(pk_i^{(t_i^*)}, sk_i^{(t_i^*)}) \leftarrow \mathsf{PKG}(1^k; r'_{g,i})$, $c_i \leftarrow \mathsf{Enc}(pk_i^{(t_i^*)}, m; r_{e,i})$, and $\widehat{r}_i \leftarrow \mathsf{rSamp}_{\Pi}(r'_{g,i}, r_{e,i}, m)$. Next, for $i \in [k]$, $\mathcal{B}_{\mathsf{p}2}$ picks $r_{g,i} \in \{0,1\}^*$ uniformly at random, and runs $(pk_i^{(1-t_i^*)}, sk_i^{(1-t_i^*)}) \leftarrow \mathsf{PKG}(1^k; r_{g,i})$. Then, $\mathcal{B}_{\mathsf{p}2}$ sets $pk \leftarrow (pk_i^{(j)})_{i\in[k], j\in\{0,1\}}$, $c \leftarrow (c_i)_{i\in[k]}$, $\widehat{sk}_{\mathsf{tag}^*} \leftarrow (\mathsf{tag}^*, (sk_i^{(1-t_i^*)})_{i\in[k]})$, and $\widehat{R} \leftarrow ((\widehat{r}_i)_{i\in[k]}, (r_{g,i})_{i\in[k]})$. Finally, $\mathcal{B}_{\mathsf{p}2}$ runs $b' \leftarrow \mathcal{A}_2(\mathsf{st}, pk, c, \widehat{sk}_{\mathsf{tag}^*}, \widehat{R})$, and terminates with output b'.

The above completes the description of $\mathcal{B}_{\mathsf{p}}$. For notational convenience, let us write $\mathsf{Expt}^{\mathtt{TS\text{-}Real}}$ and $\mathsf{Expt}^{\mathtt{TS\text{-}Sim}}$ to mean $\mathsf{Expt}^{\mathtt{TSPKE\text{-}Real}}_{\Pi,\mathcal{B}_{\mathsf{p}}}(k)$ and $\mathsf{Expt}^{\mathtt{TSPKE\text{-}Sim}}_{\Pi,\mathcal{B}_{\mathsf{p}}}(k)$, respectively. $\mathcal{B}_{\mathsf{p}}$'s advantage in attacking the trapdoor simulatability property of the underlying PKE scheme Π can be calculated as follows:

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{TSPKE}}_{\Pi,\mathcal{B}_{\mathsf{p}}}(k) &= \Big|\Pr[\mathsf{Expt}^{\mathtt{TSPKE\text{-}Real}}_{\Pi,\mathcal{B}_{\mathsf{p}}}(k) = 1] - \Pr[\mathsf{Expt}^{\mathtt{TSPKE\text{-}Sim}}_{\Pi,\mathcal{B}_{\mathsf{p}}}(k) = 1]\Big| \\
&= \Big|\Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Real}} : b' = 1] - \Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Sim}} : b' = 1]\Big| \\
&= \Big|\sum_{n\in[k]} \Big(\Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Real}} : b' = 1 \wedge u = n] - \Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Sim}} : b' = 1 \wedge u = n]\Big)\Big| \\
&= \Big|\sum_{n\in[k]} \Big(\Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Real}} : b' = 1 | u = n] \cdot \Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Real}} : u = n] \\
&\qquad - \Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Sim}} : b' = 1 | u = n] \cdot \Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Sim}} : u = n]\Big)\Big|. \qquad (7)
\end{aligned}$$

Note that $\mathcal{B}_{\mathsf{p}1}$ chooses the index $u \in [k]$ uniformly at random, independently of $\mathcal{B}_{\mathsf{p}}$'s experiment, and thus for every $n \in [k]$, we have $\Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Real}} : u = n] = \Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Sim}} : u = n] = 1/k$.

Furthermore, for each $n \in [k]$, if $\mathcal{B}_{\mathsf{p}}$ is run in $\mathsf{Expt}^{\mathtt{TS\text{-}Real}}$ and $u = n$ holds, then $\mathcal{B}_{\mathsf{p}}$ generates the first-to-$(n-1)$-th public key/ciphertext pair $(pk_i^{(t_i^*)}, c_i)_{i\in[n-1]}$ obliviously, which is exactly how they are generated in Game $n-1$. All other values are also distributed as those in Game $n-1$, and thus $\mathcal{B}_{\mathsf{p}}$ does a perfect simulation of Game $n-1$ for $\mathcal{A}$. This means that we have $\Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Real}} : b' = 1 | u = n] = \Pr[\mathsf{X}_{n-1}]$. On the other hand, if $\mathcal{B}_{\mathsf{p}}$ is run in $\mathsf{Expt}^{\mathtt{TS\text{-}Sim}}$ and $u = n$ holds, then $\mathcal{B}_{\mathsf{p}}$ generates the first-to-n-th public key/ciphertext pair $(pk_i^{(t_i^*)}, c_i)_{i\in[n]}$ obliviously, which is exactly how they are generated in Game n. Since this is the only change from the above, we have $\Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Sim}} : b' = 1 | u = n] = \Pr[\mathsf{X}_n]$.

Using the above facts in Equation (7) together with Equation (6), it is guaranteed that there exists a PPTA adversary $\mathcal{B}_{\mathsf{p}}$ satisfying Equation (5), as required. □ (**Lemma 1**)

C.2 Proof of Lemma 2: Useful Fact

Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be any PPTA $\mathcal{A}$ that runs in the "real" experiment $\mathsf{Expt}^{\mathtt{TS\text{-}Real}}_{[\mathcal{C},\mathcal{T}],\mathcal{A}}(k)$ or the "simulated" experiment $\mathsf{Expt}^{\mathtt{TS\text{-}Sim}}_{[\mathcal{C},\mathcal{T}],\mathcal{A}}(k)$. We will show that for such $\mathcal{A}$, there exist PPTAs $\mathcal{B}_{\mathsf{c}}$ and $\mathcal{B}_{\mathsf{t}}$ such that

$$\mathsf{Adv}^{\mathsf{TS}}_{[\mathcal{C},\mathcal{T}],\mathcal{A}}(k) \le \mathsf{Adv}^{\mathsf{TSCom}}_{\mathcal{C},\mathcal{B}_{\mathsf{c}}}(k) + \mathsf{Adv}^{\mathsf{TSPTBE}}_{\mathcal{T},\mathcal{B}_{\mathsf{t}}}(k). \qquad (8)$$

The proof is by a simple hybrid argument. Specifically, consider the following sequence of games:

Game 1: This is the experiment $\mathsf{Expt}^{\mathtt{TS\text{-}Real}}_{[\mathcal{C},\mathcal{T}],\mathcal{A}}(k)$.

Game 2: Same as Game 1, except that $\widehat{r}_c$ is chosen uniformly at random, and the key/commitment pair (ck, tag^*) is generated by the oblivious sampling algorithm $\mathsf{oSamp}_{\mathcal{C}}(1^k; \widehat{r}_c)$.

Game 3: This is the experiment $\mathsf{Expt}^{\mathtt{TS\text{-}Sim}}_{[\mathcal{C},\mathcal{T}],\mathcal{A}}(k)$.

For $i \in [3]$, let X_i be the event that $\mathcal{A}_2$ outputs 1 in Game i. Then, by the definition of the event and the triangle inequality, we have

$$\mathsf{Adv}^{\mathtt{TS}}_{[\mathcal{C},\mathcal{T}],\mathcal{A}}(k) = \Big|\Pr[\mathsf{X}_1] - \Pr[\mathsf{X}_3]\Big| \le \sum_{i\in[2]} \Big|\Pr[\mathsf{X}_i] - \Pr[\mathsf{X}_{i+1}]\Big|. \tag{9}$$

We will upperbound each term in the right hand side of the above inequality.

Claim 6 *There exists a PPTA* $\mathcal{B}_{\mathsf{c}}$ *such that* $\mathsf{Adv}^{\mathtt{TSCom}}_{\mathcal{C},\mathcal{B}_{\mathsf{c}}}(k) = |\Pr[\mathsf{X}_1] - \Pr[\mathsf{X}_2]|$.

Proof of Claim 6. Consider the following PPTA adversary $\mathcal{B}_{\mathsf{c}} = (\mathcal{B}_{\mathsf{c}1}, \mathcal{B}_{\mathsf{c}2})$ against the trapdoor simulatability of $\mathcal{C}$:

$\mathcal{B}_{\mathsf{c}1}(1^k)$: $\mathcal{B}_{\mathsf{c}1}$ first runs $(m, \mathsf{st}) \leftarrow \mathcal{A}_1(1^k)$, sets $\mathsf{st}_{\mathcal{B}} \leftarrow$ ($\mathcal{B}_{\mathsf{c}1}$'s entire view), and finally terminates with output $(m, \mathsf{st}_{\mathcal{B}})$.

$\mathcal{B}_{\mathsf{c}2}(\mathsf{st}_{\mathcal{B}}, ck, \mathsf{tag}^*, \widehat{r}_c)$: $\mathcal{B}_{\mathsf{c}2}$ picks $r'_g, r_t \in \{0,1\}^*$ uniformly at random, then runs $(pk, sk) \leftarrow \mathsf{TKG}(1^k; r'_g)$, $c^* \leftarrow \mathsf{TEnc}(pk, \mathsf{tag}^*, m; r_t)$, $\widehat{sk}_{\mathsf{tag}^*} \leftarrow \mathsf{Punc}(sk, \mathsf{tag}^*)$, and $\widehat{r}_t \leftarrow \mathsf{rSamp}_{\mathcal{T}}(r'_g, r_t, \mathsf{tag}^*, m)$. Then $\mathcal{B}_{\mathsf{c}2}$ runs $b' \leftarrow \mathcal{A}_2(\mathsf{st}, ck, \mathsf{tag}^*, pk, c^*, \widehat{sk}_{\mathsf{tag}^*}, \widehat{r}_c, \widehat{r}_t)$, and terminates with output b'.

The above completes the description of $\mathcal{B}_{\mathsf{c}}$.

It is easy to see that $\mathcal{B}_{\mathsf{c}}$ perfectly simulates Game 1 for $\mathcal{A}$ if $\mathcal{B}_{\mathsf{c}}$ is run in $\mathsf{Expt}^{\mathtt{TSCom\text{-}Real}}_{\mathcal{C},\mathcal{B}_{\mathsf{c}}}(k)$, and perfectly simulates Game 2 if $\mathcal{B}_{\mathsf{c}}$ is run in $\mathsf{Expt}^{\mathtt{TSCom\text{-}Sim}}_{\mathcal{C},\mathcal{B}_{\mathsf{c}}}(k)$. Specifically, in the former case, ck and tag^* are honestly generated by CKG and Com, and $\widehat{r}_c$ is generated by the inverting algorithm $\mathsf{rSamp}_{\mathcal{C}}$. Since $\mathcal{B}_{\mathsf{c}}$ uses $\mathcal{A}_2$'s output as it is, we have $\Pr[\mathsf{Expt}^{\mathtt{TSCom\text{-}Real}}_{\mathcal{C},\mathcal{B}_{\mathsf{c}}}(k) = 1] = \Pr[\mathsf{X}_1]$. On the other hand, in the latter case, $\widehat{r}_c$ is now uniformly random and ck and tag^* are generated by the oblivious sampling algorithm $\mathsf{oSamp}_{\mathcal{C}}$. With a similar argument to the above, we have $\Pr[\mathsf{Expt}^{\mathtt{TSCom\text{-}Sim}}_{\mathcal{C},\mathcal{B}_{\mathsf{c}}}(k) = 1] = \Pr[\mathsf{X}_2]$.

Using these, $\mathcal{B}_{\mathsf{c}}$'s advantage in attacking the trapdoor simulatability property of the commitment scheme $\mathcal{C}$ is calculated as follows:

$$\begin{aligned}\mathsf{Adv}^{\mathtt{TSCom}}_{\mathcal{C},\mathcal{B}_{\mathsf{c}}}(k) &= \Big|\Pr[\mathsf{Expt}^{\mathtt{TSCom\text{-}Real}}_{\mathcal{C},\mathcal{B}_{\mathsf{c}}}(k) = 1] - \Pr[\mathsf{Expt}^{\mathtt{TSCom\text{-}Sim}}_{\mathcal{C},\mathcal{B}_{\mathsf{c}}}(k) = 1]\Big| \\ &= \Big|\Pr[\mathsf{X}_1] - \Pr[\mathsf{X}_2]\Big|,\end{aligned}$$

as required. □ (**Claim 6**)

Claim 7 *There exists a PPTA* $\mathcal{B}_{\mathsf{t}}$ *such that* $\mathsf{Adv}^{\mathtt{TSPTBE}}_{\mathcal{T},\mathcal{B}_{\mathsf{t}}}(k) = |\Pr[\mathsf{X}_2] - \Pr[\mathsf{X}_3]|$.

Proof of Claim 7. Consider the following PPTA adversary $\mathcal{B}_{\mathsf{t}} = (\mathcal{B}_{\mathsf{t}1}, \mathcal{B}_{\mathsf{t}2})$ against the trapdoor simulatability property of $\mathcal{T}$:

$\mathcal{B}_{\mathsf{t}1}(1^k)$: $\mathcal{B}_{\mathsf{t}1}$ first picks $\widehat{r}_c \in \{0,1\}^*$ uniformly at random, and runs $(ck, \mathsf{tag}^*) \leftarrow \mathsf{oSamp}_{\mathcal{C}}(1^k; \widehat{r}_c)$ and $(m, \mathsf{st}) \leftarrow \mathcal{A}_1(1^k)$. Then $\mathcal{B}_{\mathsf{t}1}$ sets $\mathsf{st}_{\mathcal{B}} \leftarrow$ ($\mathcal{B}_{\mathsf{t}1}$'s entire view), and terminates with output $(\mathsf{tag}^*, m, \mathsf{st}_{\mathcal{B}})$.

$\mathcal{B}_{\mathsf{t}2}(\mathsf{st}_{\mathcal{B}}, pk, c^*, \widehat{sk}_{\mathsf{tag}^*}, \widehat{r}_t)$: $\mathcal{B}_{\mathsf{t}2}$ runs $b' \leftarrow \mathcal{A}_2(\mathsf{st}, ck, \mathsf{tag}^*, pk, c^*, \widehat{sk}_{\mathsf{tag}^*}, \widehat{r}_c, \widehat{r}_t)$, and terminates with output b'.

The above completes the description of $\mathcal{B}_{\mathsf{t}}$.

It is easy to see that $\mathcal{B}_{\mathsf{t}}$ perfectly simulates Game 2 for $\mathcal{A}$ if $\mathcal{B}_{\mathsf{t}}$ is run in $\mathsf{Expt}^{\mathtt{TSPTBE\text{-}Real}}_{\mathcal{T},\mathcal{B}_{\mathsf{t}}}(k)$, and perfectly simulates Game 3 if $\mathcal{B}_{\mathsf{t}}$ is run in $\mathsf{Expt}^{\mathtt{TSPTBE\text{-}Sim}}_{\mathcal{T},\mathcal{B}_{\mathsf{t}}}(k)$. Specifically, in the former case, pk, c^*, and $\widehat{sk}_{\mathsf{tag}^*}$ are honestly generated by TKG, TEnc, and Punc, respectively, and $\widehat{r}_t$ is generated by the inverting algorithm $\mathsf{rSamp}_{\mathcal{T}}$. Since $\mathcal{B}_{\mathsf{t}}$ uses $\mathcal{A}_2$'s output as it is, we have $\Pr[\mathsf{Expt}^{\mathtt{TSPTBE\text{-}Real}}_{\mathcal{T},\mathcal{B}_{\mathsf{t}}}(k) = 1] = \Pr[\mathsf{X}_2]$. On the other hand, in the latter case, $\widehat{r}_t$ is now uniformly random, and pk, c, and $\widehat{sk}_{\mathsf{tag}^*}$ are generated by the oblivious sampling algorithm $\mathsf{oSamp}_{\mathcal{T}}$ (using tag^* as input). With a similar argument to the above, we have $\Pr[\mathsf{Expt}^{\mathtt{TSPTBE\text{-}Sim}}_{\mathcal{T},\mathcal{B}_{\mathsf{t}}}(k) = 1] = \Pr[\mathsf{X}_3]$.

Using these, $\mathcal{B}_{\mathsf{t}}$'s advantage in attacking the trapdoor simulatability property of the PTBE scheme $\mathcal{T}$ is calculated as follows:

$$\begin{aligned}\mathsf{Adv}^{\mathtt{TSPTBE}}_{\mathcal{T},\mathcal{B}_{\mathsf{t}}}(k) &= \Big|\Pr[\mathsf{Expt}^{\mathtt{TSPTBE\text{-}Real}}_{\mathcal{T},\mathcal{B}_{\mathsf{t}}}(k) = 1] - \Pr[\mathsf{Expt}^{\mathtt{TSPTBE\text{-}Sim}}_{\mathcal{T},\mathcal{B}_{\mathsf{t}}}(k) = 1]\Big|\\ &= \Big|\Pr[\mathsf{X}_2] - \Pr[\mathsf{X}_3]\Big|,\end{aligned}$$

as required. □ (**Claim 7**)

Claims 6 and 7, and Equation (9) guarantee that there exist PPTAs $\mathcal{B}_{\mathsf{c}}$ and $\mathcal{B}_{\mathsf{t}}$ satisfying Equation (8), as required. □ (**Lemma 2**)

C.3 Proof of Theorem 2: The CCA Security of the Second Construction $\overline{\Gamma}$

The structure of the proof is very similar to the proof of Theorem 1. Thus, we recommend the reader first read it.

As in the case of the first construction, We first introduce the alternative decapsulation algorithm for $\overline{\Gamma}$, and then proceed to its security proof.

Alternative Decapsulation Algorithm. Like our first construction Γ, we can similarly define the alternative decapsulation algorithm $\overline{\mathsf{AltDecap}}$ for $\overline{\Gamma}$. For a k-bit string $\mathsf{tag}^* \in \{0,1\}^k$ and a key pair (PK, SK) output by $\overline{\mathsf{KKG}}(1^k)$, where $PK = (pk_{\mathtt{in}}, pk, ck)$ and $SK = (sk_{\mathtt{in}}, sk, PK)$, we define an "alternative" secret key $\widehat{SK}_{\mathsf{tag}^*}$ associated with $\mathsf{tag}^* \in \{0,1\}^k$ by $\widehat{SK}_{\mathsf{tag}^*} = (sk_{\mathtt{in}}, \mathsf{tag}^*, \widehat{sk}_{\mathsf{tag}^*}, PK)$, where $\widehat{sk}_{\mathsf{tag}^*} = \mathsf{Punc}(sk, \mathsf{tag}^*)$. $\overline{\mathsf{AltDecap}}$ takes an "alternative" secret key $\widehat{SK}_{\mathsf{tag}^*}$ defined as above and a ciphertext $C = (\mathsf{tag}, c)$ as input, and runs as follows:

$\overline{\mathsf{AltDecap}}(\widehat{SK}_{\mathsf{tag}^*}, C)$**:** First check if $\mathsf{tag}^* = \mathsf{tag}$, and return $\perp$ if this is the case. Otherwise, run in exactly the same way as $\overline{\mathsf{Decap}}(SK, C)$, except that "$c_{\mathtt{in}} \leftarrow \widehat{\mathsf{TDec}}(\widehat{sk}_{\mathsf{tag}^*}, \mathsf{tag}, c)$" is executed in the fourth step, instead of "$c_{\mathtt{in}} \leftarrow \mathsf{TDec}(sk, \mathsf{tag}, c)$."

As in the case of the first construction, we can show the following lemma. Since the proof is essentially the same as the proof of Lemma 3, we omit the proof.

Lemma 5. *Let* $\mathsf{tag}^* \in \{0,1\}^k$ *be a string and let* (PK, SK) *be a key pair output by* $\overline{\mathsf{KKG}}(1^k)$*. Furthermore, let* $\widehat{SK}_{\mathsf{tag}^*}$ *be an alternative secret key as defined above. Then, for any ciphertext* $C = (\mathsf{tag}, c)$ *(which could be outside the range of* $\overline{\mathsf{Encap}}(PK)$*) satisfying* $\mathsf{tag} \neq \mathsf{tag}^*$*, it holds that* $\overline{\mathsf{Decap}}(SK, C) = \overline{\mathsf{AltDecap}}(\widehat{SK}_{\mathsf{tag}^*}, C)$*.*

Proof of Theorem 2. Let $\mathcal{A}$ be any PPTA adversary that attacks the KEM $\overline{\Gamma}$ in the sense of CCA security, and makes in total Q decapsulation queries. (Since $\mathcal{A}$ is a PPTA, Q is some polynomial.) As in the proof of Theorem 1, the security proof is done via the sequence of games argument. To describe the games, we will need an extractor $\mathcal{E}$ corresponding to the following ciphertext creator $\mathcal{A}'$ that is guaranteed to exist by the $\mathtt{sPA1}_1$ security of $\Gamma_{\mathtt{in}}$. Specifically, consider the following algorithm $\mathcal{A}'$ (that internally runs $\mathcal{A}$) that runs in the experiment $\mathsf{Expt}^{\mathtt{sPA1}}_{\Gamma_{\mathtt{in}},\mathcal{A}',\mathcal{E},1}(k)$, with a corresponding extractor $\mathcal{E}$:

$\mathcal{A}'^{\mathcal{E}(\mathsf{st}_{\mathcal{E}},\cdot)}(pk'; r_{\mathcal{A}'} = (r_{\mathcal{A}}, \widehat{r}_c, \widehat{r}_t, K^*))$**:** $\mathcal{A}'$ firstly sets $pk_{\mathtt{in}} \leftarrow pk'$ (which implicitly sets $sk_{\mathtt{in}} \leftarrow sk'$, where sk' is the secret key corresponding to pk'), and runs $(ck, \mathsf{tag}^*) \leftarrow \mathsf{oSamp}_{\mathcal{C}}(1^k; \widehat{r}_c)$, $(pk, c^*, \widehat{sk}_{\mathsf{tag}^*}) \leftarrow \mathsf{oSamp}_{\mathcal{T}}(\mathsf{tag}^*; \widehat{r}_t)$. Then $\mathcal{A}'$ sets $PK \leftarrow (pk_{\mathtt{in}}, pk, ck)$ and $C^* \leftarrow (\mathsf{tag}^*, c^*)$, and then runs $\mathcal{A}(PK, C^*, K^*; r_{\mathcal{A}})$.

When $\mathcal{A}$ submits a decapsulation query C, $\mathcal{A}'$ responds to it as if it runs $\mathsf{AltDecap}(\widehat{SK}_{\mathsf{tag}^*}, C)$ where the oracle call of $\mathcal{E}$ with the input ciphertext $c_{\mathtt{in}}$ is used as a substitute for $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in}}, c_{\mathtt{in}})$.[10] More precisely, $\mathcal{A}'$ answers $\mathcal{A}$'s decapsulation query $C = (\mathsf{tag}, c)$ as follows:

1. If $\mathsf{tag} = \mathsf{tag}^*$, then return $\perp$ to $\mathcal{A}$.
2. Run $c_{\mathtt{in}} \leftarrow \widehat{\mathsf{TDec}}(\widehat{sk}_{\mathsf{tag}^*}, \mathsf{tag}, c)$, and return $\perp$ to $\mathcal{A}$ if $\widehat{\mathsf{TDec}}$ has returned $\perp$.
3. Submit a query $c_{\mathtt{in}}$ to the extractor $\mathcal{E}(\mathsf{st}_{\mathcal{E}}, \cdot)$ and receive the answer α. (Here, the answer α is expected to be $\alpha = \mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in}}, c_{\mathtt{in}})$, and the extractor $\mathcal{E}$ may update its state upon each call.)
4. If $\alpha = \perp$, then return $\perp$ to $\mathcal{A}$.
5. Parse α as $(r_c, r_t, K) \in (\{0,1\}^k)^3$.
6. If $\mathsf{Com}(ck, c_{\mathtt{in}}; r_c) = \mathsf{tag}$ and $\mathsf{TEnc}(pk, c_{\mathtt{in}}; r_t) = c$, then return K, otherwise return $\perp$, to $\mathcal{A}$.

When $\mathcal{A}$ terminates, $\mathcal{A}'$ also terminates.

The above completes the description of the algorithm $\mathcal{A}'$. The randomness $r_{\mathcal{A}'}$ consumed by $\mathcal{A}'$ is of the form $(r_{\mathcal{A}}, \widehat{r}_c, \widehat{r}_t, K^*)$, where $r_{\mathcal{A}}$, $\widehat{r}_c$, and $\widehat{r}_t$ are the randomness used by $\mathcal{A}$, $\mathsf{oSamp}_{\mathcal{C}}$, and $\mathsf{oSamp}_{\mathcal{T}}$, respectively, and K^* is a k-bit string. The corresponding extractor $\mathcal{E}$ thus receives pk' and $r_{\mathcal{A}'}$ as its initial state $\mathsf{st}_{\mathcal{E}}$. Note that since $\Gamma_{\mathtt{in}}$ is assumed to be $\mathtt{sPA1}_1$ secure and $\mathcal{A}'$ is a PPTA, $\mathsf{Adv}^{\mathtt{sPA1}}_{\Gamma_{\mathtt{in}},\mathcal{A}',\mathcal{E},1}(k)$ is negligible for this extractor $\mathcal{E}$, which will be used later in the proof. (Looking ahead, we will design the sequence of games so that $\mathcal{A}$'s view in the case $\mathcal{A}$ is internally run by $\mathcal{A}'$ and $\mathcal{A}'$ is run in $\mathsf{Expt}^{\mathtt{sPA1}}_{\Gamma_{\mathtt{in}},\mathcal{A}',\mathcal{E},1}(k)$, is identical to $\mathcal{A}$'s view in Game 6.)

For convenience, we refer to the procedure of using the extractor $\mathcal{E}$ as a substitute for $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in}}, \cdot)$, as $\overline{\mathsf{AltDecap}}'_{\mathcal{E}}$. Here, $\overline{\mathsf{AltDecap}}'_{\mathcal{E}}$ is a stateful procedure that initially takes tag^*, $\widehat{sk}_{\mathsf{tag}^*}$, and an initial state of $\mathcal{E}$ (i.e. $\mathsf{st}_{\mathcal{E}} = (pk_{\mathtt{in}}, r_{\mathcal{A}'})$) as input, and expects to receive a ciphertext $C = (\mathsf{tag}, c)$ as an input. If it receives a ciphertext $C = (\mathsf{tag}, c)$, it calculates the decapsulation result K (or $\perp$) as $\mathcal{A}'$ does for $\mathcal{A}$, using $\widehat{sk}_{\mathsf{tag}^*}$ and the extractor $\mathcal{E}$, where $\mathcal{E}$'s internal state could be updated upon each call.

Now, using the adversary $\mathcal{A}$ and the extractor $\mathcal{E}$, consider the following sequence of games: (Here, the values with asterisk (*) represent those related to the challenge ciphertext for $\mathcal{A}$.)

Game 1: This is the experiment $\mathsf{Expt}^{\mathtt{CCA}}_{\Gamma,\mathcal{A}}(k)$ itself.

Game 2: Same as Game 1, except that all decapsulation queries $C = (\mathsf{tag}, c)$ satisfying $\mathsf{tag} = \mathsf{tag}^*$ are answered with $\perp$.

Game 3: Same as Game 2, except that all decapsulation queries C are answered with $\overline{\mathsf{AltDecap}}(\widehat{SK}_{\mathsf{tag}^*}, C)$, where $\widehat{SK}_{\mathsf{tag}^*}$ is the alternative secret key corresponding to (PK, SK) and tag^*.

[10] Since in this proof we only treat an extractor that works in the single key setting, we will denote a query to $\mathcal{E}$ by c, instead of $(1, c)$, for notational convenience.

Game 4: In this game, we use $\overline{\mathsf{AltDecap}}'_{\mathcal{E}}$ (defined as above) as $\mathcal{A}$'s decapsulation oracle, where the initial state of $\mathcal{E}$ (used internally by $\overline{\mathsf{AltDecap}}'_{\mathcal{E}}$) is prepared using the "inverting algorithms" $\mathsf{rSamp}_{\mathcal{C}}$ of $\mathcal{C}$ and $\mathsf{rSamp}_{\mathcal{T}}$ of $\mathcal{T}$. We also change the ordering of the steps so that they do not affect $\mathcal{A}$'s view.

More precisely, this game is defined as follows (here, we are displaying the description of Game 4 so that it is easy to compare the difference with Game 4 used in the proof of Theorem 1):

```
Game 4:
  $(pk_{\mathtt{in}}, sk_{\mathtt{in}}) \leftarrow \mathsf{KKG}_{\mathtt{in}}(1^k)$;
  $(c^*_{\mathtt{in}}, \alpha^*) \leftarrow \mathsf{Encap}_{\mathtt{in}}(pk_{\mathtt{in}})$;
  Parse $\alpha^*$ as $(r^*_c, r^*_t, K^*_1) \in (\{0,1\}^k)^3$;
  (Continue to the center column $\nearrow$)

$r_g \leftarrow \{0,1\}^*$;
$ck \leftarrow \mathsf{CKG}(1^k; r_g)$;
$\mathsf{tag}^* \leftarrow \mathsf{Com}(ck, c^*_{\mathtt{in}}; r^*_c)$;
$\widehat{r}_c \leftarrow \mathsf{rSamp}_{\mathcal{C}}(r_g, r^*_c, c^*_{\mathtt{in}})$;
$r'_g \leftarrow \{0,1\}^*$;
$(pk, sk) \leftarrow \mathsf{TKG}(1^k; r'_g)$;
$\widehat{sk}_{\mathsf{tag}^*} \leftarrow \mathsf{Punc}(sk, \mathsf{tag}^*)$;
$c^* \leftarrow \mathsf{TEnc}(pk, \mathsf{tag}^*, c^*_{\mathtt{in}}; r^*_t)$;
$\widehat{r}_t \leftarrow \mathsf{rSamp}_{\mathcal{T}}(r'_g, r^*_t, \mathsf{tag}^*, c^*_{\mathtt{in}})$;
(Continue to the right column $\nearrow$)

$PK \leftarrow (pk_{\mathtt{in}}, pk, ck)$;
$C^* \leftarrow (\mathsf{tag}^*, c^*)$;
$K^*_0 \leftarrow \{0,1\}^k$;
$b \leftarrow \{0,1\}$;
$r_{\mathcal{A}} \leftarrow \{0,1\}^*$;
$r_{\mathcal{A}'} \leftarrow (r_{\mathcal{A}}, \widehat{r}_c, \widehat{r}_t, K^*_1)$;
$\mathsf{st}_{\mathcal{E}} \leftarrow (pk_{\mathtt{in}}, r_{\mathcal{A}'})$;
$b' \leftarrow \mathcal{A}^{\mathcal{O}}(PK, C^*, K^*_b; r_{\mathcal{A}})$
```

where the decapsulation oracle $\mathcal{O}$ that $\mathcal{A}$ has access in Game 4 is $\overline{\mathsf{AltDecap}}'_{\mathcal{E}}$ (which initially receives $\mathsf{tag}^*, \widehat{sk}_{\mathsf{tag}^*}, \mathsf{st}_{\mathcal{E}} = (pk_{\mathtt{in}}, r_{\mathcal{A}'})$ as input). Note that the extractor $\mathcal{E}$ used internally by $\overline{\mathsf{AltDecap}}'_{\mathcal{E}}$ may update its state $\mathsf{st}_{\mathcal{E}}$ upon each execution.

Game 5: Same as Game 4, except that $r^*_c, r^*_t, K^*_1 \in \{0,1\}^k$ are picked uniformly at random, independently of α^* and $c^*_{\mathtt{in}}$. That is, the steps "$(c^*_{\mathtt{in}}, \alpha^*) \leftarrow \mathsf{Encap}_{\mathtt{in}}(pk_{\mathtt{in}})$; Parse α^* as $(r^*_c, r^*_t, K^*_1) \in (\{0,1\}^k)^3$" in Game 4 are replaced with the steps "$(c^*_{\mathtt{in}}, \alpha^*) \leftarrow \mathsf{Encap}_{\mathtt{in}}(pk_{\mathtt{in}})$; $r^*_c, r^*_t, K^*_1 \leftarrow \{0,1\}^k$," and we do not use α^* anymore.

Game 6: Same as Game 5, except that the key/commitment pair (ck, tag^*) and the key/ciphertext pair (pk, c^*) and the punctured secret key $\widehat{sk}_{\mathsf{tag}^*}$ are sampled obliviously, and correspondingly the randomness $\widehat{r}_c$ and $\widehat{r}_t$ used for oblivious sampling are used in $r_{\mathcal{A}'}$.

More precisely, the steps "$r_g, r^*_c \leftarrow \{0,1\}^*$; $ck \leftarrow \mathsf{CKG}(1^k; r_g)$; $\mathsf{tag}^* \leftarrow \mathsf{Com}(ck, c^*_{\mathtt{in}}; r^*_c)$; $\widehat{r}_c \leftarrow \mathsf{rSamp}_{\mathcal{C}}(r_g, r^*_c, c^*_{\mathtt{in}})$" in Game 5 are replaced with the steps "$\widehat{r}_c \leftarrow \{0,1\}^*$; $(ck, \mathsf{tag}^*) \leftarrow \mathsf{oSamp}_{\mathcal{C}}(1^k; \widehat{r}_c)$".

Furthermore, the steps "$r'_g, r^*_t \leftarrow \{0,1\}^k$; $(pk, sk) \leftarrow \mathsf{TKG}(1^k; r'_g)$; $c^* \leftarrow \mathsf{TEnc}(pk, \mathsf{tag}^*, c^*_{\mathtt{in}}; r^*_t)$; $\widehat{r}_t \leftarrow \mathsf{rSamp}_{\mathcal{T}}(r'_g, r^*_t, \mathsf{tag}^*, c^*_{\mathtt{in}})$" in Game 5 are replaced with the steps "$\widehat{r}_t \leftarrow \{0,1\}^*$; $(pk, \widehat{sk}_{\mathsf{tag}^*}, c^*) \leftarrow \mathsf{oSamp}_{\mathcal{T}}(\mathsf{tag}^*; \widehat{r}_t)$".

The above completes the description of the games.

For $i \in [6]$, let Succ_i denote the event that $\mathcal{A}$ succeeds in guessing the challenge bit (i.e. $b' = b$ occurs) in Game i. Furthermore, for $i \in \{3, \dots, 6\}$, we define the following *bad* events in Game i:

Bad_i**:** $\mathcal{A}$ submits a decapsulation query $C = (\mathsf{tag}, c)$ satisfying the following conditions simultaneously: (1) $\mathsf{tag} \neq \mathsf{tag}^*$, (2) $\widehat{\mathsf{TDec}}(\widehat{sk}_{\mathsf{tag}^*}, \mathsf{tag}, c) = c_{\mathtt{in}} \neq \bot$, and (3) $c_{\mathtt{in}} = c^*_{\mathtt{in}}$ *or* $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in}}, c_{\mathtt{in}}) \neq \mathcal{E}(\mathsf{st}_{\mathcal{E}}, c_{\mathtt{in}})$.

$\mathsf{Bad}^{(j)}_i$**:** (where $j \in [Q]$) $\mathcal{A}$'s j-th query $C_j = (\mathsf{tag}_j, c_j)$ satisfies the conditions of Bad_i. Namely, it satisfies: (1) $\mathsf{tag}_j \neq \mathsf{tag}^*$, (2) $\widehat{\mathsf{TDec}}(\widehat{sk}_{\mathsf{tag}^*}, \mathsf{tag}_j, c_j) = c_{\mathtt{in}j} \neq \bot$, and (3) $c_{\mathtt{in}j} = c^*_{\mathtt{in}}$ or $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in}}, c_{\mathtt{in}j}) \neq \mathcal{E}(\mathsf{st}_{\mathcal{E}}, c_{\mathtt{in}j})$.

By the definitions of the games and events, we can show the following:

Claim 8 *$\mathcal{A}$'s* $\mathtt{CCA}$ *advantage* $\mathsf{Adv}^{\mathtt{CCA}}_{\Gamma,\mathcal{A}}(k)$ *can be upperbounded as follows:*

$$\mathsf{Adv}^{\mathtt{CCA}}_{\Gamma,\mathcal{A}}(k) \leq 2 \cdot \sum_{i \in [2]} \Big| \Pr[\mathsf{Succ}_i] - \Pr[\mathsf{Succ}_{i+1}] \Big| + 2 \cdot \Big| \Pr[\mathsf{Succ}_4] - \Pr[\mathsf{Succ}_5] \Big| + 2 \cdot \Big| \Pr[\mathsf{Succ}_5] - \frac{1}{2} \Big|$$
$$+ 2 \cdot \sum_{i \in \{4,5\}} \Big| \sum_{j \in [Q]} \Big(\Pr[\mathsf{Bad}^{(j)}_i] - \Pr[\mathsf{Bad}^{(j)}_{i+1}] \Big) \Big| + 2 \cdot \sum_{j \in [Q]} \Pr[\mathsf{Bad}^{(j)}_6]. \quad (10)$$

Proof of Claim 8. By the definitions of the games and events and the triangle inequality, we have:

$$\mathsf{Adv}^{\mathtt{CCA}}_{\Gamma,\mathcal{A}}(k) = 2 \cdot \Big| \Pr[\mathsf{Succ}_1] - \frac{1}{2} \Big| \leq 2 \cdot \sum_{i \in [4]} \Big| \Pr[\mathsf{Succ}_i] - \Pr[\mathsf{Succ}_{i+1}] \Big| + 2 \cdot \Big| \Pr[\mathsf{Succ}_5] - \frac{1}{2} \Big|.$$

Furthermore, notice that Game 3 and Game 4 proceed identically unless $\mathcal{A}$ submits a decapsulation query that causes the event Bad_3 (or Bad_4), and thus these games proceed identically unless Bad_3 or Bad_4 occurs in the corresponding games. Thus, we have:

$$\Big| \Pr[\mathsf{Succ}_3] - \Pr[\mathsf{Succ}_4] \Big| \leq \Pr[\mathsf{Bad}_3] = \Pr[\mathsf{Bad}_4].$$

Then, further applying the triangle inequality and the union bound, we have

$$\Pr[\mathsf{Bad}_4] = \Pr[\bigvee_{j \in [Q]} \mathsf{Bad}^{(j)}_4] \leq \sum_{j \in [Q]} \Pr[\mathsf{Bad}^{(j)}_4]$$
$$\leq \sum_{i \in \{4,5\}} \Big| \sum_{j \in [Q]} \Big(\Pr[\mathsf{Bad}^{(j)}_i] - \Pr[\mathsf{Bad}^{(j)}_{i+1}] \Big) \Big| + \sum_{j \in [Q]} \Pr[\mathsf{Bad}^{(j)}_6].$$

Combining all the inequalities yields Equation (10). □ (**Claim 8**)

In the following, we upperbound each term that appears in Equation (10).

Claim 9 *There exists a PPTA* $\mathcal{B}_{\mathtt{b}}$ *such that* $\mathsf{Adv}^{\mathtt{TBind}}_{\mathcal{C},\mathcal{B}_{\mathtt{b}}}(k) \geq |\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_2]|$.

Claim 10 $\Pr[\mathsf{Succ}_2] = \Pr[\mathsf{Succ}_3]$.

Claim 11 *There exists a PPTA* $\mathcal{B}'_{\mathtt{g}}$ *such that* $\mathsf{Adv}^{\mathtt{CPA}}_{\Gamma_{\mathtt{in}},\mathcal{B}'_{\mathtt{g}}}(k) = |\Pr[\mathsf{Succ}_4] - \Pr[\mathsf{Succ}_5]|$.

Claim 12 $\Pr[\mathsf{Succ}_5] = 1/2$.

The proofs of Claims 9, 10, 11, and 12 are essentially the same as the proofs of Claims 1, 2, 4, and 5, respectively, and thus omitted.

Claim 13 *There exists a PPTA* $\mathcal{B}_{\mathtt{g}}$ *such that* $\mathsf{Adv}^{\mathtt{1\text{-}CCA}}_{\Gamma_{\mathtt{in}},\mathcal{B}_{\mathtt{g}}}(k) = (1/Q) \cdot |\sum_{j \in [Q]} (\Pr[\mathsf{Bad}^{(j)}_4] - \Pr[\mathsf{Bad}^{(j)}_5])|$.

Proof of Claim 13. Using $\mathcal{A}$ and $\mathcal{E}$ as building blocks, we show how to construct a PPTA 1-$\mathtt{CCA}$ adversary $\mathcal{B}_{\mathtt{g}}$ against the underlying KEM $\Gamma_{\mathtt{in}}$ with the claimed advantage. The description of $\mathcal{B}_{\mathtt{g}}$ is as follows:

$\mathcal{B}_{\mathsf{g}}^{\mathcal{O}}(pk', c'^*, \alpha'^*_\beta)$**:** (where $\beta \in \{0,1\}$ is $\mathcal{B}_{\mathsf{g}}$'s challenge bit in its $\mathtt{1\text{-}CCA}$ experiment, and $\mathcal{O}$ is $\mathcal{B}_{\mathsf{g}}$'s decapsulation oracle) $\mathcal{B}_{\mathsf{g}}$ sets $pk_{\mathtt{in}} \leftarrow pk'$, $c^*_{\mathtt{in}} \leftarrow c'^*$, and $\alpha^* \leftarrow \alpha'^*_\beta$, and parses α^* as $(r^*_c, r^*_t, K^*_1) \in (\{0,1\}^k)^3$. Next, $\mathcal{B}_{\mathsf{g}}$ picks $r_g, r'_g \in \{0,1\}^*$ uniformly at random, and runs $ck \leftarrow \mathsf{CKG}(1^k; r_g)$, $\mathsf{tag}^* \leftarrow \mathsf{Com}(ck, c^*_{\mathtt{in}}; r^*_c)$, $\widehat{r}_c \leftarrow \mathsf{rSamp}_{\mathcal{C}}(r_g, r^*_c, c^*_{\mathtt{in}})$, $(pk, sk) \leftarrow \mathsf{TKG}(1^k; r'_g)$, $c^* \leftarrow \mathsf{TEnc}(pk, \mathsf{tag}^*, c^*_{\mathtt{in}}; r^*_t)$, $\widehat{sk}_{\mathsf{tag}^*} \leftarrow \mathsf{Punc}(sk, \mathsf{tag}^*)$, and $\widehat{r}_t \leftarrow \mathsf{rSamp}_{\mathcal{T}}(r'_g, r^*_t, \mathsf{tag}^*, c^*_{\mathtt{in}})$. Then $\mathcal{B}_{\mathsf{g}}$ picks $r_{\mathcal{A}} \in \{0,1\}^*$, $K^*_0 \in \{0,1\}^k$ and $b \in \{0,1\}$ uniformly at random, and then sets $PK \leftarrow (pk_{\mathtt{in}}, pk, ck)$, $C^* \leftarrow (\mathsf{tag}^*, c^*)$, $r_{\mathcal{A}'} \leftarrow (r_{\mathcal{A}}, \widehat{r}_c, \widehat{r}_t, K^*_b)$, and $\mathsf{st}_{\mathcal{E}} \leftarrow (pk_{\mathtt{in}}, r_{\mathcal{A}'})$, and runs $\mathcal{A}(PK, C^*, K^*_b; r_{\mathcal{A}})$.
$\mathcal{B}_{\mathsf{g}}$ answers $\mathcal{A}$'s decapsulation queries as $\overline{\mathsf{AltDecap}}'_{\mathcal{E}}$ does, where its initial state is tag^*, $\widehat{sk}_{\mathsf{tag}^*}$, and $\mathsf{st}_{\mathcal{E}}$. (Note that $\mathcal{E}$ used internally by $\overline{\mathsf{AltDecap}}'_{\mathcal{E}}$ may update its state $\mathsf{st}_{\mathcal{E}}$ upon each execution.) When $\mathcal{A}$ terminates, $\mathcal{B}_{\mathsf{g}}$ picks $u \in [Q]$ uniformly at random, and checks if $\mathcal{A}$'s u-th query $C_u = (\mathsf{tag}_u, c_u)$ satisfies the conditions of the event Bad_4. Namely, $\mathcal{B}_{\mathsf{g}}$ checks whether (1) $\mathsf{tag}_u \neq \mathsf{tag}^*$, (2) $\widehat{\mathsf{TDec}}(\widehat{sk}_{\mathsf{tag}^*}, \mathsf{tag}_u, c_u) = c_{\mathtt{in}u} \neq \bot$ and (3) $c_{\mathtt{in}u} = c^*_{\mathtt{in}}$ or $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in}}, c_{\mathtt{in}u}) \neq \mathcal{E}(\mathsf{st}_{\mathcal{E}}, c_{\mathtt{in}u})$, where the condition (3) can be checked by using $\mathcal{B}_{\mathsf{g}}$'s decapsulation oracle.[11] (Here, if $c_{\mathtt{in}u} = c^*_{\mathtt{in}}$, then $\mathcal{B}_{\mathsf{g}}$ need not use the decapsulation oracle.) If the u-th query satisfies the above, then $\mathcal{B}_{\mathsf{g}}$ sets $\beta' \leftarrow 1$, otherwise sets $\beta' \leftarrow 0$, and terminates with output β'.

The above completes the description of $\mathcal{B}_{\mathsf{g}}$. Note that $\mathcal{B}_{\mathsf{g}}$ never submits the prohibited query $c^*_{\mathtt{in}}$. For $j \in [Q]$, let $\mathsf{Bad}^{(j)}_{\mathcal{B}}$ be the event that $\mathcal{A}$ submits a decapsulation query that satisfies the conditions (1), (2), and (3) of $\mathsf{Bad}^{(j)}_4$, in the experiment simulated by $\mathcal{B}'_{\mathsf{g}}$. It is easy to see that $\mathcal{B}_{\mathsf{g}}$ simulates Game 4 perfectly for $\mathcal{A}$ if $\beta = 1$, and simulates Game 5 perfectly for $\mathcal{A}$ if $\beta = 0$. Furthermore, note that the choice of $u \in [Q]$ is independent of $\mathcal{A}$'s behavior and the challenge bit of $\mathcal{B}_{\mathsf{g}}$. These imply that for every $j \in [Q]$ and $\sigma \in \{0,1\}$, we have $\Pr[\mathsf{Bad}^{(u)} | u = j \wedge \beta = \sigma] = \Pr[\mathsf{Bad}^{(j)}_{5-\sigma}]$ and $\Pr[u = j | \beta = \sigma] = \Pr[u = j] = 1/Q$. Since $\mathcal{B}_{\mathsf{g}}$ outputs $\beta' = 1$ only when $\mathsf{Bad}^{(u)}_{\mathcal{B}}$ occurs, for both $\sigma \in \{0,1\}$, we have

$$\begin{aligned}\Pr[\beta' = 1 | \beta = \sigma] &= \Pr[\mathsf{Bad}^{(u)}_{\mathcal{B}} | \beta = \sigma] \\ &= \sum_{j \in [Q]} \Pr[\mathsf{Bad}^{(u)}_{\mathcal{B}} | u = j \wedge \beta = \sigma] \cdot \Pr[u = j | \beta = \sigma] \\ &= \frac{1}{Q} \cdot \sum_{j \in [Q]} \Pr[\mathsf{Bad}^{(j)}_{5-\sigma}].\end{aligned}$$

Using this, we can calculate $\mathcal{B}_{\mathsf{g}}$'s $\mathtt{1\text{-}CCA}$ advantage as follows:

$$\begin{aligned}\mathsf{Adv}^{\mathtt{1\text{-}CCA}}_{\Gamma_{\mathtt{in}}, \mathcal{B}_{\mathsf{g}}}(k) &= 2 \cdot \left| \Pr[\beta' = \beta] - \frac{1}{2} \right| = \left| \Pr[\beta' = 1 | \beta = 1] - \Pr[\beta' = 1 | \beta = 0] \right| \\ &= \frac{1}{Q} \cdot \Big| \sum_{j \in [Q]} \Big(\Pr[\mathsf{Bad}^{(j)}_4] - \Pr[\mathsf{Bad}^{(j)}_5] \Big) \Big|,\end{aligned}$$

as required. □ (**Claim 13**)

Claim 14 *There exists a PPTA* $\mathcal{B}_{\mathsf{d}}$ *such that* $\mathsf{Adv}^{\mathtt{TS}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}}(k) = (1/Q) \cdot \big| \sum_{j \in [Q]} (\Pr[\mathsf{Bad}^{(j)}_5] - \Pr[\mathsf{Bad}^{(j)}_6]) \big|$.

[11] As mentioned in Section 4.2, the step (3) can be done by a "plaintext-checking" query by submitting a pair $(c_{\mathtt{in}u}, \alpha_u)$ to the plaintext-checking oracle (which tells the one-bit $(\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in}}, c_{\mathtt{in}u}) \overset{?}{=} \alpha_u)$), where α_u is the decapsulation result of $c_{\mathtt{in}u}$ that was computed by using the extractor $\mathcal{E}$.

Proof of Claim 14. Using $\mathcal{A}$ and $\mathcal{E}$ as building blocks, we show how to construct a PPTA $\mathcal{B}$ that has the claimed advantage in distinguishing the distributions considered in Lemma 2. The description of $\mathcal{B}_{\mathsf{d}} = (\mathcal{B}_{\mathsf{d}1}, \mathcal{B}_{\mathsf{d}2})$ as follows:

$\mathcal{B}_{\mathsf{d}1}(1^k)$**:** $\mathcal{B}_{\mathsf{d}1}$ runs $(pk_{\mathtt{in}}, sk_{\mathtt{in}}) \leftarrow \mathsf{KKG}_{\mathtt{in}}(1^k)$ and $(c^*_{\mathtt{in}}, \alpha^*) \leftarrow \mathsf{Encap}_{\mathtt{in}}(pk_{\mathtt{in}})$. Then $\mathcal{B}_{\mathsf{d}1}$ sets $M \leftarrow c^*_{\mathtt{in}}$ and $\mathsf{st}_{\mathcal{B}} \leftarrow$ ($\mathcal{B}_{\mathsf{d}1}$'s entire view), and terminates with output $(M, \mathsf{st}_{\mathcal{B}})$.

$\mathcal{B}_{\mathsf{d}2}(\mathsf{st}_{\mathcal{B}}, ck, \mathsf{tag}^*, pk, c^*, \widehat{sk}_{\mathsf{tag}^*}, \widehat{r}_c, \widehat{r}_t)$**:** $\mathcal{B}_{\mathsf{d}2}$ sets $PK \leftarrow (pk_{\mathtt{in}}, pk, ck)$, $C^* \leftarrow (\mathsf{tag}^*, c^*)$, picks $K^* \in \{0,1\}^k$ and $r_{\mathcal{A}} \in \{0,1\}^*$ uniformly at random, and sets $r_{\mathcal{A}'} \leftarrow (r_{\mathcal{A}}, \widehat{r}_c, \widehat{r}_t, K^*)$ and $\mathsf{st}_{\mathcal{E}} \leftarrow (pk_{\mathtt{in}}, r_{\mathcal{A}'})$. (Recall that K^*_0 and K^*_1 in Games 5 and 6 are distributed identically, and thus it is sufficient to choose just a single value K^* and pretend as if K^* is K^*_b.) Then $\mathcal{B}_{\mathsf{d}2}$ runs $\mathcal{A}(PK, C^*, K^*; r_{\mathcal{A}})$.

$\mathcal{B}_{\mathsf{d}2}$ answers $\mathcal{A}$'s queries as Game 5 does, which is possible because $\mathcal{B}_{\mathsf{d}2}$ possesses $\widehat{sk}_{\mathsf{tag}^*}$ and $\mathsf{st}_{\mathcal{E}}$, and thus $\mathcal{B}_{\mathsf{d}2}$ can run $\overline{\mathsf{AltDecap}}'_{\mathcal{E}}$ (which internally runs the extractor $\mathcal{E}(\mathsf{st}_{\mathcal{E}}, \cdot)$).

When $\mathcal{A}$ terminates, $\mathcal{B}_{\mathsf{d}2}$ picks $u \in [Q]$ uniformly at random, and checks whether $\mathcal{A}$'s u-th decapsulation query $C_u = (\mathsf{tag}_u, c_u)$ satisfies the conditions of Bad_5, namely, (1) $\mathsf{tag}_u \neq \mathsf{tag}^*$, (2) $\widehat{\mathsf{TDec}}(\widehat{sk}_{\mathsf{tag}^*}, \mathsf{tag}_u, c_u) = c_{\mathtt{in}u} \neq \bot$, and (3) $c_{\mathtt{in}u} = c^*_{\mathtt{in}}$ or $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in}}, c_{\mathtt{in}u}) \neq \mathcal{E}(\mathsf{st}_{\mathcal{E}}, c_{\mathtt{in}u})$, which can be checked by using $sk_{\mathtt{in}}$ that $\mathcal{B}_{\mathsf{d}2}$ possesses. If the u-th query satisfies the above, then $\mathcal{B}_{\mathsf{d}2}$ outputs 1, otherwise outputs 0, and terminates.

The above completes the description of $\mathcal{B}_{\mathsf{d}}$. For $j \in [Q]$, let $\mathsf{Bad}^{(j)}_{\mathcal{B}}$ be the event that $\mathcal{A}$'s j-th decapsulation query $C_j = (\mathsf{tag}_j, c_j)$ satisfies the conditions of $\mathsf{Bad}^{(j)}_5$ in the experiment simulated by $\mathcal{B}_{\mathsf{d}}$. Note that $\mathcal{B}_{\mathsf{d}}$ submits 1 only when $\mathsf{Bad}^{(u)}_{\mathcal{B}}$ occurs. Furthermore, the choice of u is independent of $\mathcal{A}$'s behavior and whether $\mathcal{B}_{\mathsf{d}}$ is in $\mathsf{Expt}^{\mathtt{TS\text{-}Real}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}}(k)$ or $\mathsf{Expt}^{\mathtt{TS\text{-}Sim}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}}(k)$. These imply that for every $j \in [Q]$, we have $\Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Real}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}} : \mathsf{Bad}^{(u)}_{\mathcal{B}} | u = j] = \Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Real}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}} : \mathsf{Bad}^{(j)}_{\mathcal{B}}]$, $\Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Sim}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}} : \mathsf{Bad}^{(u)}_{\mathcal{B}} | u = j] = \Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Sim}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}} : \mathsf{Bad}^{(j)}_{\mathcal{B}}]$, and $\Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Real}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}} : u = j] = \Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Sim}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}} : u = j] = 1/Q$. Therefore, $\mathcal{B}_{\mathsf{d}}$'s advantage $\mathsf{Adv}^{\mathtt{TS}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}}(k)$ (regarding distinguishing $\mathsf{Expt}^{\mathtt{TS\text{-}Real}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}}(k)$ and $\mathsf{Expt}^{\mathtt{TS\text{-}Sim}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}}(k)$) can be calculated as follows:

$$\begin{aligned}
\mathsf{Adv}^{\mathtt{TS}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}}(k) &= \Big| \Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Real}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}}(k) = 1] - \Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Sim}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}}(k) = 1] \Big| \\
&= \Big| \Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Real}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}} : \mathsf{Bad}^{(u)}_{\mathcal{B}}] - \Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Sim}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}}(k) : \mathsf{Bad}^{(u)}_{\mathcal{B}}] \Big| \\
&= \Big| \sum_{j \in [Q]} \Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Real}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}} : \mathsf{Bad}^{(u)}_{\mathcal{B}} | u = j] \cdot \Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Real}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}} : u = j] \\
&\qquad - \sum_{j \in [Q]} \Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Sim}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}}(k) : \mathsf{Bad}^{(u)}_{\mathcal{B}} | u = j] \cdot \Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Sim}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}} : u = j] \Big| \\
&= \frac{1}{Q} \cdot \Big| \sum_{j \in [Q]} \Big(\Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Real}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}} : \mathsf{Bad}^{(j)}_{\mathcal{B}}] - \Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Sim}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}}(k) : \mathsf{Bad}^{(j)}_{\mathcal{B}}] \Big) \Big|.
\end{aligned}$$

Consider the case when $\mathcal{B}_{\mathsf{d}}$ is run in the "real" experiment $\mathsf{Expt}^{\mathtt{TS\text{-}Real}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}}(k)$. It is easy to see that in this case, $\mathcal{B}_{\mathsf{d}}$ simulates Game 5 perfectly for $\mathcal{A}$. Specifically, ck, pk, tag^*, c^*, and $\widehat{sk}_{\mathsf{tag}^*}$ are generated from CKG, TKG, Com, TEnc, and Punc, respectively, in such a way that tag^* is a commitment of $c^*_{\mathtt{in}}$ and c^* is an encryption of $c^*_{\mathtt{in}}$ under the tag tag^*. Furthermore, $\widehat{r}_c$ and $\widehat{r}_t$ are generated from $\mathsf{rSamp}_{\mathcal{C}}$ and $\mathsf{rSamp}_{\mathcal{T}}$, respectively, which is exactly how they are generated in Game 5. Under the situation, the probability that $\mathcal{A}$'s j-th query satisfies the conditions of the event $\mathsf{Bad}^{(j)}_{\mathcal{B}}$ in the experiment simulated by $\mathcal{B}_{\mathsf{d}}$ is exactly the same as the probability that $\mathcal{A}$'s j-th query satisfies those in Game 5. Namely, for every $j \in [Q]$, we have $\Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Real}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}}(k) : \mathsf{Bad}^{(j)}_{\mathcal{B}}] = \Pr[\mathsf{Bad}^{(j)}_{5}]$.

On the other hand, consider the case when $\mathcal{B}_{\mathsf{d}}$ is run in the "simulated" experiment $\mathsf{Expt}^{\mathtt{TS\text{-}Sim}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}}(k)$. In this case, $\mathcal{B}_{\mathsf{d}}$ simulates Game 6 perfectly for $\mathcal{A}$. Specifically, (ck, tag^*) and $(pk, c^*, \widehat{sk}_{\mathsf{tag}^*})$ are generated by $\mathsf{oSamp}_{\mathcal{C}}(1^k; \widehat{r}_c)$ and $\mathsf{oSamp}_{\mathcal{T}}(\mathsf{tag}^*; \widehat{r}_t)$ with uniformly chosen randomness $\widehat{r}_c$ and $\widehat{r}_t$, respectively, and this is exactly how these values are generated in Game 6. Since this is the only change from the above case, with a similar argument to the above, for every $j \in [Q]$ we have $\Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Sim}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}}(k) : \mathsf{Bad}^{(j)}_{\mathcal{B}}] = \Pr[\mathsf{Bad}^{(j)}_{6}]$.

In summary, we have $\mathsf{Adv}^{\mathtt{TS}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}}(k) = (1/Q) \cdot |\sum_{j \in [Q]}(\Pr[\mathsf{Bad}^{(j)}_{5}] - \Pr[\mathsf{Bad}^{(j)}_{6}])|$, as required. □ (**Claim 14**)

Claim 15 *For every $j \in [Q]$, we have* $\Pr[\mathsf{Bad}^{(j)}_{6}] \le \mathsf{Adv}^{\mathtt{sPA1}}_{\Gamma_{\mathtt{in}},\mathcal{A}',\mathcal{E},1}(k) + \mathsf{Smth}_{\Gamma_{\mathtt{in}}}(k)$.

Proof of Claim 15. Note that the view of $\mathcal{A}$ in Game 6 is exactly the same as the view of $\mathcal{A}$ when it is internally run by $\mathcal{A}'$ in the situation where $\mathcal{A}'$ is run in the experiment $\mathsf{Expt}^{\mathtt{sPA1}}_{\Gamma_{\mathtt{in}},\mathcal{A}',\mathcal{E},1}(k)$ with the extractor $\mathcal{E}$.

Now, we classify $\mathcal{A}$'s decapsulation query $C = (\mathsf{tag}, c)$ satisfying the conditions of Bad_6 into the following two types:

- (Type 1): $\mathsf{tag} \ne \mathsf{tag}^*$, $\widehat{\mathsf{TDec}}(\widehat{sk}_{\mathsf{tag}^*}, \mathsf{tag}, c) = c_{\mathtt{in}} \ne \bot$, and $c_{\mathtt{in}} = c^*_{\mathtt{in}}$.
- (Type 2): $\mathsf{tag} \ne \mathsf{tag}^*$, $\widehat{\mathsf{TDec}}(\widehat{sk}_{\mathsf{tag}^*}, \mathsf{tag}, c) = c_{\mathtt{in}} \ne \bot$, $c_{\mathtt{in}} \ne c^*_{\mathtt{in}}$, and $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in}}, c_{\mathtt{in}}) \ne \mathcal{E}(\mathsf{st}_{\mathcal{E}}, c_{\mathtt{in}})$

Note that in Game 6, $\mathcal{A}$'s view does not contain any information of $c^*_{\mathtt{in}}$ and thus $c^*_{\mathtt{in}}$ is information-theoretically hidden from $\mathcal{A}$. Under this situation, the event that $\mathcal{A}$ submits a query of Type 1 corresponds to the event that $\mathcal{A}$ succeeds in guessing an "unseen" ciphertext $c^*_{\mathtt{in}}$. However, for every $j \in [Q]$, the probability that $\mathcal{A}$'s j-th query is of Type 1 can be upperbounded by the "smoothness" $\mathsf{Smth}_{\Gamma_{\mathtt{in}}}(k)$ [4] of the KEM $\Gamma_{\mathtt{in}}$ (which is negligible if $\Gamma_{\mathtt{in}}$ satisfies $\mathtt{CPA}$ or stronger security, see Lemma 4 in Appendix A.2).

Furthermore, the probability that $\mathcal{A}$ submits a query of Type 2 is upperbounded by the probability that $\mathcal{A}'$ submits a query to $\mathcal{E}$ that makes the experiment $\mathsf{Expt}^{\mathtt{sPA1}}_{\Gamma_{\mathtt{in}},\mathcal{A}',\mathcal{E},1}(k)$ outputs 1 (i.e. $\mathcal{A}'$ submits a query $c_{\mathtt{in}}$ such that $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in}}, c_{\mathtt{in}}) \ne \mathcal{E}(\mathsf{st}_{\mathcal{E}}, c_{\mathtt{in}})$). Hence, for every $j \in [Q]$, the probability that $\mathcal{A}$'s j-th query is of Type 2 is upperbounded by $\mathsf{Adv}^{\mathtt{sPA1}}_{\Gamma_{\mathtt{in}},\mathcal{A}',\mathcal{E},1}(k)$.

In summary, for every $j \in [Q]$, the probability that $\mathcal{A}$'s j-th query satisfies the conditions of Bad_6 is upperbounded by $\mathsf{Adv}^{\mathtt{sPA1}}_{\Gamma_{\mathtt{in}},\mathcal{A}',\mathcal{E},1}(k) + \mathsf{Smth}_{\Gamma_{\mathtt{in}}}(k)$, as required. □ (**Claim 15**)

Claims 8 to 15 and Equation (10) guarantee that there exist PPTAs $\mathcal{B}_{\mathsf{b}}$, $\mathcal{B}_{\mathsf{g}}$, $\mathcal{B}_{\mathsf{d}}$, and $\mathcal{B}'_{\mathsf{g}}$ such that

$$\begin{aligned}\mathsf{Adv}^{\mathtt{CCA}}_{\overline{\Gamma},\mathcal{A}}(k) \le 2 \cdot \mathsf{Adv}^{\mathtt{TBind}}_{\mathcal{C},\mathcal{B}_{\mathsf{b}}}(k) + 2 \cdot \mathsf{Adv}^{\mathtt{CPA}}_{\Gamma_{\mathtt{in}},\mathcal{B}'_{\mathsf{g}}}(k) + 2Q \cdot \mathsf{Adv}^{\mathtt{1\text{-}CCA}}_{\Gamma_{\mathtt{in}},\mathcal{B}_{\mathsf{g}}} + 2Q \cdot \mathsf{Adv}^{\mathtt{TS}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}}(k)\\ + 2Q \cdot \mathsf{Adv}^{\mathtt{sPA1}}_{\Gamma_{\mathtt{in}},\mathcal{A}',\mathcal{E},1}(k) + 2Q \cdot \mathsf{Smth}_{\Gamma_{\mathtt{in}}}(k),\end{aligned}$$

which, due to our assumptions on the building blocks and Lemmas 2 and 4 (where the latter is stated in Appendix A.2), implies that $\mathsf{Adv}^{\mathtt{CCA}}_{\overline{\Gamma},\mathcal{A}}(k)$ is negligible. Recall that the choice of the PPTA $\mathtt{CCA}$ adversary $\mathcal{A}$ was arbitrarily, and thus for any PPTA $\mathtt{CCA}$ adversary $\mathcal{A}$ we can show a negligible upperbound for $\mathsf{Adv}^{\mathtt{CCA}}_{\overline{\Gamma},\mathcal{A}}(k)$ as above. Hence, the KEM $\overline{\Gamma}$ is $\mathtt{CCA}$ secure. □ (**Theorem 2**)

```
$\mathsf{KKG}_{\mathtt{DF}}(1^k)$ :
    $\forall (i,j) \in [k] \times \{0,1\}$ :
        $(pk_i^{(j)}, sk_i^{(j)}) \leftarrow \mathsf{KKG}(1^k)$
    $\kappa \leftarrow \mathsf{HKG}(1^k)$
    $pk \leftarrow ((pk_i^{(j)})_{i\in[k],j\in\{0,1\}}, \kappa)$
    $sk \leftarrow ((sk_i^{(j)})_{i\in[k],j\in\{0,1\}}, \kappa)$
    Return $(pk, sk)$.
```

```
$\mathsf{Encap}_{\mathtt{DF}}(pk)$ :
    $((pk_i^{(j)})_{i\in[k],j\in\{0,1\}}, \kappa) \leftarrow pk$
    $(vk, sigk) \leftarrow \mathsf{SKG}(1^k)$
    $h \leftarrow \mathsf{H}_\kappa(vk)$
    View $h$ as $(h_1 \| \dots \| h_k) \in \{0,1\}^k$.
    $\forall i \in [k] : (c_i, K_i) \leftarrow \mathsf{Encap}(pk_i^{(h_i)})$
    $\sigma \leftarrow \mathsf{Sign}(sigk, (c_i)_{i\in[k]})$
    $c \leftarrow (vk, (c_i)_{i\in[k]}, \sigma)$
    $K \leftarrow \bigoplus_{i\in[k]} K_i$
    Return $(c, K)$.
```

```
$\mathsf{Decap}_{\mathtt{DF}}(sk, c)$ :
    $((sk_i^{(j)})_{i\in[k],j\in\{0,1\}}, \kappa) \leftarrow sk$
    $(vk, (c_i)_{i\in[k]}, \sigma) \leftarrow c$
    If $\mathsf{SVer}(vk, (c_i)_{i\in[k]}, \sigma) = \perp$ then return $\perp$.
    $h \leftarrow \mathsf{H}_\kappa(vk)$
    View $h$ as $(h_1 \| \dots \| h_k) \in \{0,1\}^k$.
    $\forall i \in [k] : K_i \leftarrow \mathsf{Decap}(sk^{(h_i)}, c_i)$
    If $\exists i \in [k] : K_i = \perp$ then return $\perp$.
    Return $K \leftarrow \bigoplus_{i\in[k]} K_i$.
```

Fig. 7. A KEM variant of the Dodis-Fiore construction $\mathit{\Gamma}_{\mathtt{DF}}$.

D A KEM Variant of the Dodis-Fiore Construction

Here, we recall the KEM-variant of the Dodis-Fiore construction [22, Appendix C]. The original construction in [22] constructs a PKE scheme from a building block PKE scheme and a one-time signature scheme. Here, we construct a KEM by replacing the building block PKE scheme with a KEM. The reasons for considering such a KEM-variant are that (1) it is sufficient for our second construction, and (2) using a KEM as a building block can in general result in smaller ciphertext size.

Formally, the construction is as follows. Let $\mathit{\Gamma} = (\mathsf{KKG}, \mathsf{Encap}, \mathsf{Decap})$ be a KEM, $\mathcal{H} = (\mathsf{HKG}, \mathsf{H})$ be a UOWHF, and $\mathit{\Sigma} = (\mathsf{SKG}, \mathsf{Sign}, \mathsf{SVer})$ be a signature scheme. Then, we construct the KEM-analogue of the Dodis-Fiore construction, denoted by $\mathit{\Gamma}_{\mathtt{DF}} = (\mathsf{KKG}_{\mathtt{DF}}, \mathsf{Encap}_{\mathtt{DF}}, \mathsf{Decap}_{\mathtt{DF}})$, as in Fig. 7.

Security of $\mathit{\Gamma}_{\mathtt{DF}}$. From the description, it is straightforward to see the following:

Theorem 3. *If the underlying KEM $\mathit{\Gamma}$ is $\mathtt{sPA1}_{2k}$ secure, then the KEM $\mathit{\Gamma}_{\mathtt{DF}}$ is $\mathtt{sPA1}_1$ secure.*

Furthermore, regarding 1-CCA security, the following can be established:

Theorem 4. *If the underlying KEM is CPA secure, the signature scheme is SOT secure, and $\mathcal{H}$ is a UOWHF, then the KEM $\mathit{\Gamma}_{\mathtt{DF}}$ is 1-CCA secure.*

Since we did not find a proof (for the original Dodis-Fiore construction) in [22], we provide it here.

Proof of Theorem 4. Let $\mathcal{A}$ be any PPTA adversary that attacks the 1-CCA security of the KEM $\mathit{\Gamma}_{\mathtt{DF}}$. Consider the following sequence of games (where the values with asterisk (*) denote those related to the challenge ciphertext $c^* = (vk^*, (c_i^*)_{i\in[k]}, \sigma^*)$):

Game 1: This is the experiment $\mathsf{Expt}^{\mathtt{1\text{-}CCA}}_{\mathit{\Gamma}_{\mathtt{DF}},\mathcal{A}}(k)$ itself.

Game 2: Same as Game 1, except that if $\mathcal{A}$'s decapsulation query $c = (vk, (c_i)_{i\in[k]}, \sigma)$ satisfies $vk = vk^*$, then it is answered with $\perp$.

Game 3: Same as Game 2, except that if $\mathcal{A}$'s decapsulation query $c = (vk, (c_i)_{i\in[k]},, \sigma)$ satisfies $vk \neq vk^*$ and $h = \mathsf{H}_\kappa(vk) = \mathsf{H}_\kappa(vk^*) = h^*$, then it is answered with $\perp$. (Thus, a decapsulation query satisfying $h = \mathsf{H}_\kappa(vk) = \mathsf{H}_\kappa(vk^*) = h^*$ is answered with $\perp$.)

For $i \in [3]$, let Succ_i be the probability that $\mathcal{A}$ succeeds in guessing the challenge bit (i.e. $b' = b$ occurs) in Game i. Then, using the triangle inequality, $\mathcal{A}$'s 1-CCA advantage can be estimated as follows:

$$\begin{aligned}\mathsf{Adv}^{\mathtt{1\text{-}CCA}}_{\Gamma_{\mathtt{DF}},\mathcal{A}}(k) &= 2 \cdot \left|\Pr[\mathsf{Succ}_1] - \frac{1}{2}\right| \\ &\leq 2 \cdot \sum_{i \in [2]} \left|\Pr[\mathsf{Succ}_i] - \Pr[\mathsf{Succ}_{i+1}]\right| + 2 \cdot \left|\Pr[\mathsf{Succ}_3] - \frac{1}{2}\right|. \end{aligned} \tag{11}$$

Hence, it remains to show that each term in the right hand side is negligible.

Firstly, note that Game 1 and Game 2 proceed identically unless $\mathcal{A}$'s query $c = (vk, (c_i)_{i\in[k]}, \sigma)$ satisfies $vk = vk^*$ and $\mathsf{Decap}_{\mathtt{DF}}(sk, c) \neq \bot$ (which in particular implies $\mathsf{SVer}(vk, (c_i)_{i\in[k]}, \sigma) = \top$) in Game 1 or Game 2. Therefore, $|\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_2]|$ is upperbounded by the probability that $\mathcal{A}$'s query satisfies the conditions in Game 1. However, by the rule of the $\mathtt{1\text{-}CCA}$ security experiment, $\mathcal{A}$'s query c must be different from c^*, and thus $vk = vk^*$ implies $((c_i)_{i\in[k]}, \sigma) \neq ((c_i^*)_{i\in[k]}, \sigma^*)$. but all of these conditions together are exactly those of violating the $\mathtt{SOT}$ security of Σ, and hence such a query is hard to find by assumption. (We omit the details of this step because it is straightforward.) Hence, $|\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_2]|$ is negligible.

Secondly, note that Game 2 and Game 3 proceed identically unless $\mathcal{A}$'s query $c = (vk, (c_i)_{i\in[k]}, \sigma)$ satisfies $vk \neq vk^*$, $\mathsf{H}_\kappa(vk) = \mathsf{H}_\kappa(vk^*)$, and $\mathsf{Decap}_{\mathtt{DF}}(sk, c) \neq \bot$. Therefore, $|\Pr[\mathsf{Succ}_2] - \Pr[\mathsf{Succ}_3]|$ is upperbounded by the probability that $\mathcal{A}$'s query satisfies these conditions. However, these are the exactly the conditions of violating the security of the UOWHF $\mathcal{H}$, and hence such a query is hard to find by assumption. (We also omit the details on this step, because this is again straightforward.) Hence, $|\Pr[\mathsf{Succ}_2] - \Pr[\mathsf{Succ}_3]|$ is negligible.

Finally, by the $\mathtt{CPA}$ security of the underlying KEM Γ, we can show that $|\Pr[\mathsf{Succ}_3] - 1/2|$ is negligible. For simplicity, in the following we assume that $\mathcal{A}$'s decapsulation query $c = (vk, (c_i)_{i\in[k]}, \sigma)$ always satisfies $\mathsf{SVer}(vk, (c_i)_{i\in[k]}, \sigma) = \top$ and $h = \mathsf{H}_\kappa(vk) \neq \mathsf{H}_\kappa(vk^*) = h^*$. [12]

Now, using $\mathcal{A}$ as a building block, we show how to construct a $\mathtt{CPA}$ adversary $\mathcal{B}$ against the underlying KEM Γ. The description of $\mathcal{B}$ is as follows:

$\mathcal{B}(pk', c'^*, K'_\beta)$: (where $\beta \in \{0,1\}$ is $\mathcal{B}$'s challenge bit) $\mathcal{B}$ first picks $u \in [k]$ uniformly at random. $\mathcal{B}$ then runs $\kappa \leftarrow \mathsf{HKG}(1^k)$, $(vk^*, sigk^*) \leftarrow \mathsf{SKG}(1^k)$, and $h^* = (h_1^* \| \dots \| h_k^*) \leftarrow \mathsf{H}_\kappa(vk^*)$. Then, $\mathcal{B}$ generates each key pair $(pk_i^{(j)}, sk_i^{(j)})$, except for $(pk_u^{(h_u^*)}, sk_u^{(h_u^*)})$, by running $\mathsf{KKG}(1^k)$ $2k-1$ times, and also generates each ciphertext/session-key pair (c_i^*, K_i^*), except for (c_u^*, K_u^*), by executing $(c_i^*, K_i^*) \leftarrow \mathsf{Encap}(pk_i^{(h_i^*)})$ for every $i \in [k] \backslash \{u\}$. Next, $\mathcal{B}$ sets $pk_u^{(h_u^*)} \leftarrow pk'$ and $c_u^* \leftarrow c'^*$, and generates $\sigma^* \leftarrow \mathsf{Sign}(sigk^*, (c_i^*)_{i\in[k]})$ and $K^* \leftarrow K'_\beta \oplus \bigoplus_{i\in[k]\backslash\{\ell\}} K_i$. Then, $\mathcal{B}$ sets $pk \leftarrow ((pk_i^{(j)})_{i\in[k], j\in\{0,1\}}, \kappa)$ and $c^* \leftarrow (vk^*, (c_i^*)_{i\in[k]}, \sigma^*)$, and runs $\mathcal{A}(pk, c^*, K^*)$.

For $\mathcal{A}$'s decapsulation query $c = (vk, (c_i)_{i\in[k]}, \sigma)$ (for which we have already assumed that $\mathsf{SVer}(vk, (c_i)_{i\in[k]}, \sigma) = \top$ and $h = \mathsf{H}_\kappa(vk) \neq h^*$), $\mathcal{B}$ checks if u is the smallest integer in the set $\{i \in [k] \mid h_i \neq h_i^*\}$. If this is *not* the case, then $\mathcal{B}$ gives up, outputs a random bit $\beta' \in \{0,1\}$, and terminates. Otherwise, $\mathcal{B}$ possesses all secret keys $(sk_i^{(h_i)})_{i\in[k]}$ needed to decrypt each c_i, [13] and thus using them $\mathcal{B}$ computes the decapsulation result of c exactly as $\mathsf{Decap}_{\mathtt{DF}}$ does, and returns the result K (or $\bot$) to $\mathcal{A}$.

When $\mathcal{A}$ terminates with output bit b', $\mathcal{B}$ also terminates with output this b'.

[12] This assumption can be easily removed by considering a "wrapper" algorithm $\mathcal{A}'$ for $\mathcal{A}$ that runs in exactly the same way as $\mathcal{A}$ but if $\mathcal{A}$'s decapsulation query does not satisfy the above, $\mathcal{A}'$ instead makes a dummy decapsulation query satisfying them, and $\mathcal{A}'$ returns $\bot$ to $\mathcal{A}$. The $\mathtt{1\text{-}CCA}$ advantage of $\mathcal{A}'$ is exactly the same as that of $\mathcal{A}$.

[13] This is because the only missing secret key that $\mathcal{B}$ does not possesses is $sk_u^{(h_u^*)}$, but $h_u \neq h_u^*$ guarantees that $sk_u^{(h_u^*)}$ is not needed to decrypt c_u.

The above completes the description of $\mathcal{B}$. Note that we have assumed that $h \neq h^*$, and thus there must exist at least one position $j \in [k]$ such that $h_j \neq h_j^*$. Let us denote by $\mathsf{Good}_{\mathcal{B}}$ the event that u *is* the smallest integer in the set $\{i \in [k] | h_i \neq h_i^*\}$. Since u is chosen uniformly at random, and its information is information-theoretically hidden from $\mathcal{A}$'s view during the experiment simulated by $\mathcal{B}$, the probability that $\mathsf{Good}_{\mathcal{B}}$ occurs is exactly $1/k$. Furthermore, if $\mathsf{Good}_{\mathcal{B}}$ occurs, $\mathcal{B}$ simulates Game 3 perfectly for $\mathcal{A}$ so that $\mathcal{A}$'s challenge bit is that of $\mathcal{B}$'s. In particular, if $\beta = 1$, then K^* is the XOR of all the correctly generated session-keys K_i, while if $\beta = 0$, then K^* is a uniformly distributed random string due to the addition of $K_0'^*$ which is also a uniformly random string. Since $\mathcal{B}$ uses $\mathcal{A}$'s output as it is, we have $\Pr[\beta' = \beta \wedge \mathsf{Good}_{\mathcal{B}}] = \Pr[\mathsf{Good}_{\mathcal{B}}] \cdot \Pr[\mathsf{Succ}_3] = (1/k) \cdot \Pr[\mathsf{Succ}_3]$. Furthermore, if $\mathsf{Good}_{\mathcal{B}}$ does not occur, $\mathcal{B}$ uses a random bit for β', which means that $\Pr[\beta' = \beta \wedge \overline{\mathsf{Good}_{\mathcal{B}}}] = (1/2) \cdot \Pr[\overline{\mathsf{Good}_{\mathcal{B}}}] = (k-1)/2k$.

In summary, $\mathcal{B}$'s $\mathtt{CPA}$ advantage is calculated as follows:

$$\begin{aligned}
\mathsf{Adv}^{\mathtt{CPA}}_{\Gamma,\mathcal{B}}(k) &= 2 \cdot \left|\Pr[\beta' = \beta] - \frac{1}{2}\right| \\
&= 2 \cdot \left|\Pr[\beta' = \beta \wedge \mathsf{Good}_{\mathcal{B}}] + \Pr[\beta' = \beta \wedge \overline{\mathsf{Good}_{\mathcal{B}}}] - \frac{1}{2}\right| \\
&= 2 \cdot \left|\frac{1}{k} \cdot \Pr[\mathsf{Succ}_3] + \frac{k-1}{2k} - \frac{1}{2}\right| \\
&= \frac{2}{k} \cdot \left|\Pr[\mathsf{Succ}_3] - \frac{1}{2}\right|.
\end{aligned}$$

This gives us $|\Pr[\mathsf{Succ}_3] - 1/2| = (k/2) \cdot \mathsf{Adv}^{\mathtt{CPA}}_{\Gamma,\mathcal{B}}(k)$. Since Γ is assumed to be $\mathtt{CPA}$ secure, the right hand side is negligible, and thus so is $|\Pr[\mathsf{Succ}_3] - 1/2|$.

We have seen that the right hand side of Equation (11) is negligible, and hence so is $\mathsf{Adv}^{\mathtt{1\text{-}CCA}}_{\Gamma_{\mathtt{DF}},\mathcal{A}}(k)$. Since the choice of $\mathcal{A}$ was arbitrarily, we can show that the 1-$\mathtt{CCA}$ advantage is negligible for any PPTA 1-$\mathtt{CCA}$ adversary $\mathcal{A}$. This means that $\Gamma_{\mathtt{DF}}$ is 1-$\mathtt{CCA}$ secure. □ (**Theorem 4**)