The Boneh-Katz Transformation, Revisited: Pseudorandom/Obliviously-Samplable PKE from Lattices and Codes and Its Application

Keita Xagawa[1]

NTT Secure Platform Laboratories, keita.xagawa.zv@hco.ntt.co.jp

Abstract. The Boneh-Katz transformation (CT-RSA 2005) converts a selectively-secure identity/tag-based encryption scheme into a public-key encryption scheme secure against chosen-ciphertext attacks. We show that if the underlying primitives are pseudorandom, then the public-key encryption scheme obtained by the Boneh-Katz transformation is also pseudorandom. A similar result holds for oblivious sampleability (Canetti and Fischlin (CRYPTO 2001)). As applications, we can construct

- pseudorandom and obliviously-samplable public-key encryption schemes from lattices and codes,
- universally-composable non-interactive bit-commitment from lattices,
- public-key steganography which is steganographically secure against adaptive chosen-covertext attacks and steganographic key-exchange from lattices and codes,
- anonymous authenticated key exchange from lattices and codes,
- public-key encryption secure against simulation-based, selective-opening chosen-ciphertext attacks from lattices and codes.

Keywords: Public-Key Encryption, Tag-Based Encryption, Post-Quantum Cryptography, the Boneh-Katz Transformation, Selective-Opening Security, Anonymity

1 Introduction

Public-key encryption (PKE) is the most basic primitive in asymmetric-key cryptography since it allows us to transmit data over the public channel securely if the receiver's encryption key is available. There are several security notions and properties of PKE and the researchers exploited those to construct interesting primitives and protocols. One of the most basic security notions is indistinguishability (IND-security) which means any efficient adversary cannot distinguish a ciphertext of plaintext with another ciphertext of another plaintext [GM84].

Anonymity and pseudorandomness: Although indistinguishability under chosen-plaintext/ciphertext attacks (IND-CPA/CCA security) ensures the confidentiality of contents [GM84, NY89, RS92], it does not imply anonymity and privacy of the receiver. Bellare, Boldyreva, Desai, and Pointcheval [BBDP01] defined indistinguishability of keys under chosen-plaintext/ciphertext attacks (IK-CPA/CCA security) to capture anonymity; in the security game, the adversary is, given two encryption keys, asked to determine which encryption key is used to encrypt a plaintext. This security notion has several applications: anonymous communication, anonymous authentication [CL01], auction [Sak00], and so on.

We also note that pseudorandom PKE is related to anonymity. We say a PKE scheme is pseudorandom (PR-secure) if its ciphertext is indistinguishable from a random string from a set specified by the security parameter, encryption key, and the length of the message. We also say a PKE scheme is strongly-pseudorandom (SPR-secure) if the set is independent of an encryption key. It is easy to see that SPR-secure PKE scheme is anonymous. Pseudorandom PKE also has applications for public-key steganography and steganographic key exchange [vH04], and backdoored pseudorandom generators (PRG) [DGG+15]. We also note that we have subliminal communication based on pseudorandom key-exchange [HPRV19], which can be constructed from PR-CPA-secure PKE if its encryption key is pseudorandom.

The constructions of SPR-CCA-secure PKE schemes from elliptic curves [Möl04] and the DDH group [Hop05] are known. To the authors' best knowledge, we have no *explicit* construction of post-quantum (S)PR-CCA-secure ones *in the standard model* except one from puncturable pseudorandom function (PRF) and indistinguishablity obfuscation (iO) [SW14, LP15].

Oblivious sampleability: Canetti and Fischlin [CF01] introduced *oblivious sampleability (OS-security)*, which is an enhancement of PR-security; oblivious sampleability requires (1) a ciphertext is indistinguishable from a random string generated by a sampling algorithm on input the encryption key and (2) an explanation algorithm to explain how one samples the random string, e.g., if a ciphertext consists of group elements, then the randomness used to make the group elements are required. Combining OS-CCA-secure PKE with trapdoor commitments, they obtain UC-secure non-interactive bit commitment against adaptive corruption without erasure [CF01]. We do not know whether every IND-CCA-secure PKE scheme is OS-CCA-secure or not [1].
This security notion is strongly related to efficiently-samplable and explainable (ESE) ciphertext space. See [FHKW10, LP15] for its application to simulation-based, sender selective-opening security against chosen-ciphertext attacks (SIM-SSO-CCA security) of PKE.
Although there are several OS-CCA-secure PKE/KEM schemes from number-theoretic assumptions (see [CF01, FHKW10, LP15]), but we have no *explicit* construction of post-quantum OS-CCA-secure ones *in the standard model* except one from puncturable pseudo-random function PRF and iO [SW14, LP15].

1.1 Our Contribution

The Boneh-Katz transformation, revisited: We revisit the Boneh-Katz (BK) transformation [BK05, BCHK07], which obtains IND-CCA-secure PKE from selectively-secure identity-based encryption (IBE) (or tag-based encryption (TBE)), weakly-secure commitment, and secure message authentication code (MAC). We show that the BK transformation *preserves* pseudorandomness and oblivious sampleability: If the underlying primitives are pseudorandom and obliviously-samplable, then the PKE scheme obtained by the transformations is also pseudorandom and obliviously-samplable, respectively.
We also note that similar results hold for Zhang's T1/T2 transformations [Zha07]. Unfortunately, they require strong properties of underlying primitives[2] and current post-quantum primitives seem hard to suffice their requirements. Thus, we omit the proofs for Zhang's T1/T2 transformations.

SPR-CCA/SOS-CCA-*secure PKEs:* Using the above theorem, we obtain SPR-CCA-secure and SOS-CCA-secure PKEs from lattices and codes with various parameter settings upon existing IBE/TBE schemes [CHKP12, ABB10, MP12, BBDQ18, DMN09, DMN12, KMP14, YZ16]. As a byproduct, we show the Kiltz-Masny-Pietrzak TBE scheme [KMP14] and the Yu-Zhang TBE scheme [YZ16] based on the LPN problems are indeed pseudorandom and obliviously-samplable without changing the assumptions.

Applications: Employing them, we then obtain

- non-interactive bit commitment that is adaptively UC-secure in the non-erasure model under a re-usable common reference string from lattices through [CF01],
- public-key steganography which is steganographically secure against adaptive chosen-covertext attacks and steganographic key-exchange from lattice and codes through [Hop05, BL18]
- anonymous authenticated key exchange from lattices and codes through [FSXY15], and
- public-key encryption secure against simulation-based, selective-opening chosen-ciphertext attacks from lattices and codes through [LP15].

Note on the Canetti-Halevi-Katz (CHK) transformation: The Canetti-Halevi-Katz (CHK) transformation [BCHK07] allows us to obtain IND-CCA-secure PKE from selectively-secure identity-based encryption (IBE) (or tag-based encryption (TBE)) and one-time signature. Unfortunately, a PKE scheme obtained by the CHK transformation cannot be obliviously-samplable even if the underlying IBE/TBE is obliviously-samplable, since we can verify the one-time signature in the ciphertext of PKE. The random string should contain the verification key of one-time signature and signature on the ciphertext of IBE/TBE. Roughly speaking, we cannot explain the randomness of the key generation of one-time signature, because once this randomness is leaked, then we can forge any message under the verification key and may be able to mount chosen-ciphertext attacks.[3]

[1] Ishai et al. [IKOS10] refuted the hypothesis that every efficiently-samplable distribution has an invertible-sampling algorithm assuming the strong version of extractable OWF and the NIWI proofs for all NP (or assuming non-interactively extractable OWF and the NIZK proofs for all NP). Although this is not applicable to PKE, this is supporting evidence.

[2] Separability of TBE for T1 and oracle collision resistance for T2.

[3] If the underlying IBE/TBE is malleable, we modify the ciphertext of the IBE/TBE, sign it with the signing key of the one-time signature, and obtain a new valid ciphertext related to the challenge ciphertext.

1.2 Related Works

Anonymous PKE: Bellare et al. [BBDP01] put forth the notion of anonymity of PKE and introduced indistinguishability of keys (IK-security). (See also Camenisch and Lysyanskaya [CL01] and Sako [Sak00].) Paterson and Srinivasan [PS08] defined Trusted Authority's anonymity (TA anonymity) of IBE. They [PS09] showed that if the underlying IBE scheme satisfies TA anonymity, then the PKE scheme obtained by the CHK transformation is also key-private. As we explained, this is not pseudorandom. They refer to the BK transformation but omit the detail. This work can be considered as the follow-up of the case of the BK conversion. We note that the anonymity of PKE is insufficient for UC-Commitment and SIM-SSO-CCA-secure PKE.

Obliviously-samplable PKE/KEM: Canetti and Fischlin [CF01] introduced the notion of *oblivious sampleability (OS-security)* and its application to UC-secure commitment. They showed that the Cramer-Shoup PKE [CS98] over the subgroup $\mathbb{G} \subseteq \mathbb{Z}_p^*$ of prime order $q \mid p-1$ satisfies their requirements because we can *explain* how to generate a random element in $\mathbb{G}$. As far as we know, there is no explicit construction of post-quantum PKE scheme satisfying OS-CCA security in the standard model except one from puncturable PRF and iO [SW14, LP15]. Thus, this paper first gives a post-quantum OS-CCA-secure PKE scheme without iO.

Public-key steganography: Public-key steganography is formalized by von Ahn and Hopper [vH04]. See Berndt and Liśkiewicz [BL18] for the survey. Backes and Cachin [BC05] studied public-key steganography against active attacks. Hopper [Hop05] also studied it and gave a construction of public-key steganography secure against adaptive chosen-covertext attacks (SS-CCA-security) against a single channel from SPR-CCA-secure PKE. Berndt and Liśkiewicz [BL18] improved the constructions to achieve SS-CCA-secure public-key steganography against every memoryless channel from SPR-CCA-secure PKE, pseudorandom permutations (PRPs), and collision-resistant hash functions (CRHFs).
Since there are no explicit constructions of post-quantum SPR-CCA-secure PKE in the standard model, our result is the first construction of such public-key steganography in the standard model.

Anonymous AKE: We next consider anonymity of authenticated key exchange (AKE), that is, the anonymity of the participants from the outsider: The outsider obtains public keys of the participants and a transcript and try to determine whether the transcript is the results of the communications between the participants or not.
In general, the signature-based AKE (e.g., the signed DH [DvW92, HC98, PQR21]), in which the messages are signed by the sender, is not anonymous from the outsider. This is because the outsider can verify the signatures in the transcripts with the participants' public keys. So, one might need to encrypt signatures to get anonymity.
On the other hand, KEM-based AKEs [BCGNP09, FSXY13, FSXY15, SSW20] could achieve anonymity from the outsider. Roughly speaking, in the KEM-based AKEs, the first message consists of pk_{tmp} and $ct_{A \to B}$ and the second message consists of ct_{tmp} and $ct_{B \to A}$, where pk_{tmp} is the encapsulation key of KEM, $ct_{A \to B}$ is a ciphertext of KEM under Bob's encapsulation key, ct_{tmp} is a ciphertext of KEM under pk_{tmp}, and $ct_{B \to A}$ is a ciphertext of KEM under Alice's encapsulation key. Thus, it achieves anonymity if the underlying KEMs are key-private or pseudorandom. Moreover, such KEM-based AKE can achieve *weak* offline deniability. [4]
Recently, a new AKE is proposed by Hashimoto, Katsumata, Kwiatkowski, and Prest [HKKP21], which is a hybrid of signature-based AKE and KEM-based AKE.[5] If the underlying PKEs are key-private and pseudorandom, then it achieves anonymity and weak offline deniability. They discuss how to achieve 'weak deniability' using stronger primitives [HKKP21, Section 6].

SIM-SSO-CCA PKE: We review PKE schemes satisfying simulation-based, sender-selective-opening security against chosen-ciphertext attack (SIM-SSO-CCA security in short). We omit the constructions in the (quantum) random oracle model or (quantum) ideal cipher model [HJKS15, HP16, SS19].

[4] The offline deniability [DGK06] requires any PPT judge cannot distinguish simulated transcripts from transcripts where one of the parties may be malicious. Here, 'weak' means that any PPT judge cannot distinguish simulated transcripts from *honestly-generated transcripts.*

[5] Very roughly speaking, the first message consists of pk_{tmp} and the second message consists of $ct_{B \to A}$, ct_{tmp}, and c, where pk_{tmp} is the encapsulation key of KEM, ct_{tmp} is a ciphertext of KEM under pk_{tmp}, $ct_{B \to A}$ is a ciphertext of KEM under Alice's encapsulation key, and c is a masked ciphertext of the signature signed by Bob.

Constructions from lossy primitives: Hemenway, Libert, Ostrovsky, and Vergnaud [HLOV11] proposed lossy encryption and showed that PKE scheme satisfying indistinguishability-based, sender-selective-opening security against chosen-ciphertext attack (IND-SSO-CCA security in short) can be constructed from a 'separable' TBE scheme satisfying a weak variant of IND-SSO-CCA security (IND-SSO-st-wCCA security) with chameleon hash following Zhang's T1 [Zha07] and commented that the CHK conversion fails because it uses one-time signature. They constructed a 'separable' IND-SSO-st-wCCA-secure TBE scheme from lossy trapdoor function (LTF) and all-but-N function. Hofheintz [Hof12] proposed all-but-many lossy trapdoor functions (ABM LTFs) based on DCR or pairing and use them to construct SIM-SSO-CCA-secure PKE schemes with compactness or tighter security, respectively. Boyen and Li [BL17] proposed ABM LTF from LWE and constructed an IND-SSO-CCA-secure PKE scheme by using their ABM LTFs. Libert, Sakzad, Stehlé, Steinfeld [LSSS17] also proposed ABM LTF from LWE, constructed an IND-SSO-CCA-secure PKE scheme by using their ABM LTFs, and then enhanced it into a SIM-SSO-CCA-secure PKE scheme. Lyu, Liu, Han, and Gu [LLHG18] gave a SIM-SSO-CCA-secure PKE scheme based on the matrix DDH assumption with a tighter security reduction. Lai, Liu, and Wang [LLW20] improved ABM LTFs with polynomial modulus from LWE.

Constructions with cross-authentication codes (XACs): Fehr, Hofheinz, Kiltz, and Wee [FHKW10] constructed a SIM-SSO-CCA-secure PKE by using extended hash proof systems with collision-resistant hash functions and cross-authentication codes (XAC). As pointed out in [HLQ13, HLQC13], a stronger property of XAC is required to make this proof rigorous. Liu and Paterson [LP15] constructed a SIM-SSO-CCA secure PKE scheme using a special KEM scheme and strengthened XAC. They constructed special KEM schemes from hash proof systems, from n-linear assumption, and from indistinguishability obfuscation (iO) and a special puncturable PRF. Libert et al. [LSSS17] wrote "*So far, the only known method [LP15] to attain the same security notion under quantum-resistant assumptions was to apply a generic construction where each bit of plaintext requires a full key encapsulation (KEM) using a CCA2-secure KEM.*" However, there is a gap between the special KEM in [LP15] and known post-quantum IND-CCA-secure KEM schemes, which we fill in this paper.
Moreover, as far as we know, there is no explicit SIM-SSO-CCA construction from LPN and codes.

1.3 Organization

We review notations and cryptographic schemes in section 2. We review the Boneh-Katz transformation and prove its pseudorandomness and oblivious sampleability in section 3. We discuss how to instantiate applications through PR-CCA-secure/OS-CCA-secure PKE in section 4. In appendix, we review the LPN-related assumptions section A, review the Kiltz-Masny-Pietrzak TBE scheme and the Yu-Zhang TBE scheme and prove their PR-CCA-security in section B and section C, respectively.

2 Definitions

Notations: A security parameter is denoted by κ. We use the standard O-notations. DPT and PPT stand for deterministic polynomial time and probabilistic polynomial time. A function $f(\kappa)$ is said to be *negligible* if $f(\kappa) = \kappa^{-\omega(1)}$. We denote a set of negligible functions by $\mathsf{negl}(\kappa)$. For a distribution χ, we often write "$x \leftarrow \chi$," which indicates that we take a sample x according to χ. For a finite set S, $U(S)$ denotes the uniform distribution over S. We often write "$x \leftarrow S$" instead of "$x \leftarrow U(S)$." For a set S and a deterministic algorithm A, $\mathsf{A}(S)$ denotes the set $\{\mathsf{A}(x) \mid x \in S\}$. If inp is a string, then "$\mathsf{out} \leftarrow \mathsf{A}(\mathsf{inp})$" denotes the output of algorithm A when run on input inp. If A is deterministic, then out is a fixed value and we write "$\mathsf{out} := \mathsf{A}(\mathsf{inp})$." We also use the notation "$\mathsf{out} := \mathsf{A}(\mathsf{inp}; r)$" to make the randomness r explicit.
For a statement P (e.g., $r \in [0, 1]$), we define $\mathrm{boole}(P) = 1$ if P is satisfied and 0 otherwise.

Efficiently-samplable and explainable domain: A domain $\mathcal{D}$ is said to be *efficiently samplable and explainable (ESE)* [FHKW10] if there are two PPT algorithms defined as follows:
- $\mathsf{Sample}(\mathcal{D}; \rho)$: On input domain $\mathcal{D}$ and random coins $\rho \leftarrow \mathcal{R}$, this algorithm outputs an element x according to the uniform distribution over $\mathcal{D}$.
- $\mathsf{Sample}^{-1}(\mathcal{D}, x)$: On input domain $\mathcal{D}$ and any $x \in \mathcal{D}$, this algorithm outputs ρ that is uniformly distributed over the set $\{\rho \in \mathcal{R} \mid \mathsf{Sample}(\mathcal{D}; \rho) = x\}$.

For example, $\mathcal{D} = \{0,1\}^\kappa$ is ESE with $\rho = \mathsf{Sample}(\mathcal{D}; \rho) = \mathsf{Sample}^{-1}(\mathcal{D}, \rho)$. Damgård and Nielsen [DN00] showed that any dense subset of an efficiently samplable domain is ESE if the dense subset allows an efficient membership test. Canetti and Fischlin [CF01] showed that the subgroup $\mathbb{G} \subseteq \mathbb{Z}_p^*$ of prime order $q \mid p - 1$ is ESE.

2.1 Public-Key Encryption (PKE)

The model for PKE schemes is summarized as follows:

Definition 2.1. *A PKE scheme* PKE *consists of the following triple of PPT algorithms* $(\mathsf{Gen}_{\mathsf{PKE}}, \mathsf{Enc}_{\mathsf{PKE}}, \mathsf{Dec}_{\mathsf{PKE}})$.

- $\mathsf{Gen}_{\mathsf{PKE}}(1^\kappa) \to (ek, dk)$*: a key-generation algorithm that on input* 1^κ*, where* κ *is the security parameter, outputs a pair of keys* (ek, dk)*.* ek *and* dk *are called the encryption key and decryption key, respectively.*
- $\mathsf{Enc}_{\mathsf{PKE}}(ek, m) \to c$*: an encryption algorithm that takes as input encryption key* ek *and message* $m \in \mathcal{M}$ *and outputs ciphertext* $c \in C$*.*
- $\mathsf{Dec}_{\mathsf{PKE}}(dk, c) \to m/\perp$*: a decryption algorithm that takes as input decryption key* dk *and ciphertext* c *and outputs message* $m \in \mathcal{M}$ *or a rejection symbol* $\perp \notin \mathcal{M}$*.*

Definition 2.2 (Correctness). *We say* $\mathsf{PKE} = (\mathsf{Gen}_{\mathsf{PKE}}, \mathsf{Enc}_{\mathsf{PKE}}, \mathsf{Dec}_{\mathsf{PKE}})$ *has* perfect correctness *if for any* (ek, dk) *generated by* $\mathsf{Gen}_{\mathsf{PKE}}$ *and for any* $m \in \mathcal{M}$*, we have*

$$\Pr[c \leftarrow \mathsf{Enc}_{\mathsf{PKE}}(ek, m) : \mathsf{Dec}_{\mathsf{PKE}}(dk, c) = m] = 1.$$

Security Notions: We review indistinguishability under chosen-ciphertext attacks (IND-CCA) [RS92, BDPR98], pseudorandom under chosen-ciphertext attacks (PR-CCA) (as known as IND\$-CCA) [vH04, Hop05], oblivious sampleability under chosen-ciphertext attacks (OS-CCA) [CF01] and their strong versions (SPR-CCAand SOS-CCA) for PKE.

In order to define oblivious sampleability, we introduce two additional algorithms, $\mathsf{Rnd}_{\mathsf{PKE}}$ and $\mathsf{Expl}_{\mathsf{PKE}}$: $\mathsf{Rnd}_{\mathsf{PKE}}$ takes an encryption key ek, a length of message 0^ℓ, and randomness $\rho \in \mathcal{R}_{\mathsf{Rnd}_{\mathsf{PKE}}, ek, \ell}$ and outputs $c \in C$; $\mathsf{Expl}_{\mathsf{PKE}}$ takes ek and $c \in C$ and outputs a randomness ρ. Roughly speaking, we say a PKE scheme is obliviously samplable if there exist $\mathsf{Rnd}_{\mathsf{PKE}}$ and $\mathsf{Expl}_{\mathsf{PKE}}$ that a dummy ciphertext c generated by $\mathsf{Rnd}_{\mathsf{PKE}}$ with randomness ρ and a real ciphertext c^* of m^* and corresponding fake randomness ρ^* generated by $\mathsf{Expl}_{\mathsf{PKE}}$ are indistinguishable.

Definition 2.3 (Security notions for PKE). *Let* $\mathcal{D}_{\mathcal{M}}$ *be a distribution over the message space* $\mathcal{M}$*. For any adversary* $\mathcal{A}$*, we define its* IND-CCA*,* PR-CCA*, and* OS-CCA *advantages against a PKE scheme* $\mathsf{PKE} = (\mathsf{Gen}_{\mathsf{PKE}}, \mathsf{Enc}_{\mathsf{PKE}}, \mathsf{Dec}_{\mathsf{PKE}})$ *and two additional PPT algorithms* $\mathsf{Rnd}_{\mathsf{PKE}}$ *and* $\mathsf{Expl}_{\mathsf{PKE}}$ *as follows:*

$$\begin{aligned}
\mathsf{Adv}^{\text{ind-cca}}_{\mathsf{PKE},\mathcal{A}}(\kappa) &:= \left|\Pr[\mathsf{Expt}^{\text{ind-cca},0}_{\mathsf{PKE},\mathcal{A}}(\kappa) = 1] - \Pr[\mathsf{Expt}^{\text{ind-cca},1}_{\mathsf{PKE},\mathcal{A}}(\kappa) = 1]\right|,\\
\mathsf{Adv}^{\text{pr-cca}}_{\mathsf{PKE},\mathcal{A}}(\kappa) &:= \left|\Pr[\mathsf{Expt}^{\text{pr-cca},0}_{\mathsf{PKE},\mathcal{A}}(\kappa) = 1] - \Pr[\mathsf{Expt}^{\text{pr-cca},1}_{\mathsf{PKE},\mathcal{A}}(\kappa) = 1]\right|,\\
\mathsf{Adv}^{\text{os-cca}}_{\mathsf{PKE},\mathcal{A}}(\kappa) &:= \left|\Pr[\mathsf{Expt}^{\text{os-cca},0}_{\mathsf{PKE},\mathcal{A}}(\kappa) = 1] - \Pr[\mathsf{Expt}^{\text{os-cca},1}_{\mathsf{PKE},\mathcal{A}}(\kappa) = 1]\right|,
\end{aligned}$$

where $\mathsf{Expt}^{\text{ind-cca},b}_{\mathsf{PKE},\mathcal{A}}(\kappa)$*,* $\mathsf{Expt}^{\text{pr-cca},b}_{\mathsf{PKE},\mathcal{A}}(\kappa)$*, and* $\mathsf{Expt}^{\text{os-cca},b}_{\mathsf{PKE},\mathcal{A}}(\kappa)$ *are experiments described in Figure 1. We say that* PKE *is* IND-CCA*-secure,* PR-CCA*-secure, and* OS-CCA*-secure if* $\mathsf{Adv}^{\text{ind-cca}}_{\mathcal{A},\mathsf{PKE}}(\kappa)$*,* $\mathsf{Adv}^{\text{pr-cca}}_{\mathcal{A},\mathsf{PKE}}(\kappa)$*, and* $\mathsf{Adv}^{\text{os-cca}}_{\mathcal{A},\mathsf{PKE}}(\kappa)$ *is negligible for any PPT adversary* $\mathcal{A}$*, respectively.*

We also say that PKE *is* SPR-CCA*-secure if it is* PR-CCA*-secure and its ciphertext space* C *depends on only* κ *and is independent from* ek*. We also say that* PKE *is* SOS-CCA*-secure if it is* OS-CCA*-secure and its additional algorithms take* 1^κ *instead of* ek *as a part of input.*

Remark 2.1. We note that if a PKE scheme is PR-CCA-secure and its ciphertext space C is ESE, then the PKE scheme is OS-CCA-secure.

2.2 Tag-Based Encryption (TBE)

MacKenzie, Reiter, and Yang [MRY04] introduced a notion of *tag-based encryption* (TBE). They show that applying the CHK transformation to TBE results in IND-CCA-secure PKE independently.

The model for TBE schemes is summarized as follows:

Definition 2.4. *A TBE scheme* TBE *consists of the following triple of PPT algorithms* $(\mathsf{Gen}_{\mathsf{TBE}}, \mathsf{Enc}_{\mathsf{TBE}}, \mathsf{Dec}_{\mathsf{TBE}})$.

- $\mathsf{Gen}_{\mathsf{TBE}}(1^\kappa) \to (ek, dk)$*: a key-generation algorithm that on input* 1^κ*, where* κ *is the security parameter, outputs a pair of keys* (ek, dk)*.* ek *and* dk *are called the encryption key and decryption key, respectively.*
- $\mathsf{Enc}_{\mathsf{TBE}}(ek, \tau, m) \to c$*: an encryption algorithm that takes as input encryption key* ek*, tag* $\tau \in \mathcal{T}$*, and message* $m \in \mathcal{M}$ *and outputs ciphertext* $c \in C$*.*
- $\mathsf{Dec}_{\mathsf{TBE}}(dk, \tau, c) \to m/\perp$*: a decryption algorithm that takes as input decryption key* dk*, tag* τ*, and ciphertext* c *and outputs message* $m \in \mathcal{M}$ *or a rejection symbol* $\perp \notin \mathcal{M}$*.*

Definition 2.5 (Correctness). *We say* $\mathsf{TBE} = (\mathsf{Gen}_{\mathsf{TBE}}, \mathsf{Enc}_{\mathsf{TBE}}, \mathsf{Dec}_{\mathsf{TBE}})$ *has* perfect correctness *if for any* (ek, dk) *generated by* $\mathsf{Gen}_{\mathsf{TBE}}$*, for any tag* $\tau \in \mathcal{T}$ *and for any* $m \in \mathcal{M}$*, we have*

$$\Pr[c \leftarrow \mathsf{Enc}_{\mathsf{TBE}}(ek, \tau, m) : \mathsf{Dec}_{\mathsf{TBE}}(dk, \tau, c) = m] = 1.$$

$\underline{\mathsf{Expt}^{\text{ind-cca},b}_{\mathsf{PKE},\mathcal{A}}(\kappa)}$

$(ek, dk) \leftarrow \mathsf{Gen}_{\mathsf{PKE}}(1^\kappa)$

$(m_0, m_1, state) \leftarrow \mathcal{A}_1^{\textsc{Dec}_\perp(\cdot)}(ek)$

$c^* \leftarrow \mathsf{Enc}_{\mathsf{PKE}}(ek, m_b)$

$b' \leftarrow \mathcal{A}_2^{\textsc{Dec}_{c^*}(\cdot)}(c^*, state)$

return b'

$\underline{\textsc{Dec}_a(c)}$

if $c = a$, return $\perp$

$m := \mathsf{Dec}_{\mathsf{PKE}}(dk, c)$

return m

$\underline{\mathsf{Expt}^{\text{pr-cca},b}_{\mathsf{PKE},\mathcal{A}}(\kappa)}$

$(ek, dk) \leftarrow \mathsf{Gen}_{\mathsf{PKE}}(1^\kappa)$

$(m, state) \leftarrow \mathcal{A}_1^{\textsc{Dec}_\perp(\cdot)}(ek)$

$c_0^* \leftarrow \mathsf{Enc}_{\mathsf{PKE}}(ek, m)$

$c_1^* \leftarrow \mathcal{C}_{ek}$

$b' \leftarrow \mathcal{A}_2^{\textsc{Dec}_{c_b^*}(\cdot)}(c_b^*, state)$

return b'

$\underline{\mathsf{Expt}^{\text{os-cca},b}_{\mathsf{PKE},\mathcal{A}}(\kappa)}$

$(ek, dk) \leftarrow \mathsf{Gen}_{\mathsf{PKE}}(1^\kappa)$

$(m, state) \leftarrow \mathcal{A}_1^{\textsc{Dec}_\perp(\cdot)}(ek)$

$c_0^* \leftarrow \mathsf{Enc}_{\mathsf{PKE}}(ek, m)$

$\rho_0^* \leftarrow \mathsf{Expl}_{\mathsf{PKE}}(ek, c_0^*)$

$\rho_1^* \leftarrow \mathcal{R}_{\mathsf{Rnd}_{\mathsf{PKE}}, ek, |m|}$

$c_1^* \leftarrow \mathsf{Rnd}_{\mathsf{PKE}}(ek, 0^{|m|}; \rho_1^*)$

$b' \leftarrow \mathcal{A}_2^{\textsc{Dec}_{c_b^*}(\cdot)}(c_b^*, \rho_b^*, state)$

return b'

Fig. 1. Games for PKE schemes

Security Notions: We review indistinguishability under selective-tag and weak chosen-ciphertext attacks IND-st-wCCA [Kil06]. In addition, we define PR-st-wCCA and OS-st-wCCA by using $\mathsf{Rnd}_{\mathsf{TBE}}$ and $\mathsf{Expl}_{\mathsf{TBE}}$.
In order to define oblivious sampleability, we introduce two additional algorithms, $\mathsf{Rnd}_{\mathsf{TBE}}$ and $\mathsf{Expl}_{\mathsf{TBE}}$: $\mathsf{Rnd}_{\mathsf{TBE}}$ takes an encryption key ek, a length of message 0^ℓ, and randomness $\rho \in \mathcal{R}_{\mathsf{Rnd}_{\mathsf{TBE}}, ek, \ell}$ and outputs $c \in C$; $\mathsf{Expl}_{\mathsf{PKE}}$ takes ek and $c \in C$ and outputs a randomness ρ.

Definition 2.6 (Security notion for TBE). *For any adversary $\mathcal{A}$, we define its* IND-st-wCCA *and* OS-st-wCCA *advantages against a TBE scheme* $\mathsf{TBE} = (\mathsf{Gen}_{\mathsf{TBE}}, \mathsf{Enc}_{\mathsf{TBE}}, \mathsf{Dec}_{\mathsf{TBE}})$ *with additional PPT algorithms* $\mathsf{Rnd}_{\mathsf{TBE}}$ *and* $\mathsf{Expl}_{\mathsf{TBE}}$ *as follows:*

$$\mathsf{Adv}^{\text{ind-st-wcca}}_{\mathsf{TBE},\mathcal{A}}(\kappa) := \left|\Pr[\mathsf{Expt}^{\text{ind-st-wcca},0}_{\mathsf{TBE},\mathcal{A}}(\kappa) = 1] - \Pr[\mathsf{Expt}^{\text{ind-st-wcca},1}_{\mathsf{TBE},\mathcal{A}}(\kappa) = 1]\right|,$$
$$\mathsf{Adv}^{\text{pr-st-wcca}}_{\mathsf{TBE},\mathcal{A}}(\kappa) := \left|\Pr[\mathsf{Expt}^{\text{pr-st-wcca},0}_{\mathsf{TBE},\mathcal{A}}(\kappa) = 1] - \Pr[\mathsf{Expt}^{\text{pr-st-wcca},1}_{\mathsf{TBE},\mathcal{A}}(\kappa) = 1]\right|,$$
$$\mathsf{Adv}^{\text{os-st-wcca}}_{\mathsf{TBE},\mathcal{A}}(\kappa) := \left|\Pr[\mathsf{Expt}^{\text{os-st-wcca},0}_{\mathsf{TBE},\mathcal{A}}(\kappa) = 1] - \Pr[\mathsf{Expt}^{\text{os-st-wcca},1}_{\mathsf{TBE},\mathcal{A}}(\kappa) = 1]\right|,$$

where $\mathsf{Expt}^{\text{ind-st-wcca},b}_{\mathsf{TBE},\mathcal{A}}(\kappa)$, $\mathsf{Expt}^{\text{pr-st-wcca},b}_{\mathsf{TBE},\mathcal{A}}(\kappa)$, *and* $\mathsf{Expt}^{\text{os-st-wcca},b}_{\mathsf{TBE},\mathcal{A}}(\kappa)$ *are experiments described in Figure 2. We say that* TBE *is* IND-st-wCCA*-secure,* PR-st-wCCA*-secure, and* OS-st-wCCA*-secure if* $\mathsf{Adv}^{\text{ind-st-wcca}}_{\mathsf{TBE},\mathcal{A}}(\kappa)$, $\mathsf{Adv}^{\text{pr-st-wcca}}_{\mathsf{TBE},\mathcal{A}}(\kappa)$, *and* $\mathsf{Adv}^{\text{os-st-wcca}}_{\mathsf{TBE},\mathcal{A}}(\kappa)$ *are negligible for any PPT adversary $\mathcal{A}$, respectively.*
We also say that TBE *is* SPR-st-wCCA*-secure if it is* PR-st-wCCA*-secure and its ciphertext space C depends on only κ and is independent from ek. We also say that* TBE *is* SOS-st-wCCA*-secure if it is* OS-st-wCCA*-secure and its additional algorithms take 1^κ instead of ek.*

Remark 2.2. Again, we note that if a TBE scheme is PR-st-wCCA-secure and its ciphertext space C is ESE, then the TBE scheme is OS-st-wCCA-secure.

2.3 Weak Commitment also known as Encapsulation

Boneh et al. introduced an encapsulation [BCHK07], which is a weak variant of commitment [Blu81]. Weak commitment is summarized as follows:

Definition 2.7. *A* weak commitment scheme wCom *consists of the following triple of PPT algorithms* $(\mathsf{Init}, \mathsf{S}, \mathsf{R})$*:*

- $\mathsf{Init}(1^\kappa) \to pub$*: an initialization algorithm that takes on input 1^κ, where κ is the security parameter, and outputs a string pub.*
- $\mathsf{S}(1^\kappa, pub) \to (r, com, dec)$*: a sender algorithm that takes as input 1^κ and pub and outputs (r, com, dec) with $r \in \{0,1\}^\kappa$, where we refer to com as the commitment string and dec as the decommitment string.*

$\underline{\mathsf{Expt}^{\text{ind-st-wcca},b}_{\mathsf{TBE},\mathcal{A}}(\kappa)}$

$(\tau^*, \mathit{state}) \leftarrow \mathcal{A}_0(1^\kappa)$
$(ek, dk) \leftarrow \mathsf{Gen}_{\mathsf{TBE}}(1^\kappa)$
$(m_0, m_1, \mathit{state}) \leftarrow \mathcal{A}_1^{\textsc{Dec}_{\tau^*}(\cdot)}(ek, \mathit{state})$
$c^* \leftarrow \mathsf{Enc}_{\mathsf{TBE}}(ek, \tau^*, m_b)$
$b' \leftarrow \mathcal{A}_2^{\textsc{Dec}_{\tau^*}(\cdot)}(c^*, \mathit{state})$
return b'

$\underline{\textsc{Dec}_{\tau^*}(\tau, c)}$

if $\tau = \tau^*$, **return** $\perp$
$m := \mathsf{Dec}_{\mathsf{TBE}}(dk, \tau, c)$
return m

$\underline{\mathsf{Expt}^{\text{pr-st-wcca},b}_{\mathsf{TBE},\mathcal{A}}(\kappa)}$

$(\tau^*, \mathit{state}) \leftarrow \mathcal{A}_0(1^\kappa)$
$(ek, dk) \leftarrow \mathsf{Gen}_{\mathsf{TBE}}(1^\kappa)$
$(m, \mathit{state}) \leftarrow \mathcal{A}_1^{\textsc{Dec}_{\tau^*}(\cdot)}(ek, \mathit{state})$
$c_0^* \leftarrow \mathsf{Enc}_{\mathsf{TBE}}(ek, \tau^*, m)$
$c_1^* \leftarrow C$
$b' \leftarrow \mathcal{A}_2^{\textsc{Dec}_{\tau^*}(\cdot)}(c_b^*, \mathit{state})$
return b'

$\underline{\mathsf{Expt}^{\text{os-st-wcca},b}_{\mathsf{TBE},\mathcal{A}}(\kappa)}$

$(\tau^*, \mathit{state}) \leftarrow \mathcal{A}_0(1^\kappa)$
$(ek, dk) \leftarrow \mathsf{Gen}_{\mathsf{TBE}}(1^\kappa)$
$(m, \mathit{state}) \leftarrow \mathcal{A}_1^{\textsc{Dec}_{\tau^*}(\cdot)}(ek, \mathit{state})$
$c_0^* \leftarrow \mathsf{Enc}_{\mathsf{TBE}}(ek, \tau^*, m)$
$\rho_0^* \leftarrow \mathsf{Expl}_{\mathsf{TBE}}(ek, c_0^*)$
$\rho_1^* \leftarrow \mathcal{R}_{\mathsf{Rnd}_{\mathsf{TBE}}, ek, |m|}$
$c_1^* \leftarrow \mathsf{Rnd}_{\mathsf{TBE}}(ek, 0^{|m|}; \rho_1^*)$
$b' \leftarrow \mathcal{A}_2^{\textsc{Dec}_{\tau^*}(\cdot)}(c_b^*, \rho_b^*, \mathit{state})$
return b'

Fig. 2. Games for TBE schemes

– $\mathsf{R}(pub, com, dec) \to r/\perp$: *a receiver algorithm that takes as input* (pub, com, dec) *and outputs* $r \in \{0,1\}^\kappa$ *or a rejection symbol* $\perp \notin \{0,1\}^\kappa$.

Definition 2.8 (Correctness). *We say* $\mathsf{wCom} = (\mathsf{Init}, \mathsf{S}, \mathsf{R})$ *has* perfect correctness *if for any pub generated by* Init*, we have*

$$\Pr[(r, com, dec) \leftarrow \mathsf{S}(1^\kappa, pub) : \mathsf{R}(pub, com, dec) = r] = 1.$$

We review the definitions of hiding property and binding property [BCHK07]. We note that we here only require binding *for honestly generated commitments.* In addition, we define oblivious sampleability of encapsulation by using $\mathsf{Rnd}_{\mathsf{wCom}}$ and $\mathsf{Expl}_{\mathsf{wCom}}$. We also define non-invertibility, which states it is hard to generate meaningful decommitment for obliviously-sampled com and ρ.

Definition 2.9. *For any adversary* $\mathcal{A}$*, we define its four advantages against an encapsulation scheme* $\mathsf{wCom} = (\mathsf{Init}, \mathsf{S}, \mathsf{R})$ *and two PPT algorithms* $(\mathsf{Rnd}_{\mathsf{wCom}}, \mathsf{Expl}_{\mathsf{wCom}})$ *as follows:*

$$\begin{aligned}
\mathsf{Adv}^{\text{hiding}}_{\mathsf{wCom},\mathcal{A}}(\kappa) &:= \left|\Pr[\mathsf{Expt}^{\text{hiding},0}_{\mathsf{wCom},\mathcal{A}}(\kappa) = 1] = \Pr[\mathsf{Expt}^{\text{hiding},1}_{\mathsf{wCom},\mathcal{A}}(\kappa) = 1]\right|,\\
\mathsf{Adv}^{\text{binding}}_{\mathsf{wCom},\mathcal{A}}(\kappa) &:= \Pr[\mathsf{Expt}^{\text{binding}}_{\mathsf{wCom},\mathcal{A}}(\kappa) = 1],\\
\mathsf{Adv}^{\text{os}}_{\mathsf{wCom},\mathcal{A}}(\kappa) &:= \left|\Pr[\mathsf{Expt}^{\text{os},0}_{\mathsf{wCom},\mathcal{A}}(\kappa) = 1] - \Pr[\mathsf{Expt}^{\text{os},1}_{\mathsf{wCom},\mathcal{A}}(\kappa) = 1]\right|,\\
\mathsf{Adv}^{\text{non-inv}}_{\mathsf{wCom},\mathcal{A}}(\kappa) &:= \Pr[\mathsf{Expt}^{\text{non-inv}}_{\mathsf{wCom},\mathcal{A}}(\kappa) = 1],
\end{aligned}$$

where $\mathsf{Expt}^{\text{hiding},b}_{\mathsf{wCom},\mathcal{A}}(\kappa)$, $\mathsf{Expt}^{\text{binding}}_{\mathsf{wCom},\mathcal{A}}(\kappa)$, $\mathsf{Expt}^{\text{os},b}_{\mathsf{wCom},\mathcal{A}}(\kappa)$, *and* $\mathsf{Expt}^{\text{non-inv}}_{\mathsf{wCom},\mathcal{A}}(\kappa)$ *are experiments described in Figure 3.*
We say that wCom *is* secure *if* $\mathsf{Adv}^{\text{hiding}}_{\mathsf{wCom},\mathcal{A}}(\kappa)$ *and* $\mathsf{Adv}^{\text{binding}}_{\mathsf{wCom},\mathcal{A}}(\kappa)$ *are negligible for any PPT adversary* $\mathcal{A}$*. We also say that* wCom *is* OS-secure *if* $\mathsf{Adv}^{\text{os}}_{\mathsf{wCom},\mathcal{A}}(\kappa)$ *is negligible for any PPT adversary* $\mathcal{A}$*. We also say that* wCom *is* non-invertible *if* $\mathsf{Adv}^{\text{non-inv}}_{\mathsf{wCom},\mathcal{A}}(\kappa)$ *is negligible for any PPT adversary* $\mathcal{A}$*.*

Concrete construction: Let $\mathcal{H}_{\text{uow}} = \{H_s \colon \{0,1\}^{k_1} \to \{0,1\}^k\}$ be a family of universal one-way hash function (UOWHF) and let $\mathcal{H} = \{h \colon \{0,1\}^{k_1} \to \{0,1\}^k\}$ be a family of pairwise-independent hash function. Let $k_1 = 2k + \delta$. Boneh and Katz [BK05] gave a concrete construction of weak commitments from them as follows:

– $\mathsf{Init}(1^\kappa)$: choose H_s and h and output $pub = (h, s)$.
– $\mathsf{S}(pub)$: take $x \leftarrow \{0,1\}^{k_1}$ and output $(r, com, dec) = (h(x), H_s(x), x)$.
– $\mathsf{R}(pub, com, dec)$: output $h(dec)$ if $H_s(dec) = com$ and $\perp$ otherwise.

We require the following properties:

$\mathsf{Expt}^{\text{hiding},b}_{\mathsf{wCom},\mathcal{A}}(\kappa)$	$\mathsf{Expt}^{\text{binding}}_{\mathsf{wCom},\mathcal{A}}(\kappa)$
$pub \leftarrow \mathsf{Init}(1^\kappa)$	$pub \leftarrow \mathsf{Init}(1^\kappa)$
$(r_0, com, dec) \leftarrow \mathsf{S}(1^\kappa, pub)$	$(r, com, dec) \leftarrow \mathsf{S}(1^\kappa, pub)$
$r_1 \leftarrow \{0,1\}^\kappa$	$dec' \leftarrow \mathcal{A}(1^\kappa, pub, com, dec)$
$b' \leftarrow \mathcal{A}(1^\kappa, pub, com, r_b)$	$r' \leftarrow \mathsf{R}(pub, com, dec')$
return b'	**return** $\mathsf{boole}(r' \notin \{\perp, r\})$

$\mathsf{Expt}^{\text{os},b}_{\mathsf{wCom},\mathcal{A}}(\kappa)$	$\mathsf{Expt}^{\text{non-inv}}_{\mathsf{wCom},\mathcal{A}}(\kappa)$
$pub \leftarrow \mathsf{Init}(1^\kappa)$	$pub \leftarrow \mathsf{Init}(1^\kappa)$
$(r_0, com_0, dec_0) \leftarrow \mathsf{S}(1^\kappa, pub)$	$\rho \leftarrow \mathcal{R}_{\mathsf{Rnd}_{\mathsf{wCom}}, pub}$
$\rho_0 \leftarrow \mathsf{Expl}_{\mathsf{wCom}}(pub, com_0)$	$com \leftarrow \mathsf{Rnd}_{\mathsf{wCom}}(pub; \rho)$
$\rho_1 \leftarrow \mathcal{R}_{\mathsf{Rnd}_{\mathsf{wCom}}, pub}$	$dec \leftarrow \mathcal{A}(1^\kappa, pub, (com, \rho))$
$com_1 \leftarrow \mathsf{Rnd}_{\mathsf{wCom}}(pub; \rho_1)$	$r \leftarrow \mathsf{R}(pub, com, dec)$
$b' \leftarrow \mathcal{A}(1^\kappa, pub, (com_b, \rho_b))$	**return** $\mathsf{boole}(r \neq \perp)$
return b'	

Fig. 3. Games for weak commitment schemes

- H_s is universal one-way for the binding property. (See [BCHK07, Theorem 4].)
- $2 \cdot 2^{\frac{2k-k_1}{3}} = 2^{-\delta/3+1}$ is negligible in the security parameter for the hiding property. (See [BCHK07, Theorem 4].)
- $H_s(U(\{0,1\}^{k_1}))$ is pseudorandom for the OS property. See Lemma 2.1 below.
- $H_s(U(\{0,1\}^{k_1}))$ is pseudorandom and one-way for the non-invertible property. See Lemma 2.2 below.

We have several instantiating way of H_s.

- The easiest way is employing the standard hash functions, say, $H_s(x) = \mathsf{SHA3\text{-}256}(s, x)$. This keyed function is collision-resistant; and it is reasonable to assume that $(s, H_s(u))$ with $u \leftarrow \{0,1\}^{k_1}$ is close to uniform.
- (From lattices:) for example, Ajtai's hash function from lattices is collision-resistant if SIS is hard [Ajt96, GGH96]. This hash function is strongly universal (see e.g., Regev [Reg09, Section 5]) and, thus, pseudorandom.
- (From codes:) for example, we can use the Expand-then-Shrink hash function as known as FSB [AFS05, BLVW19, YZW+19]. Let $k_1 = k_1' \cdot w$ and $m = k_1' \cdot 2^w$ for some w. Let $\boldsymbol{e}_i$ is the i-th unit vector of dimension 2^w. The hash function is defined as $h_{\boldsymbol{M}}(x) = \boldsymbol{M} \cdot \mathsf{Expand}(x)$, where $\boldsymbol{M} \leftarrow \mathbb{Z}_2^{k \times m}$ and $\mathsf{Expand}(x) = \boldsymbol{e}_{\mathsf{int}(x_1)} \| \dots \| \boldsymbol{e}_{\mathsf{int}(x_{k_1'})} \in \mathbb{Z}_2^m$ with $x = x_1 \| \dots \| x_{k_1'}$ for each $x_i \in \mathbb{Z}_2^w$. Brakerski et al. [BLVW19] and Yu et al. [YZW+19] showed that their hash functions are collision-resistant assuming the extremely low-noise LPN. We can show its pseudorandomness by assuming the hash function is one-way by applying the result of Mol and Micciancio [MM11], which states pseudorandomness of $(g, \sum_i x_i \cdot g_i)$ with $g \leftarrow \mathbb{G}^m$ and $x \leftarrow \mathcal{X}$, where $\mathcal{X}$ is an arbitrary distribution over $\{0,1\}^m$, if $(g, f_g(x))$ is one-way.

Lemma 2.1. *Suppose that $(H_s, H_s(x))$ is computationally indistinguishable from (H_s, u), where $H_s \leftarrow \mathcal{H}_{\mathrm{uow}}, x \leftarrow \{0,1\}^{k_1}$, and $u \leftarrow \{0,1\}^k$. Then, the scheme is obliviously sampleable with $\mathcal{R}_{\mathsf{Rnd}_{\mathsf{wCom}}, pub} = \{0,1\}^k$, $\mathsf{Rnd}_{\mathsf{wCom}}(pub, \cdot)$ and $\mathsf{Expl}_{\mathsf{wCom}}(pub, \cdot)$ are the identity function over $\{0,1\}^k$.*

Proof. We consider the following three games:

- Game 0: $H_s \leftarrow \mathcal{H}_{\mathrm{uow}}$, $h \leftarrow \mathcal{H}$, $x \leftarrow \{0,1\}^{k_1}$, $com_0 \leftarrow H_s(x)$, and $\rho_0 \leftarrow \mathsf{Expl}_{\mathsf{wCom}}(pub, com_0) = com_0$. Output $b' \leftarrow \mathcal{A}(1^\kappa, (H_s, h), (com_0, \rho_0))$.
- Hybrid: $H_s \leftarrow \mathcal{H}_{\mathrm{uow}}$, $h \leftarrow \mathcal{H}$, $x \leftarrow \{0,1\}^{k_1}$, $com \leftarrow \{0,1\}^k$, and $\rho \leftarrow \mathsf{Expl}_{\mathsf{wCom}}(pub, com) = com$. Output $b' \leftarrow \mathcal{A}(1^\kappa, (H_s, h), (com, \rho))$.
- Game 1: $H_s \leftarrow \mathcal{H}_{\mathrm{uow}}$, $h \leftarrow \mathcal{H}$, $x \leftarrow \{0,1\}^{k_1}$, $\rho_1 \leftarrow \{0,1\}^k$, and $com_1 \leftarrow \mathsf{Rnd}_{\mathsf{wCom}}(pub, \rho_1) = \rho_1$. Output $b' \leftarrow \mathcal{A}(1^\kappa, (H_s, h), (com_1, \rho_1))$.

We suppose that $(H_s, H_s(x))$ is computationally indistinguishable from (H_s, u), where $H_s \leftarrow \mathcal{H}_{\mathrm{uow}}, x \leftarrow \{0,1\}^{k_1}$, and $u \leftarrow \{0,1\}^k$. Thus, it is easy to see that Game 0 and Hybrid are computationally indistinguishable. It is obvious that Hybrid and Game 1 are equivalent. Hence, the lemma follows. ◻

Lemma 2.2. *Suppose that $(H_s, H_s(x))$ is computationally indistinguishable from (H_s, u), where $H_s \leftarrow \mathcal{H}_{\mathrm{uow}}, x \leftarrow \{0,1\}^{k_1}$, and $u \leftarrow \{0,1\}^k$. Moreover, suppose that H_s is one-way. Then, the scheme is non-invertible.*

Proof. We consider the following two games:

- Game 0: $H_s \leftarrow \mathcal{H}_{\mathrm{uow}}, h \leftarrow \mathcal{H}, \rho \leftarrow \{0,1\}^k$, and $com \leftarrow \mathsf{Rnd}_{\mathsf{wCom}}(pub, \rho) = \rho$. $dec \leftarrow \mathcal{A}(1^\kappa, (H_s, h), (com, \rho))$. Output 1 if $H_s(dec) = com$ and 0 otherwise.
- Game 1: $H_s \leftarrow \mathcal{H}_{\mathrm{uow}}$, $h \leftarrow \mathcal{H}$, $x \leftarrow \{0,1\}^{k_1}$, $com \leftarrow H_s(x)$, and $\rho \leftarrow \mathsf{Expl}_{\mathsf{wCom}}(pub, com) = com$. $dec \leftarrow \mathcal{A}(1^\kappa, (H_s, h), (com, \rho))$. Output 1 if $H_s(dec) = com$ and 0 otherwise.

Game 0 is $\mathsf{Expt}^{\text{non-inv}}_{\mathsf{wCom},\mathcal{A}}(\kappa)$. In the hypothesis, we suppose that $(H_s, H_s(x))$ is computationally indistinguishable from (H_s, u), where $H_s \leftarrow \mathcal{H}_{\mathrm{uow}}, x \leftarrow \{0,1\}^{k_1}$, and $u \leftarrow \{0,1\}^k$. Thus, it is easy to see that Game 0 and Game 1 are computationally indistinguishable. Moreover, it is easy to verify that there exists an adversary $\mathcal{A}_{\mathrm{ow}}$ breaking one-wayness of H_s whose advantage is equivalent to $\Pr[\mathcal{A}$ wins Game 1$]$. Now, the lemma follows. ◻

2.4 Message Authentication Code (MAC)

The model for MAC is summarized as follows:

Definition 2.10. *A* MAC *scheme* MAC *consists of the following pair of polynomial-time algorithms* (T, V)*:*

- $\mathsf{T}(r, \mu) \rightarrow \sigma$*: a tagging algorithm that takes on input $r \in \{0,1\}^\kappa$ and a message $\mu \in \{0,1\}^*$, where κ is the security parameter, and outputs a tag σ.*
- $\mathsf{V}(r, \mu, \sigma) \rightarrow \top/\bot$*: a verification algorithm that takes as input r, μ, and a tag σ, and outputs $\top$ as "acceptance" or $\bot$ as "rejection."*

Definition 2.11 (Correctness). *We say* $\mathsf{MAC} = (\mathsf{T}, \mathsf{V})$ *has* perfect correctness *if for any $r \in \{0,1\}^\kappa$ and $\mu \in \{0,1\}^*$, we have*

$$\Pr[\sigma \leftarrow \mathsf{T}(r, \mu) : \mathsf{V}(r, \mu, \sigma) = \top] = 1.$$

We define strong existential-unforgeability against one-time chosen-message attack. In addition, we define oblivious sampleability by using $\mathsf{Rnd}_{\mathsf{MAC}}$ and $\mathsf{Expl}_{\mathsf{MAC}}$.

Definition 2.12. *For any adversary $\mathcal{A}$, we define its advantages against a MAC scheme* $\mathsf{MAC} = (\mathsf{T}, \mathsf{V})$ *and two PPT algorithms* $(\mathsf{Rnd}_{\mathsf{MAC}}, \mathsf{Expl}_{\mathsf{MAC}})$ *as follows:*

$$\mathsf{Adv}^{\text{seuf-ot-cma}}_{\mathsf{MAC},\mathcal{A}}(\kappa) := \Pr[\mathsf{Expt}^{\text{seuf-ot-cma}}_{\mathsf{MAC},\mathcal{A}}(\kappa) = 1],$$
$$\mathsf{Adv}^{\text{os}}_{\mathsf{MAC},\mathcal{A}}(\kappa) := \left|\Pr[\mathsf{Expt}^{\text{os},0}_{\mathsf{MAC},\mathcal{A}}(\kappa) = 1] - \Pr[\mathsf{Expt}^{\text{os},1}_{\mathsf{MAC},\mathcal{A}}(\kappa) = 1]\right|,$$

where $\mathsf{Expt}^{\text{seuf-ot-cma}}_{\mathsf{MAC},\mathcal{A}}(\kappa)$ and $\mathsf{Expt}^{\text{os},b}_{\mathsf{MAC},\mathcal{A}}(\kappa)$, are the experiments described in Figure 4.
We say that MAC *is* sEUF-ot-CMA-secure *and* OS-secure *if $\mathsf{Adv}^{\text{seuf-ot-cma}}_{\mathsf{MAC},\mathcal{A}}(\kappa)$ and $\mathsf{Adv}^{\text{os}}_{\mathsf{MAC},\mathcal{A}}(\kappa)$ is negligible for any PPT adversary $\mathcal{A}$, respectively.*

Concrete construction: It is known that the standard universal hash function provides a one-time secure MAC as follows: Let us identify $\{0,1\}^k$ with $\mathrm{GF}(2^k)$. For $a, b \in \{0,1\}^k$, we define $H_{a,b} : \{0,1\}^k \rightarrow \{0,1\}^k : \mu \mapsto a\mu + b \in \{0,1\}^k$. Thus, we have an sEUF-ot-CMA-secure MAC scheme unconditionally. Combining with collision-resistant hash function $h : \{0,1\}^* \rightarrow \{0,1\}^k$, we can extend the domain of the MAC as we want. Moreover, this extended MAC is OS-secure since the distribution of $\sigma = H_{a,b}(h(\mu))$ is uniform over $\{0,1\}^k$ if $a, b \leftarrow \{0,1\}^k$.

3 The Boneh-Katz transformation, Revisited

Let us review the Boneh-Katz transformation [BCHK07, Section 5] for IBE, but we here adopt it for TBE.
Let $\mathsf{TBE} = (\mathsf{Gen}_{\mathsf{TBE}}, \mathsf{Enc}_{\mathsf{TBE}}, \mathsf{Dec}_{\mathsf{TBE}})$ be a TBE scheme whose plaintext space is $\mathcal{M}_{\mathsf{TBE}} = \mathcal{M} \times \mathcal{D}$ and tag space is $\mathcal{T}$. Let $\mathsf{wCom} = (\mathsf{Init}, \mathsf{S}, \mathsf{R})$ be a weak commitment scheme whose commitment space is $\mathcal{T}$ and decommitment space is $\mathcal{D}$. Let $\mathsf{MAC} = (\mathsf{T}, \mathsf{V})$ be a MAC scheme. $\mathsf{PKE} = (\mathsf{Gen}_{\mathsf{PKE}}, \mathsf{Enc}_{\mathsf{PKE}}, \mathsf{Dec}_{\mathsf{PKE}}) = \mathsf{BK}[\mathsf{TBE}, \mathsf{wCom}, \mathsf{MAC}]$ is defined as follows:

$\mathsf{Expt}^{\text{seuf-ot-cma}}_{\mathsf{MAC},\mathcal{A}}(\kappa)$

$r \leftarrow \{0,1\}^\kappa$, $(\mu, \sigma) \leftarrow (\bot, \bot)$
$(\mu^*, \sigma^*) \leftarrow \mathcal{A}^{\textsc{Tag}(\cdot)}(1^\kappa)$
$d \leftarrow \mathsf{V}(r, \mu^*, \sigma^*)$
$p \leftarrow \mathsf{boole}((\mu, \sigma) \neq (\mu^*, \sigma^*))$
return $p \wedge d$

$\textsc{Tag}(\mu)$

if $\sigma \neq \bot$, then **return** $\bot$
return $\sigma \leftarrow \mathsf{T}(r, \mu)$

$\mathsf{Expt}^{\text{os},b}_{\mathsf{MAC},\mathcal{A}}(\kappa)$

$r \leftarrow \{0,1\}^\kappa$
$(\mu^*, \mathit{state}) \leftarrow \mathcal{A}_0(1^\kappa)$
$\sigma_0 \leftarrow \mathsf{T}(r, \mu^*)$
$\rho_0 \leftarrow \mathsf{Expl}_{\mathsf{MAC}}(\sigma_0)$
$\rho_1 \leftarrow \mathcal{R}_{\mathsf{Rnd}_{\mathsf{MAC}}}$
$\sigma_1 \leftarrow \mathsf{Rnd}_{\mathsf{MAC}}(1^\kappa; \rho_1)$
$b' \leftarrow \mathcal{A}_1(1^\kappa, (\sigma_b, \rho_b), \mathit{state})$
return b'

Fig. 4. Games for MAC schemes

Table 1. Summary of Games for the Proof of Theorem 3.1: Expl implies ρ^*_X is generated by Expl_X. Rand implies ρ^*_X is chosen from $\mathcal{R}_{\mathsf{Rand}_X}$ and a part of ct is generated by Rand_X.

Game	com^*	c^*	σ^*	ρ^*_{wCom}	ρ^*_{TBE}	ρ^*_{MAC}	$\textsc{Dec}$	When com^* is generated
Game_0	Real	Real	$\mathsf{T}(r^*, c^*)$	Expl	Expl	Expl	Original	Original
Game_1	Real	Real	$\mathsf{T}(r^*, c^*)$	Expl	Expl	Expl	Original	Generate com^* at the beginning
Game_2	Real	Real	$\mathsf{T}(r^*, c^*)$	Expl	Expl	Expl	Reject if $com = com^*$	Generate com^* at the beginning
Game_3	Real	Rand	$\mathsf{T}(r^*, c^*)$	Expl	Rand	Expl	Reject if $com = com^*$	Generate com^* at the beginning
Game_4	Real	Rand	$\mathsf{T}(r^+, c^*)$	Expl	Rand	Expl	Reject if $com = com^*$	Generate com^* at the beginning
Game_5	Real	Rand	Rand	Expl	Rand	Rand	Reject if $com = com^*$	Generate com^* at the beginning
Game_6	Rand	Rand	Rand	Rand	Rand	Rand	Reject if $com = com^*$	Generate com^* at the beginning
Game_7	Rand	Rand	Rand	Rand	Rand	Rand	Original	Original

$\mathsf{Gen}_{\mathsf{PKE}}(1^\kappa) \rightarrow (ek, dk)$

$(ek_{\mathsf{TBE}}, dk_{\mathsf{TBE}}) \leftarrow \mathsf{Gen}_{\mathsf{TBE}}(1^\kappa)$
$pub \leftarrow \mathsf{Init}(1^\kappa)$
$ek := (ek_{\mathsf{TBE}}, pub)$
$dk := dk_{\mathsf{TBE}}$
return (ek, dk)

$\mathsf{Enc}_{\mathsf{PKE}}(ek, m) \rightarrow ct$

$(r, com, dec) \leftarrow \mathsf{S}(1^\kappa, pub)$
$c \leftarrow \mathsf{Enc}_{\mathsf{TBE}}(ek_{\mathsf{TBE}}, com, (m, dec))$
$\sigma \leftarrow \mathsf{T}(r, c)$
$ct := (com, c, \sigma)$
return ct

$\mathsf{Dec}_{\mathsf{PKE}}(dk, ct) \rightarrow m/\bot$

Parse $ct = (com, c, \sigma)$
$(m, dec) \leftarrow \mathsf{Dec}_{\mathsf{TBE}}(dk_{\mathsf{TBE}}, com, c)$
if $(m, dec) = \bot$ **then return** $\bot$
$r \leftarrow \mathsf{R}(pub, com, dec)$
if $r = \bot$ **then return** $\bot$
if $\mathsf{V}(r, c, \sigma) = \bot$ **then return** $\bot$
return m

Adjusting the security proof in [BCHK07], we can show that PKE is IND-CCA secure if TBE is IND-sID-CPA secure, wCom is secure, and MAC is sEUF-OT-CMA secure, as noted (but not proven) in Kiltz [Kil06, Section 4]. We here show that PKE is OS-CCA-secure if the underlying primitives are OS-CCA-secure. The proof is easily adapted into the PR-CCA case.

Theorem 3.1. *If* TBE *is* OS-ST-wCCA*-secure,* wCom *is secure and OS-secure, and* MAC *is* sEUF-OT-CMA*-secure and OS-secure, then,* PKE *is* OS-CCA*-secure.*

We use the game-hopping proof. Let S_i denote the event that the adversary outputs $b' = 1$ in the i-th game Game_i. Let Q denote the number of decryption queries the adversary makes.

Game_0*:* This is the original game for $b = 0$. The challenge is

$$ct^*_0 = (com^*, c^*, \sigma^*) = \Big(com^*, \mathsf{Enc}_{\mathsf{TBE}}(ek_{\mathsf{TBE}}, com^*, (m^*, dec^*)), \mathsf{T}(r^*, c^*)\Big),$$

$$\rho^*_0 = (\rho^*_{\mathsf{wCom}}, \rho^*_{\mathsf{TBE}}, \rho^*_{\mathsf{MAC}}) = \Big(\mathsf{Expl}_{\mathsf{wCom}}(pub, com^*), \mathsf{Expl}_{\mathsf{TBE}}(ek_{\mathsf{TBE}}, c^*), \mathsf{Expl}_{\mathsf{MAC}}(1^\kappa, \sigma^*)\Big).$$

We have

$$\Pr[S_0] = \Pr[\mathsf{Expt}^{\text{os-cca},0}_{\mathsf{PKE},\mathcal{A}} = 1].$$

Game_1*:* We modify the game as follows: In this game, the challenger generates $pub \leftarrow \mathsf{Init}(1^\kappa)$, $(r^*, com^*, dec^*) \leftarrow \mathsf{S}(pub)$, and $(ek_{\mathsf{TBE}}, dk_{\mathsf{TBE}}) \leftarrow \mathsf{Gen}_{\mathsf{TBE}}(1^\kappa)$. It then runs the adversary on input $ek = (ek_{\mathsf{TBE}}, pub)$.
Since, this change is just conceptual, the two games are equivalent.

Lemma 3.1. *We have*

$$\Pr[S_0] = \Pr[S_1].$$

Game_2*:* We modify Game_1 as follows: The decryption oracle always rejects a query $ct = (com, c, \sigma)$ if $com = com^*$.
We define Valid as the event that $\mathcal{A}$ submits a query $ct = (com^*, c, \sigma) \neq ct^*$ which is valid, that is, the decryption result is not $\perp$. Since Game_1 and Game_2 are equivalent until Valid occurs, we have the following lemma.

Lemma 3.2. *We have*

$$|\Pr[S_1] - \Pr[S_2]| \leq \Pr[\mathsf{Valid}_1] = \Pr[\mathsf{Valid}_2].$$

Let us decompose Valid into two events:

- We define NoBind as the event that $\mathcal{A}$ queries a ciphertext $ct = (com^*, c, \sigma)$ such that $(m', dec') \leftarrow \mathsf{Dec}_{\mathsf{TBE}}(dk_{\mathsf{TBE}}, com^*, c)$, $r \leftarrow \mathsf{R}(pub, com^*, dec')$, and $r \notin \{r^*, \perp\}$.
- We also define Forge as the event that $\mathcal{A}$ queries $ct = (com^*, c, \sigma)$ such that $(c, \sigma) \neq (c^*, \sigma^*)$ and $\mathsf{V}(r^*, c, \sigma) = \top$.

Lemma 3.3. *We have*

$$\Pr[\mathsf{Valid}_2] \leq \Pr[\mathsf{NoBind}_2] + \Pr[\mathsf{Forge}_2].$$

We show that the adversary making NoBind_2 true breaks the binding property of wCom.

Lemma 3.4. *There exists a PPT adversary $\mathcal{A}_{\mathsf{wCom}}$ satisfying*

$$\Pr[\mathsf{NoBind}_2] \leq \mathsf{Adv}^{\text{binding}}_{\mathsf{wCom},\mathcal{A}_{\mathsf{wCom}}}(\kappa).$$

Proof. We construct $\mathcal{A}_{\mathsf{wCom}}$ as follows:

1. $\mathcal{A}_{\mathsf{wCom}}$ is given $(1^\kappa, pub, com^*, dec^*)$ from its challenger, where $pub \leftarrow \mathsf{Init}(1^\kappa)$ and $(r^*, com^*, dec^*) \leftarrow \mathsf{S}(1^\kappa, pub)$. It obtains $r^* \leftarrow \mathsf{R}(pub, com^*, dec^*)$. It generates $(ek_{\mathsf{TBE}}, dk_{\mathsf{TBE}}) \leftarrow \mathsf{Gen}_{\mathsf{TBE}}(1^\kappa)$, and runs $\mathcal{A}$ on input $ek := (ek_{\mathsf{TBE}}, pub)$.
2. $\mathcal{A}_{\mathsf{wCom}}$ generates the challenge on a query m from $\mathcal{A}$ as follows: it computes $ct_0^* = (com^*, c^*, \sigma^*)$ with $c^* \leftarrow \mathsf{Enc}_{\mathsf{TBE}}(ek_{\mathsf{TBE}}, com^*, (m, dec^*))$ and $\sigma^* \leftarrow \mathsf{T}(r^*, c^*)$ and generates ρ_0^* by randomness sampling algorithms. It sends ct_0^* and ρ_0^* to $\mathcal{A}$.
3. $\mathcal{A}_{\mathsf{wCom}}$ simulates the decryption oracle in Game_2 by using its decryption key dk_{TBE} as follows: it obtains $(m', dec') \leftarrow \mathsf{Dec}_{\mathsf{TBE}}(dk_{\mathsf{TBE}}, com, c)$, $r \leftarrow \mathsf{R}(pub, com, dec')$, and outputs m' if $(m', dec') \neq \perp$, $r \neq \perp$, and $\mathsf{V}(r, c, \sigma) = \top$. Once $\mathcal{A}_{\mathsf{wCom}}$ detects NoBind, that is, on the query (com^*, c, σ), it obtains $(m', dec') \leftarrow \mathsf{Dec}_{\mathsf{TBE}}(dk_{\mathsf{TBE}}, com^*, c)$ and $r' \leftarrow \mathsf{R}(pub, com^*, dec')$ with $r' \notin \{r^*, \perp\}$, then $\mathcal{A}_{\mathsf{wCom}}$ outputs dec' and halts.

Since the simulation of Game_2 is perfect, $\mathcal{A}$ correctly works. Once NoBind occurs, $\mathcal{A}_{\mathsf{wCom}}$ breaks the binding property. Thus, the lemma holds. □

Game_3*:* We modify Game_2 as follows: In this game, the challenge ciphertext is

$$ct^* = (com^*, c^*, \sigma^*) = \Big(com^*, \mathsf{Rnd}_{\mathsf{TBE}}(ek_{\mathsf{TBE}}, 0^{|m|+|dec^*|}; \rho^*_{\mathsf{TBE}}), \mathsf{T}(r^*, c^*)\Big),$$

$$\rho^* = (\rho^*_{\mathsf{wCom}}, \rho^*_{\mathsf{TBE}}, \rho^*_{\mathsf{MAC}}) = \Big(\mathsf{Expl}_{\mathsf{wCom}}(com^*), \rho^*_{\mathsf{TBE}}, \mathsf{Expl}_{\mathsf{MAC}}(\sigma^*)\Big).$$

Lemma 3.5. *There exists a PPT adversary $\mathcal{A}_{\mathsf{TBE}}$ satisfying*

$$|\Pr[S_2] - \Pr[S_3]| \leq \mathsf{Adv}^{\text{os-st-wcca}}_{\mathsf{TBE},\mathcal{A}_{\mathsf{TBE}}}(\kappa).$$

Proof. We construct a PPT adversary $\mathcal{A}_{\mathsf{TBE}}$ as follows:

1. $\mathcal{A}_{\mathsf{TBE}}$ on input 1^κ, it generates $pub \leftarrow \mathsf{Init}(1^\kappa)$ and $(r^*, com^*, dec^*) \leftarrow \mathsf{S}(1^\kappa, pub)$. It declares com^* as the challenge tag.
2. The challenger generates $(ek_{\mathsf{TBE}}, dk_{\mathsf{TBE}}) \leftarrow \mathsf{Gen}_{\mathsf{TBE}}(1^\kappa)$ and $\mathcal{A}_{\mathsf{TBE}}$ receives ek_{TBE}.
3. $\mathcal{A}_{\mathsf{TBE}}$ runs $\mathcal{A}$ on input $ek := (ek_{\mathsf{TBE}}, pub)$.
4. $\mathcal{A}_{\mathsf{TBE}}$ simulates the challenge ciphertext on input m from $\mathcal{A}$ as follows: It sends (m, dec^*) to its challenger and receives c^*_γ and ρ^*_γ, which is a real ciphertext $\mathsf{Enc}_{\mathsf{TBE}}(ek_{\mathsf{TBE}}, com^*, (m, dec^*))$ if $\gamma = 0$ and a random ciphertext if $\gamma = 1$. It generates a tag $\sigma^* \leftarrow \mathsf{T}(r^*, c^*_\gamma)$. It also generates randomness ρ^*_{wCom} and ρ^*_{MAC} by using $\mathsf{Expl}_{\mathsf{wCom}}$ and $\mathsf{Expl}_{\mathsf{MAC}}$. It sends $ct^* = (com^*, c^*_\gamma, \sigma^*)$ and $\rho^* = (\rho^*_{\mathsf{wCom}}, \rho^*_\gamma, \rho^*_{\mathsf{MAC}})$ to $\mathcal{A}$.
5. $\mathcal{A}_{\mathsf{TBE}}$ simulates the decryption oracle as follows: Upon receiving $ct = (com, c, \sigma)$, if $com = com^*$, then it returns $\perp$. Otherwise, it queries com and c to its decryption oracle. If it receives $\perp$, then return $\perp$; Otherwise, that is, it receives (m', dec'). It computes $r' \leftarrow \mathsf{R}(pub, com, dec')$. If $r' = \perp$, then it returns $\perp$. Otherwise, it computes $d \leftarrow \mathsf{V}(r', c, \sigma)$. If $d = \perp$, then it returns $\perp$; Otherwise, it returns m'
6. Eventually, $\mathcal{A}$ outputs its guess b'. $\mathcal{A}_{\mathsf{TBE}}$ also outputs b' as its guess γ'.

If $\gamma = 0$, then $\mathcal{A}_{\mathsf{TBE}}$ perfectly simulates Game$_2$. If $\gamma = 1$, then $\mathcal{A}_{\mathsf{TBE}}$ perfectly simulates Game$_3$. Thus, the lemma follows. ◻

Lemma 3.6. *There exists a PPT adversary $\mathcal{A}'_{\mathsf{TBE}}$ satisfying*

$$\left|\Pr[\mathsf{Forge}_2] - \Pr[\mathsf{Forge}_3]\right| \le \mathsf{Adv}^{\text{os-st-wcca}}_{\mathsf{TBE},\mathcal{A}'_{\mathsf{TBE}}}(\kappa).$$

Proof. We construct a PPT adversary $\mathcal{A}'_{\mathsf{TBE}}$ as follows:

1. $\mathcal{A}'_{\mathsf{TBE}}$ on input 1^κ, it generates $pub \leftarrow \mathsf{Init}(1^\kappa)$ and $(r^*, com^*, dec^*) \leftarrow \mathsf{S}(1^\kappa, pub)$. It declares com^* as the challenge tag.
2. The challenger generates $(ek_{\mathsf{TBE}}, dk_{\mathsf{TBE}}) \leftarrow \mathsf{Gen}_{\mathsf{TBE}}(1^\kappa)$ and $\mathcal{A}_{\mathsf{TBE}}$ receives ek_{TBE}.
3. $\mathcal{A}_{\mathsf{TBE}}$ runs $\mathcal{A}$ on input $ek := (ek_{\mathsf{TBE}}, pub)$.
4. $\mathcal{A}_{\mathsf{TBE}}$ simulates the challenge ciphertext on input m from $\mathcal{A}$ as follows: It sends (m, dec^*) to its challenger and receives c^*_γ and ρ^*_γ, which is a real ciphertext $\mathsf{Enc}_{\mathsf{TBE}}(ek_{\mathsf{TBE}}, com^*, (m, dec^*))$ if $\gamma = 0$ and a random ciphertext if $\gamma = 1$. It generates a tag $\sigma^* \leftarrow \mathsf{T}(r^*, c^*_\gamma)$. It also generates randomness ρ^*_{wCom} and ρ^*_{MAC} by using $\mathsf{Expl}_{\mathsf{wCom}}$ and $\mathsf{Expl}_{\mathsf{MAC}}$. It sends $ct^* = (com^*, c^*_\gamma, \sigma^*)$ and $\rho^* = (\rho^*_{\mathsf{wCom}}, \rho^*_\gamma, \rho^*_{\mathsf{MAC}})$ to $\mathcal{A}$.
5. $\mathcal{A}_{\mathsf{TBE}}$ simulates the decryption oracle as follows: Upon receiving $ct = (com, c, \sigma)$, if $com = com^*$, then it returns $\perp$; in addition, if $\mathsf{V}(r^*, c, \sigma) = \top$ and $(c, \sigma) \neq (c^*_\gamma, \sigma^*)$, then it outputs 1 and halts. If $com \neq com^*$, it queries com and c to its decryption oracle. If it receives $\perp$, then return $\perp$; Otherwise, that is, it receives (m', dec'). It computes $r' \leftarrow \mathsf{R}(pub, com, dec')$. If $r' = \perp$, then it returns $\perp$. Otherwise, it computes $d \leftarrow \mathsf{V}(r', c, \sigma)$. If $d = \perp$, then it returns $\perp$; Otherwise, it returns m'.
6. Eventually, $\mathcal{A}$ outputs its guess b' and halts. $\mathcal{A}_{\mathsf{TBE}}$ outputs 0 and halts.

If $\gamma = 0$, then $\mathcal{A}_{\mathsf{TBE}}$ perfectly simulates Game$_2$. If $\gamma = 1$, then $\mathcal{A}_{\mathsf{TBE}}$ perfectly simulates Game$_3$. Moreover, once $\mathcal{A}$ makes Forge true, then $\mathcal{A}_{\mathsf{TBE}}$ outputs 1 and halts. Thus, the lemma follows. ◻

Game_4*:* We modify Game$_3$ as follows: In this game, the challenge ciphertext is

$$ct^* = (com^*, c^*, \sigma^*) = \Big(com^*, \mathsf{Rnd}_{\mathsf{TBE}}(ek_{\mathsf{TBE}}, 0^{|m|+|dec^*|}; \rho^*_{\mathsf{TBE}}), \mathsf{T}(r^+, c^*)\Big),$$
$$\rho^* = (\rho^*_{\mathsf{wCom}}, \rho^*_{\mathsf{TBE}}, \rho^*_{\mathsf{MAC}}) = \Big(\mathsf{Expl}_{\mathsf{wCom}}(com^*), \rho^*_{\mathsf{TBE}}, \mathsf{Expl}_{\mathsf{MAC}}(\sigma^*)\Big),$$

where $r^+ \leftarrow \{0,1\}^\kappa$.
We define Forge_4 as the event that $\mathcal{A}$ queries $ct = (com^*, c, \sigma)$ such that $(c, \sigma) \neq (c^*, \sigma^*)$ and $\mathsf{V}(r^+, c, \sigma) = \top$ (instead of $\mathsf{V}(r^*, c, \sigma) = \top$).

Lemma 3.7. *There exists a PPT adversary $\mathcal{A}'_{\mathsf{wCom}}$ satisfying*

$$|\Pr[S_3] - \Pr[S_4]| \le \mathsf{Adv}^{\text{hiding}}_{\mathsf{wCom},\mathcal{A}'_{\mathsf{wCom}}}(\kappa).$$

Proof. We construct a PPT adversary $\mathcal{A}'_{\mathsf{wCom}}$ as follows:

1. $\mathcal{A}'_{\mathsf{wCom}}$ is given 1^κ and (pub, com^*, r_γ), where r_0 is real and r_1 is random. It then generates $(ek_{\mathsf{TBE}}, dk_{\mathsf{TBE}}) \leftarrow \mathsf{Gen}_{\mathsf{TBE}}(1^\kappa)$ and runs $\mathcal{A}$ on input $ek := (ek_{\mathsf{TBE}}, pub)$.

2. $\mathcal{A}'_{\mathsf{wCom}}$ simulates the challenge ciphertext on input m from $\mathcal{A}$ as follows: It computes $c^* \leftarrow \mathsf{Rnd}_{\mathsf{TBE}}(ek_{\mathsf{TBE}}, 0^{|m|+|dec^*|}; \rho^*_{\mathsf{TBE}})$. It also computes $\sigma^* \leftarrow \mathsf{T}(r_\gamma, c^*)$. It also generates randomness ρ^*_{wCom} and ρ^*_{MAC} by using $\mathsf{Expl}_{\mathsf{wCom}}$ and $\mathsf{Expl}_{\mathsf{MAC}}$. It sends $ct^* = (com^*, c^*, \sigma^*)$ and $\rho^* = (\rho^*_{\mathsf{wCom}}, \rho^*_\gamma, \rho^*_{\mathsf{MAC}})$ to $\mathcal{A}$.
3. $\mathcal{A}'_{\mathsf{wCom}}$ simulates the decryption oracle as follows: Upon receiving $ct = (com, c, \sigma)$, if $com = com^*$, then it returns $\perp$; otherwise, that is, if $com \neq com^*$, it decrypts c into $(m', dec') \leftarrow \mathsf{Dec}_{\mathsf{TBE}}(dk_{\mathsf{TBE}}, com, c)$. If the result is $\perp$, then it return $\perp$; Otherwise, it computes $r' \leftarrow \mathsf{R}(pub, com, dec')$. If $r' = \perp$, then it returns $\perp$. Otherwise, it computes $d \leftarrow \mathsf{V}(r', c, \sigma)$. If $d = \perp$, then it returns $\perp$; Otherwise, it returns m'.
4. Eventually, $\mathcal{A}$ outputs its guess b' and halts. $\mathcal{A}'_{\mathsf{wCom}}$ outputs b' as a guess of γ and halts.

If $\gamma = 0$, then $\mathcal{A}'_{\mathsf{wCom}}$ perfectly simulates Game$_3$. If $\gamma = 1$, then $\mathcal{A}'_{\mathsf{wCom}}$ perfectly simulates Game$_4$. Thus, the lemma follows. □

Lemma 3.8. *There exists a PPT adversary $\mathcal{A}''_{\mathsf{wCom}}$ satisfying*

$$\left|\Pr[\mathsf{Forge}_3] - \Pr[\mathsf{Forge}_4]\right| \leq \mathsf{Adv}^{\mathrm{hiding}}_{\mathsf{wCom}, \mathcal{A}''_{\mathsf{wCom}}}(\kappa).$$

Proof. We construct a PPT adversary $\mathcal{A}''_{\mathsf{wCom}}$ as follows:

1. $\mathcal{A}''_{\mathsf{wCom}}$ is given 1^κ and (pub, com^*, r_γ), where r_0 is real and r_1 is random. It then generates $(ek_{\mathsf{TBE}}, dk_{\mathsf{TBE}}) \leftarrow \mathsf{Gen}_{\mathsf{TBE}}(1^\kappa)$ and runs $\mathcal{A}$ on input $ek := (ek_{\mathsf{TBE}}, pub)$.
2. $\mathcal{A}''_{\mathsf{wCom}}$ simulates the challenge ciphertext on input $m$as follows: It first computes $c^* \leftarrow \mathsf{Rnd}_{\mathsf{TBE}}(ek_{\mathsf{TBE}}, 0^{|m|+|dec^*|}; \rho^*_{\mathsf{TBE}})$. It also computes $\sigma^* \leftarrow \mathsf{T}(r_\gamma, c^*)$. It also generates randomness ρ^*_{wCom} and ρ^*_{MAC} by using $\mathsf{Expl}_{\mathsf{wCom}}$ and $\mathsf{Expl}_{\mathsf{MAC}}$. It sends $ct^* = (com^*, c^*, \sigma^*)$ and $\rho^* = (\rho^*_{\mathsf{wCom}}, \rho^*_{\mathsf{TBE}}, \rho^*_{\mathsf{MAC}})$ to $\mathcal{A}$.
3. $\mathcal{A}''_{\mathsf{wCom}}$ simulates the decryption oracle as follows: Upon receiving $ct = (com, c, \sigma)$, if $com = com^*$, then it returns $\perp$; in addition, if $\mathsf{V}(r_\gamma, c, \sigma) = \top$, then it outputs 1 and halts. If $com \neq com^*$, it decrypts c into $(m', dec') \leftarrow \mathsf{Dec}_{\mathsf{TBE}}(dk_{\mathsf{TBE}}, com, c)$. If the result is $\perp$, then it return $\perp$; Otherwise, it computes $r' \leftarrow \mathsf{R}(pub, com, dec')$. If $r' = \perp$, then it returns $\perp$. Otherwise, it computes $d \leftarrow \mathsf{V}(r', c, \sigma)$. If $d = \perp$, then it returns $\perp$; Otherwise, it returns m'.
4. Eventually, $\mathcal{A}$ outputs its guess b' and halts. $\mathcal{A}''_{\mathsf{wCom}}$ outputs 0 and halts.

If $\gamma = 0$, then $\mathcal{A}''_{\mathsf{wCom}}$ perfectly simulates Game$_3$. If $\gamma = 1$, then $\mathcal{A}''_{\mathsf{wCom}}$ perfectly simulates Game$_4$. Moreover, once $\mathcal{A}$ makes Forge true, then $\mathcal{A}''_{\mathsf{wCom}}$ outputs 1 and halts. Thus, the lemma follows. □

Lemma 3.9. *There exists a PPT adversary $\mathcal{A}_{\mathsf{MAC}}$ satisfying*

$$\Pr[\mathsf{Forge}_4] \leq Q \cdot \mathsf{Adv}^{\mathrm{seuf\text{-}ot\text{-}cma}}_{\mathsf{MAC}, \mathcal{A}_{\mathsf{MAC}}}(\kappa).$$

Proof. We construct a PPT adversary $\mathcal{A}_{\mathsf{MAC}}$ as follows:

1. $\mathcal{A}_{\mathsf{MAC}}$ is given 1^κ. It chooses a random index $i^* \leftarrow \{1, \ldots, Q\}$. It then generates $pub \leftarrow \mathsf{Init}(1^\kappa)$ and $(r^*, com^*, dec^*) \leftarrow \mathsf{S}(pub)$. It then generates $(ek_{\mathsf{TBE}}, dk_{\mathsf{TBE}}) \leftarrow \mathsf{Gen}_{\mathsf{TBE}}(1^\kappa)$ and runs $\mathcal{A}$ on input $ek := (ek_{\mathsf{TBE}}, pub)$.
2. $\mathcal{A}_{\mathsf{MAC}}$ simulates the decryption oracle on a query $ct = (com, c, \sigma)$ as follows: If it receives the i^*-th decryption query, then it outputs (c, σ) as a forgery and halts. Otherwise, if $com = com^*$, then it returns $\perp$. If $com \neq com^*$, it decrypts c into $(m', dec') \leftarrow \mathsf{Dec}_{\mathsf{TBE}}(dk_{\mathsf{TBE}}, com, c)$. If the result is $\perp$, then it return $\perp$; Otherwise, it computes $r' \leftarrow \mathsf{R}(pub, com, dec')$. If $r' = \perp$, then it returns $\perp$. Otherwise, it computes $d \leftarrow \mathsf{V}(r', c, \sigma)$. If $d = \perp$, then it returns $\perp$; Otherwise, it returns m'.
3. $\mathcal{A}_{\mathsf{MAC}}$ simulates the challenge ciphertext on input m from $\mathcal{A}$ as follows: It computes $c^* \leftarrow \mathsf{Rnd}_{\mathsf{TBE}}(ek_{\mathsf{TBE}}, 0^{|m|+|dec^*|}; \rho^*_{\mathsf{TBE}})$. It then query c^* to its tagging oracle and receives $\sigma^* \leftarrow \mathsf{T}(r^+, c^*)$, where $r^* \leftarrow \{0,1\}^\kappa$. It also generates randomness ρ^*_{wCom} and ρ^*_{MAC} by using $\mathsf{Expl}_{\mathsf{wCom}}$ and $\mathsf{Expl}_{\mathsf{MAC}}$. It sends $ct^* = (com^*, c^*, \sigma^*)$ and $\rho^* = (\rho^*_{\mathsf{wCom}}, \rho^*_{\mathsf{TBE}}, \rho^*_{\mathsf{MAC}})$ to $\mathcal{A}$.

$\mathcal{A}_{\mathsf{MAC}}$ perfectly simulates Game$_4$ until Forge_4 occurs. Since i^* is chosen uniformly at random, the success probability that $\mathcal{A}_{\mathsf{MAC}}$ forges is at least $\Pr[\mathsf{Forge}_4]/Q$. Thus, the lemma follows. □

Game$_5$*:* We modify Game$_4$ as follows: In this game, the challenge ciphertext is

$$ct^* = (com^*, c^*, \sigma^*) = \left(com^*, \mathsf{Rnd}_{\mathsf{TBE}}(ek_{\mathsf{TBE}}, 0^{|m|+|dec^*|}; \rho^*_{\mathsf{TBE}}), \mathsf{Rnd}_{\mathsf{MAC}}(1^\kappa; \rho^*_{\mathsf{MAC}})\right),$$

$$\rho^* = (\rho^*_{\mathsf{wCom}}, \rho^*_{\mathsf{TBE}}, \rho^*_{\mathsf{MAC}}) = \left(\mathsf{Expl}_{\mathsf{wCom}}(com^*), \rho^*_{\mathsf{TBE}}, \rho^*_{\mathsf{MAC}}\right)$$

Lemma 3.10. *There exists a PPT adversary $\mathcal{A}'_{\mathsf{MAC}}$ satisfying*

$$|\Pr[S_4] - \Pr[S_5]| \leq \mathsf{Adv}^{\mathrm{os}}_{\mathsf{MAC}, \mathcal{A}'_{\mathsf{MAC}}}(\kappa).$$

Proof. We construct a PPT adversary $\mathcal{A}'_{\mathsf{MAC}}$ as follows:

1. $\mathcal{A}'_{\mathsf{MAC}}$ is given 1^κ. It then generates $pub \leftarrow \mathsf{Init}(1^\kappa)$ and $(r^*, com^*, dec^*) \leftarrow \mathsf{S}(pub)$. It then generates $(ek_{\mathsf{TBE}}, dk_{\mathsf{TBE}}) \leftarrow \mathsf{Gen}_{\mathsf{TBE}}(1^\kappa)$ and runs $\mathcal{A}$ on input $ek := (ek_{\mathsf{TBE}}, pub)$.
2. $\mathcal{A}'_{\mathsf{MAC}}$ simulates the decryption oracle on a query $ct = (com, c, \sigma)$ as follows: If $com = com^*$, then it returns $\perp$. If $com \neq com^*$, it decrypts c into $(m', dec') \leftarrow \mathsf{Dec}_{\mathsf{TBE}}(dk_{\mathsf{TBE}}, com, c)$. If the result is $\perp$, then it return $\perp$; otherwise, it computes $r' \leftarrow \mathsf{R}(pub, com, dec')$. If $r' = \perp$, then it returns $\perp$; otherwise, it computes $d \leftarrow \mathsf{V}(r', c, \sigma)$. If $d = \perp$, then it returns $\perp$; otherwise, it returns m'.
3. $\mathcal{A}'_{\mathsf{MAC}}$ simulates the challenge ciphertext on input m from $\mathcal{A}$ as follows: It computes $c^* \leftarrow \mathsf{Rnd}_{\mathsf{TBE}}(ek_{\mathsf{TBE}}, 0^{|m|+|dec^*|}; \rho^*_{\mathsf{TBE}})$. It then query c^* to its tagging oracle and receives σ_γ and ρ_γ, where $\sigma_0 \leftarrow \mathsf{T}(r^+, c^*)$ with random $r^+ \leftarrow \{0,1\}^\kappa$, $\rho_0 \leftarrow \mathsf{Expl}_{\mathsf{MAC}}(1^\kappa, \sigma_0)$, and $\sigma_1 \leftarrow \mathsf{Rnd}_{\mathsf{MAC}}(1^\kappa; \rho_1)$. It also generates randomness ρ^*_{wCom} by using $\mathsf{Expl}_{\mathsf{wCom}}$. It sends $ct^* = (com^*, c^*, \sigma_\gamma)$ and $\rho^* = (\rho^*_{\mathsf{wCom}}, \rho^*_\gamma, \rho_\gamma)$ to $\mathcal{A}$.
4. Eventually, $\mathcal{A}$ outputs its guess b' and halts. $\mathcal{A}'_{\mathsf{MAC}}$ outputs b' as a guess of γ and halts.

If $\gamma = 0$, then $\mathcal{A}'_{\mathsf{MAC}}$ perfectly simulates Game_4. If $\gamma = 1$, then $\mathcal{A}'_{\mathsf{MAC}}$ perfectly simulates Game_5. Thus, the lemma follows. □

Game_6*:* We modify Game_5 as follows: In this game, the challenge ciphertext is

$$ct^* = (com^*, c^*, \sigma^*) = \Big(\mathsf{Rnd}_{\mathsf{wCom}}(pub; \rho^*_{\mathsf{wCom}}), \mathsf{Rnd}_{\mathsf{TBE}}(ek_{\mathsf{TBE}}, 0^{|m|+|dec^*|}; \rho^*_{\mathsf{TBE}}), \mathsf{Rnd}_{\mathsf{MAC}}(1^\kappa; \rho^*_{\mathsf{MAC}})\Big),$$
$$\rho^* = (\rho^*_{\mathsf{wCom}}, \rho^*_{\mathsf{TBE}}, \rho^*_{\mathsf{MAC}}).$$

Lemma 3.11. *There exists a PPT adversary $\mathcal{A}'''_{\mathsf{wCom}}$ satisfying* $|\Pr[S_5] - \Pr[S_6]| \leq \mathsf{Adv}^{\mathrm{os}}_{\mathsf{wCom}, \mathcal{A}'''_{\mathsf{wCom}}}(\kappa)$.

Proof. We construct a PPT adversary $\mathcal{A}'''_{\mathsf{wCom}}$ as follows:

1. $\mathcal{A}'''_{\mathsf{wCom}}$ is given 1^κ, pub, and $(com_\gamma, \rho_\gamma)$, where the challenger computes $pub \leftarrow \mathsf{Init}(1^\kappa)$, $(r_0, com_0, dec_0) \leftarrow \mathsf{S}(1^\kappa, pub)$, $\rho_0 \leftarrow \mathsf{Expl}_{\mathsf{wCom}}(1^\kappa, com_0)$, and $com_1 \leftarrow \mathsf{Rnd}_{\mathsf{wCom}}(pub; \rho_1)$. It then generates $(ek_{\mathsf{TBE}}, dk_{\mathsf{TBE}}) \leftarrow \mathsf{Gen}_{\mathsf{TBE}}(1^\kappa)$ and runs $\mathcal{A}$ on input $ek := (ek_{\mathsf{TBE}}, pub)$.
2. $\mathcal{A}'''_{\mathsf{wCom}}$ simulates the challenge ciphertext on input m from $\mathcal{A}$ as follows: It computes $c^* \leftarrow \mathsf{Rnd}_{\mathsf{TBE}}(ek_{\mathsf{TBE}}, 0^{|m|+|dec^*|}; \rho^*_{\mathsf{TBE}})$ and $\sigma^* \leftarrow \mathsf{Rnd}_{\mathsf{MAC}}(1^\kappa; \rho^*_{\mathsf{MAC}})$. It sends $ct^* = (com_\gamma, c^*, \sigma^*)$ and $\rho^* = (\rho_\gamma, \rho^*_{\mathsf{TBE}}, \rho^*_{\mathsf{MAC}})$ to $\mathcal{A}$.
3. $\mathcal{A}'''_{\mathsf{wCom}}$ simulates the decryption oracle on a query $ct = (com, c, \sigma)$ as follows: If $com = com^*$, then returns $\perp$. Otherwise, it decrypts c into $(m', dec') \leftarrow \mathsf{Dec}_{\mathsf{TBE}}(dk_{\mathsf{TBE}}, com, c)$. If the result is $\perp$, then it return $\perp$; Otherwise, it computes $r' \leftarrow \mathsf{R}(pub, com, dec')$. If $r' = \perp$, then it returns $\perp$. Otherwise, it computes $d \leftarrow \mathsf{V}(r', c, \sigma)$. If $d = \perp$, then it returns $\perp$; Otherwise, it returns m'.
4. Eventually, $\mathcal{A}$ outputs its guess b' and halts. $\mathcal{A}'''_{\mathsf{wCom}}$ outputs b' as a guess of γ and halts.

If $\gamma = 0$, then $\mathcal{A}'''_{\mathsf{wCom}}$ perfectly simulates Game_5. If $\gamma = 1$, then $\mathcal{A}'''_{\mathsf{wCom}}$ perfectly simulates Game_6. Thus, the lemma follows. □

Game_7*:* We modify Game_6 as follows: In this game, the challenger generates $(ek_{\mathsf{TBE}}, dk_{\mathsf{TBE}}) \leftarrow \mathsf{Gen}_{\mathsf{TBE}}(1^\kappa)$, $pub \leftarrow \mathsf{Init}(1^\kappa)$ and runs the adversary with $ek = (ek_{\mathsf{TBE}}, pub)$. It generates $com^* \leftarrow \mathsf{Rnd}_{\mathsf{wCom}}(pub)$ when it generates the challenge ciphertext as in Game_0. The decryption oracle decrypts a query $ct = (com^*, c, \sigma)$ if $(c, \sigma) \neq (c^*, \sigma^*)$ as in Game_0.
By the definition, we have

$$\Pr[S_7] = \Pr[\mathsf{Expt}^{\text{os-cca},1}_{\mathsf{PKE},\mathcal{A}}(\kappa) = 1].$$

We again recall the event Valid that the adversary queries a valid ciphertext $ct = (com^*, c, \sigma)$ with $(c, \sigma) \neq (c^*, \sigma^*)$. Since Game_6 and Game_7 are equivalent until Valid occurs, we have the following lemma:

Lemma 3.12. *We have*

$$|\Pr[S_6] - \Pr[S_7]| \leq \Pr[\mathsf{Valid}_6] = \Pr[\mathsf{Valid}_7].$$

Let us consider what is a valid ciphertext. If (com^*, c, σ) is valid, we have $(m, dec) \leftarrow \mathsf{Dec}_{\mathsf{TBE}}(dk_{\mathsf{TBE}}, com^*, c)$ with $(m, dec) \neq \perp$, $r \leftarrow \mathsf{R}(pub, com^*, dec)$ with $r \neq \perp$, and $\mathsf{V}(r, c, \sigma) = \top$ in decryption.
Let us consider an event Inv as the event that we have $r \neq \perp$ in decryption. Notice that if Valid occurs, then Inv should occur internally. Thus, we have

$$\Pr[\mathsf{Valid}_7] \leq \Pr[\mathsf{Inv}_7].$$

Lemma 3.13. *There exists a PPT adversary $\mathcal{A}''''_{\mathsf{wCom}}$ satisfying*

$$\Pr[\mathsf{Inv}_7] \leq \mathsf{Adv}^{\text{non-inv}}_{\mathsf{wCom},\mathcal{A}''''_{\mathsf{wCom}}}(\kappa).$$

Proof. We construct $\mathcal{A}''''_{\mathsf{wCom}}$ as follows:

1. $\mathcal{A}''''_{\mathsf{wCom}}$ is given $(1^\kappa, pub, com^*, \rho^*)$ from its challenger, where $pub \leftarrow \mathsf{Init}(1^\kappa)$ and $com^* \leftarrow \mathsf{Rnd}_{\mathsf{wCom}}(pub, pub; \rho^*_{\mathsf{wCom}})$. It generates $(ek_{\mathsf{TBE}}, dk_{\mathsf{TBE}}) \leftarrow \mathsf{Gen}_{\mathsf{TBE}}(1^\kappa)$, and runs $\mathcal{A}$ on input $ek := (ek_{\mathsf{TBE}}, pub)$.
2. $\mathcal{A}''''_{\mathsf{wCom}}$ generates the challenge on a query m from $\mathcal{A}$ as follows: It computes $c^* \leftarrow \mathsf{Rnd}_{\mathsf{TBE}}(ek_{\mathsf{TBE}}, 0^{|m|+|dec^*|}; \rho^*_{\mathsf{TBE}})$ and $\sigma^* \leftarrow \mathsf{Rnd}_{\mathsf{MAC}}(1^\kappa; \rho^*_{\mathsf{MAC}})$. sends $ct^* = (com^*, c^*, \sigma^*)$ and $\rho^* = (\rho^*_{\mathsf{wCom}}, \rho^*_{\mathsf{TBE}}, \rho^*_{\mathsf{MAC}})$ to $\mathcal{A}$.
3. $\mathcal{A}''''_{\mathsf{wCom}}$ simulates the decryption oracle in Game_7 by using its decryption key dk_{TBE} as follows: If $ct = ct^*$, then return $\perp$. Otherwise, it obtains $(m', dec') \leftarrow \mathsf{Dec}_{\mathsf{TBE}}(dk_{\mathsf{TBE}}, com, c)$, $r \leftarrow \mathsf{R}(pub, com, dec)$, and outputs m' if $(m', dec') \neq \perp$, $r \neq \perp$, and $\mathsf{V}(r, c, \sigma) = \top$. Once $\mathcal{A}''''_{\mathsf{wCom}}$ detects Inv, that is, on the query (com^*, c, σ), it obtains $(m', dec') \leftarrow \mathsf{Dec}_{\mathsf{TBE}}(dk_{\mathsf{TBE}}, com^*, c)$ and $r' \leftarrow \mathsf{R}(pub, com^*, dec')$ with $r' \neq \perp$, then $\mathcal{A}''''_{\mathsf{wCom}}$ outputs dec' and halts.

Since the simulation of Game_7 is perfect, $\mathcal{A}$ correctly works. Once Inv occurs, $\mathcal{A}''''_{\mathsf{wCom}}$ breaks the non-invertible property. Thus, the lemma holds. □

Summary: Summing up the bounds in following lemmas, we obtain Theorem 3.1.

$$\begin{aligned}
\mathsf{Adv}^{\text{os-cca}}_{\mathsf{PKE}}(\kappa) &= \left|\Pr[\mathsf{Expt}^{\text{os-cca},0}_{\mathsf{PKE},\mathcal{A}}(\kappa) = 1] - \Pr[\mathsf{Expt}^{\text{os-cca},1}_{\mathsf{PKE},\mathcal{A}}(\kappa) = 1]\right| \\
&= \left|\Pr[S_0] - \Pr[S_7]\right| \leq \sum_{i=0}^{6}\left|\Pr[S_i] - \Pr[S_{i+1}]\right| \\
&\leq 0 + \Pr[\mathsf{Valid}_2] + \mathsf{Adv}^{\text{os-st-wcca}}_{\mathsf{TBE},\mathcal{A}_{\mathsf{TBE}}}(\kappa) + \mathsf{Adv}^{\text{hiding}}_{\mathsf{wCom},\mathcal{A}'_{\mathsf{wCom}}}(\kappa) \\
&\qquad + \mathsf{Adv}^{\text{os}}_{\mathsf{MAC},\mathcal{A}'_{\mathsf{MAC}}}(\kappa) + \mathsf{Adv}^{\text{os}}_{\mathsf{wCom},\mathcal{A}'''_{\mathsf{wCom}}}(\kappa) + \Pr[\mathsf{Valid}_7] \\
&\leq \Pr[\mathsf{NoBind}_2] + \Pr[\mathsf{Forge}_2] \\
&\qquad + \Pr[\mathsf{Inv}_7] \\
&\qquad + \mathsf{Adv}^{\text{os-st-wcca}}_{\mathsf{TBE},\mathcal{A}_{\mathsf{TBE}}}(\kappa) + \mathsf{Adv}^{\text{hiding}}_{\mathsf{wCom},\mathcal{A}'_{\mathsf{wCom}}}(\kappa) + \mathsf{Adv}^{\text{os}}_{\mathsf{MAC},\mathcal{A}'_{\mathsf{MAC}}}(\kappa) + \mathsf{Adv}^{\text{os}}_{\mathsf{wCom},\mathcal{A}'''_{\mathsf{wCom}}}(\kappa) \\
&\leq +\mathsf{Adv}^{\text{binding}}_{\mathsf{wCom},\mathcal{A}_{\mathsf{wCom}}}(\kappa) + \left|\Pr[\mathsf{Forge}_2] - \Pr[\mathsf{Forge}_3]\right| + \left|\Pr[\mathsf{Forge}_3] - \Pr[\mathsf{Forge}_4]\right| + \Pr[\mathsf{Forge}_4] \\
&\qquad + \mathsf{Adv}^{\text{non-inv}}_{\mathsf{wCom},\mathcal{A}''''_{\mathsf{wCom}}}(\kappa) \\
&\qquad + \mathsf{Adv}^{\text{os-st-wcca}}_{\mathsf{TBE},\mathcal{A}_{\mathsf{TBE}}}(\kappa) + \mathsf{Adv}^{\text{hiding}}_{\mathsf{wCom},\mathcal{A}'_{\mathsf{wCom}}}(\kappa) + \mathsf{Adv}^{\text{os}}_{\mathsf{MAC},\mathcal{A}'_{\mathsf{MAC}}}(\kappa) + \mathsf{Adv}^{\text{os}}_{\mathsf{wCom},\mathcal{A}'''_{\mathsf{wCom}}}(\kappa) \\
&\leq +\mathsf{Adv}^{\text{binding}}_{\mathsf{wCom},\mathcal{A}_{\mathsf{wCom}}}(\kappa) + \mathsf{Adv}^{\text{os-st-wcca}}_{\mathsf{TBE},\mathcal{A}'_{\mathsf{TBE}}}(\kappa) + \mathsf{Adv}^{\text{hiding}}_{\mathsf{wCom},\mathcal{A}''_{\mathsf{wCom}}}(\kappa) + Q \cdot \mathsf{Adv}^{\text{seuf-ot-cma}}_{\mathsf{MAC},\mathcal{A}_{\mathsf{MAC}}}(\kappa) \\
&\qquad + \mathsf{Adv}^{\text{non-inv}}_{\mathsf{wCom},\mathcal{A}''''_{\mathsf{wCom}}}(\kappa) + \mathsf{Adv}^{\text{os-st-wcca}}_{\mathsf{TBE},\mathcal{A}_{\mathsf{TBE}}}(\kappa) + \mathsf{Adv}^{\text{hiding}}_{\mathsf{wCom},\mathcal{A}'_{\mathsf{wCom}}}(\kappa) \\
&\qquad + \mathsf{Adv}^{\text{os}}_{\mathsf{MAC},\mathcal{A}'_{\mathsf{MAC}}}(\kappa) + \mathsf{Adv}^{\text{os}}_{\mathsf{wCom},\mathcal{A}'''_{\mathsf{wCom}}}(\kappa).
\end{aligned}$$

4 Instantiations and Applications

Instantiations: We have several lattice/code-based IBE/TBE schemes allowing us to construct OS-CCA/PR-CCA-secure PKE schemes by combining them with appropriate commitment scheme and MAC scheme from symmetric-key primitives.

From Lattices: The CHKP IBE scheme [CHKP12], the ABB IBE scheme [ABB10], and the MP TBE scheme [MP12] (and its variant the BBDQ TBE scheme [BBDQ18]) from lattices are PR-ST-wCCA-secure under the LWE assumptions with suitable parameter settings. Moreover, their ciphertext spaces are of the form $\mathbb{Z}_q^k$ for positive integers q and k and, thus, the ciphertext spaces are ESE.

From Codes: The DMQN09 TBE scheme [DMN09] and the DMQN12 TBE scheme [DMN12] are also PR-ST-wCCA-secure under the assumption that their keys are pseudorandom and the LPN assumptions. Their ciphertext spaces are of the form $\mathbb{F}_2^k$ for positive integer k and, thus, the ciphertext spaces are ESE.
The KMP TBE scheme [KMP14] and the YZ TBE scheme [YZ16] are IND-ST-wCCA-secure under the assumption that the low-noise LPN problem is hard and the assumption that the constant-noise LPN problem is sub-exponentially hard, respectively. Fortunately, we can show that they are PR-ST-wCCA-secure under the same assumptions. See section B and section C for the details.

Fully-equipped, UC-secure bit commitment: Canetti and Fischlin [CF01] constructed a UC-secure non-interactive bit commitment for adaptive corruption without erasures in the re-usable CRS model from trapdoor commitment (as known as chameleon hash function [KR00]) and OS-CCA-secure PKE.
We have a trapdoor commitment scheme from lattices [CHKP12]. Combining it with OS-CCA-secure PKE scheme from lattice, we obtain fully-equipped, UC-secure bit commitment under the LWE assumption.
Unfortunately, we do not know any non-interactive trapdoor commitment scheme from codes/LPN and this is a long-standing open problem. The construction of fully-equipped UC-secure commitment from codes/LPN is still an open problem, although we have *interactive* UC-secure commitment from LPN, for example, one obtained by combining UC-secure commitment in the OT-hybrid model [CDD+16] and 2-round OT from LPN [DGH+20].

Public-key steganography: Hopper [Hop05] also studied it and gave a construction of public-key steganography secure against adaptive chosen-covertext attacks (SS-CCA-security) against a single channel from SPR-CCA-secure PKE [Hop05]. Berndt and Liśkiewicz [BL18] improved the constructions to achieve SS-CCA-secure public-key steganography against every memoryless channel from SPR-CCA-secure PKE, PRPs, and CRHFs.
Since we have SPR-CCA-secure PKE from lattices and codes, we obtain SS-CCA-secure public-key steganography from lattices and codes through [Hop05, BL18].

Anonymous AKE: KEM-based AKEs [BCGNP09, FSXY13, FSXY15, SSW20] have a chance to get anonymity of AKE. Such AKEs employ IND-CCA-secure KEM and IND-CPA-secure KEM. Roughly speaking, the first message from Alice is $pk_{\mathrm{tmp}}, ct_{A \to B} = \mathsf{Enc}_{\mathrm{cca}}(pk_B)$ and the second message from Bob is $ct_{\mathrm{tmp}} = \mathsf{Enc}_{\mathrm{cpa}}(pk_{\mathrm{tmp}}), ct_{B \to A} = \mathsf{Enc}_{\mathrm{cca}}(pk_A)$. Thus, if the ciphertexts of IND-CCA-secure KEM are pseudorandom, then the AKE is anonymous from the outsider's view.

SIM-SSO-CCA PKE: Following and repairing Fehr, Hofheinz, Kiltz, and Wee [FHKW10], Liu and Paterson [LP15] constructed a SIM-SSO-CCA secure PKE scheme using a special KEM scheme, which they call "tailored" KEM; roughly speaking, they required the following properties: 1) *ESE domains:* the key space and ciphertext space are efficiently samplable and explainable (ESE), 2) *tailored decapsulation:* the valid ciphertexts should be a small subset of ciphertext space, and 3) *tailored security:* it should satisfy tailored, constrained CCA security, which is weaker than IND-CCA security.
It is easy to convert OS-CCA-secure PKE scheme into OS-CCA-secure KEM scheme if the message space is ESE; choosing a key $K \leftarrow \mathcal{M}$ and encrypting it as $C = \mathsf{Enc}_{\mathsf{PKE}}(ek, K; \rho)$. We note that the OS-CCA-secure PKE scheme obtained by the BK transformation satisfies the tailored decapsulation since its ciphertext contains a MAC tag. Thus, following [LP15], OS-CCA-secure PKE (with an ESE key space) implies SIM-SSO-CCA secure PKE. Instantiating OS-CCA-secure from lattices and codes, we obtain SIM-SSO-CCA-secure PKEs in the standard model from lattice and codes, respectively.

References

ABB10. Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Gilbert [Gil10], pages 553–572. 2, 15

AFS05. Daniel Augot, Matthieu Finiasz, and Nicolas Sendrier. A family of fast syndrome based cryptographic hash functions. In Ed Dawson and Serge Vaudenay, editors, *Progress in Cryptology - Mycrypt 2005, First International Conference on Cryptology in Malaysia, Kuala Lumpur, Malaysia, September 28-30, 2005, Proceedings*, volume 3715 of *Lecture Notes in Computer Science*, pages 64–83. Springer, 2005. 8

Ajt96. Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th ACM STOC*, pages 99–108. ACM Press, May 1996. 8

AP12. Jacob Alperin-Sheriff and Chris Peikert. Circular and KDM security for identity-based encryption. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 334–352. Springer, Heidelberg, May 2012. 21

BBDP01. Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 566–582. Springer, Heidelberg, December 2001. 1, 3

BBDQ18. Fabrice Benhamouda, Olivier Blazy, Léo Ducas, and Willy Quach. Hash proof systems over lattices revisited. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 644–674. Springer, Heidelberg, March 2018. 2, 15

BC05. Michael Backes and Christian Cachin. Public-key steganography with active attacks. In Joe Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 210–226. Springer, Heidelberg, February 2005. 3

BCGNP09. Colin Boyd, Yvonne Cliff, Juan Manuel González Nieto, and Kenneth G. Paterson. One-round key exchange in the standard model. *Int. J. Appl. Cryptogr.*, 1(3):181–199, 2009. 3, 16

BCHK07. Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. *SIAM J. Comput.*, 36(5):1301–1328, 2007. 2, 6, 7, 8, 9, 10

BDPR98. Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In Krawczyk [Kra98], pages 26–45. 5

BFKL94. Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 278–291. Springer, Heidelberg, August 1994. 21

BK05. Dan Boneh and Jonathan Katz. Improved efficiency for CCA-secure cryptosystems built using identity-based encryption. In Alfred Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 87–103. Springer, Heidelberg, February 2005. 2, 7

BL17. Xavier Boyen and Qinyi Li. All-but-many lossy trapdoor functions from lattices and applications. In Katz and Shacham [KS17], pages 298–331. 4

BL18. Sebastian Berndt and Maciej Liskiewicz. On the gold standard for security of universal steganography. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 29–60. Springer, Heidelberg, April / May 2018. 2, 3, 16

Blu81. Manuel Blum. Coin flipping by telephone. In Allen Gersho, editor, *CRYPTO'81*, volume ECE Report 82-04, pages 11–15. U.C. Santa Barbara, Dept. of Elec. and Computer Eng., 1981. 6

BLVW19. Zvika Brakerski, Vadim Lyubashevsky, Vinod Vaikuntanathan, and Daniel Wichs. Worst-case hardness for LPN and cryptographic hashing via code smoothing. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 619–635. Springer, Heidelberg, May 2019. 8

CDD+16. Ignacio Cascudo, Ivan Damgård, Bernardo David, Nico Döttling, and Jesper Buus Nielsen. Rate-1, linear time and additively homomorphic UC commitments. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 179–207. Springer, Heidelberg, August 2016. 16

CF01. Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 19–40. Springer, Heidelberg, August 2001. 2, 3, 4, 5, 16

CHKP12. David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. *Journal of Cryptology*, 25(4):601–639, October 2012. 2, 15, 16

CL01. Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, Heidelberg, May 2001. 1, 3

CS98. Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Krawczyk [Kra98], pages 13–25. 3

DGG+15. Yevgeniy Dodis, Chaya Ganesh, Alexander Golovnev, Ari Juels, and Thomas Ristenpart. A formal treatment of backdoored pseudorandom generators. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 101–126. Springer, Heidelberg, April 2015. 1

DGH+20. Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, Daniel Masny, and Daniel Wichs. Two-round oblivious transfer from CDH or LPN. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 768–797. Springer, Heidelberg, May 2020. 16

DGK06. Mario Di Raimondo, Rosario Gennaro, and Hugo Krawczyk. Deniable authentication and key exchange. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 400–409. ACM Press, October / November 2006. 3

DMN09. Rafael Dowsley, Jörn Müller-Quade, and Anderson C. A. Nascimento. A CCA2 secure public key encryption scheme based on the McEliece assumptions in the standard model. In Marc Fischlin, editor, *CT-RSA 2009*, volume 5473 of *LNCS*, pages 240–251. Springer, Heidelberg, April 2009. 2, 16

DMN12. Nico Döttling, Jörn Müller-Quade, and Anderson C. A. Nascimento. IND-CCA secure cryptography based on a variant of the LPN problem. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 485–503. Springer, Heidelberg, December 2012. 2, 16

DN00. Ivan Damgård and Jesper Buus Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 432–450. Springer, Heidelberg, August 2000. 4

DvW92. Whitfield Diffie, Paul C. van Oorschot, and Michael J. Wiener. Authentication and authenticated key exchanges. *Des. Codes Cryptogr.*, 2(2):107–125, 1992. 3

FHKW10. Serge Fehr, Dennis Hofheinz, Eike Kiltz, and Hoeteck Wee. Encryption schemes secure against chosen-ciphertext selective opening attacks. In Gilbert [Gil10], pages 381–402. 2, 4, 16

FSXY13. Atsushi Fujioka, Koutarou Suzuki, Keita Xagawa, and Kazuki Yoneyama. Practical and post-quantum authenticated key exchange from one-way secure key encapsulation mechanism. In Kefei Chen, Qi Xie, Weidong Qiu, Ninghui Li, and Wen-Guey Tzeng, editors, *ASIACCS 13*, pages 83–94. ACM Press, May 2013. 3, 16

FSXY15. Atsushi Fujioka, Koutarou Suzuki, Keita Xagawa, and Kazuki Yoneyama. Strongly secure authenticated key exchange from factoring, codes, and lattices. *Des. Codes Cryptogr.*, 76(3):469–504, 2015. 2, 3, 16

GGH96. Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Collision-free hashing from lattice problems. Cryptology ePrint Archive, Report 1996/009, 1996. https://eprint.iacr.org/1996/009. 8

Gil10. Henri Gilbert, editor. *EUROCRYPT 2010*, volume 6110 of *LNCS*. Springer, Heidelberg, May / June 2010. 16, 18

GM84. Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984. 1

HC98. Dan Harkins and Dave Carrel. The Internet Key Exchange (IKE). IETF RFC 2409 (Proposed Standard), 1998. 3

HJKS15. Felix Heuer, Tibor Jager, Eike Kiltz, and Sven Schäge. On the selective opening security of practical public-key encryption schemes. In Katz [Kat15], pages 27–51. 3

HKKP21. Keitaro Hashimoto, Shuichi Katsumata, Kris Kwiatkowski, and Thomas Prest. An efficient and generic construction for signal's handshake (X3DH): post-quantum, state leakage secure, and deniable. In Juan A. Garay, editor, *Public-Key Cryptography - PKC 2021 - 24th IACR International Conference on Practice and Theory of Public Key Cryptography, Virtual Event, May 10-13, 2021, Proceedings, Part II*, volume 12711 of *Lecture Notes in Computer Science*, pages 410–440. Springer, 2021. 3

HLOV11. Brett Hemenway, Benoît Libert, Rafail Ostrovsky, and Damien Vergnaud. Lossy encryption: Constructions from general assumptions and efficient selective opening chosen ciphertext security. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 70–88. Springer, Heidelberg, December 2011. 4

HLQ13. Zhengan Huang, Shengli Liu, and Baodong Qin. Sender-equivocable encryption schemes secure against chosen-ciphertext attacks revisited. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 369–385. Springer, Heidelberg, February / March 2013. 4

HLQC13. Zhengan Huang, Shengli Liu, Baodong Qin, and Kefei Chen. Fixing the sender-equivocable encryption scheme in Eurocrypt 2010. In *2013 5th International Conference on Intelligent Networking and Collaborative Systems, Xi'an city, Shaanxi province, China, September 9-11, 2013*, pages 366–372. IEEE, 2013. 4

Hof12. Dennis Hofheinz. All-but-many lossy trapdoor functions. In Pointcheval and Johansson [PJ12], pages 209–227. 4

Hop05. Nicholas Hopper. On steganographic chosen covertext security. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP 2005*, volume 3580 of *LNCS*, pages 311–323. Springer, Heidelberg, July 2005. 1, 2, 3, 5, 16

HP16. Felix Heuer and Bertram Poettering. Selective opening security from simulatable data encapsulation. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 248–277. Springer, Heidelberg, December 2016. 3

HPRV19. Thibaut Horel, Sunoo Park, Silas Richelson, and Vinod Vaikuntanathan. How to subvert backdoored encryption: Security against adversaries that decrypt all ciphertexts. In Avrim Blum, editor, *ITCS 2019*, volume 124, pages 42:1–42:20. LIPIcs, January 2019. 1

IKOS10. Yuval Ishai, Abishek Kumarasubramanian, Claudio Orlandi, and Amit Sahai. On invertible sampling and adaptive security. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 466–482. Springer, Heidelberg, December 2010. 2

Kat15. Jonathan Katz, editor. *PKC 2015*, volume 9020 of *LNCS*. Springer, Heidelberg, March / April 2015. 18, 19

Kil06. Eike Kiltz. Chosen-ciphertext security from tag-based encryption. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 581–600. Springer, Heidelberg, March 2006. 6, 10

KMP14. Eike Kiltz, Daniel Masny, and Krzysztof Pietrzak. Simple chosen-ciphertext security from low-noise LPN. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 1–18. Springer, Heidelberg, March 2014. 2, 16, 21, 22, 23, 24, 25

KR00. Hugo Krawczyk and Tal Rabin. Chameleon signatures. In *NDSS 2000*. The Internet Society, February 2000. 16

Kra98. Hugo Krawczyk, editor. *CRYPTO'98*, volume 1462 of *LNCS*. Springer, Heidelberg, August 1998. 17

KS17. Jonathan Katz and Hovav Shacham, editors. *CRYPTO 2017, Part III*, volume 10403 of *LNCS*. Springer, Heidelberg, August 2017. 17, 19

LLHG18. Lin Lyu, Shengli Liu, Shuai Han, and Dawu Gu. Tightly SIM-SO-CCA secure public key encryption from standard assumptions. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 62–92. Springer, Heidelberg, March 2018. 4

LLW20. Qiqi Lai, Feng-Hao Liu, and Zhedong Wang. Almost tight security in lattices with polynomial moduli - PRF, IBE, all-but-many LTF, and more. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 652–681. Springer, Heidelberg, May 2020. 4

LP15. Shengli Liu and Kenneth G. Paterson. Simulation-based selective opening CCA security for PKE from key encapsulation mechanisms. In Katz [Kat15], pages 3–26. 1, 2, 3, 4, 16

LSSS17. Benoît Libert, Amin Sakzad, Damien Stehlé, and Ron Steinfeld. All-but-many lossy trapdoor functions and selective opening chosen-ciphertext security from LWE. In Katz and Shacham [KS17], pages 332–364. 4

MM11. Daniele Micciancio and Petros Mol. Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 465–484. Springer, Heidelberg, August 2011. 8, 21

Möl04. Bodo Möller. A public-key encryption scheme with pseudo-random ciphertexts. In Pierangela Samarati, Peter Y. A. Ryan, Dieter Gollmann, and Refik Molva, editors, *ESORICS 2004*, volume 3193 of *LNCS*, pages 335–351. Springer, Heidelberg, September 2004. 1

MP12. Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In Pointcheval and Johansson [PJ12], pages 700–718. 2, 15

MRY04. Philip D. MacKenzie, Michael K. Reiter, and Ke Yang. Alternatives to non-malleability: Definitions, constructions, and applications (extended abstract). In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 171–190. Springer, Heidelberg, February 2004. 5

NY89. Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *21st ACM STOC*, pages 33–43. ACM Press, May 1989. 1

PJ12. David Pointcheval and Thomas Johansson, editors. *EUROCRYPT 2012*, volume 7237 of *LNCS*. Springer, Heidelberg, April 2012. 18, 19

PQR21. Jiaxin Pan, Chen Qian, and Magnus Ringerud. Signed diffie-hellman key exchange with tight security. In Kenneth G. Paterson, editor, *Topics in Cryptology - CT-RSA 2021 - Cryptographers' Track at the RSA Conference 2021, Virtual Event, May 17-20, 2021, Proceedings*, volume 12704 of *Lecture Notes in Computer Science*, pages 201–226. Springer, 2021. 3

PS08. Kenneth G. Paterson and Sriramkrishnan Srinivasan. Security and anonymity of identity-based encryption with multiple trusted authorities. In Steven D. Galbraith and Kenneth G. Paterson, editors, *PAIRING 2008*, volume 5209 of *LNCS*, pages 354–375. Springer, Heidelberg, September 2008. 3

PS09. Kenneth G. Paterson and Sriramkrishnan Srinivasan. Building key-private public-key encryption schemes. In Colin Boyd and Juan Manuel González Nieto, editors, *ACISP 09*, volume 5594 of *LNCS*, pages 276–292. Springer, Heidelberg, July 2009. 3

Reg09. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):34:1–34:40, 2009. 8

RS92. Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 433–444. Springer, Heidelberg, August 1992. 1, 5

Sak00. Kazue Sako. An auction protocol which hides bids of losers. In Hideki Imai and Yuliang Zheng, editors, *PKC 2000*, volume 1751 of *LNCS*, pages 422–432. Springer, Heidelberg, January 2000. 1, 3

SS19. Shingo Sato and Junji Shikata. SO-CCA secure PKE in the quantum random oracle model or the quantum ideal cipher model. In Martin Albrecht, editor, *17th IMA International Conference on Cryptography and Coding*, volume 11929 of *LNCS*, pages 317–341. Springer, Heidelberg, December 2019. 3

SSW20. Peter Schwabe, Douglas Stebila, and Thom Wiggers. Post-quantum TLS without handshake signatures. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 20*, pages 1461–1480. ACM Press, November 2020. 3, 16

SW14. Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014. 1, 2, 3

vH04. Luis von Ahn and Nicholas J. Hopper. Public-key steganography. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 323–341. Springer, Heidelberg, May 2004. 1, 3, 5

YZ16. Yu Yu and Jiang Zhang. Cryptography with auxiliary input and trapdoor from constant-noise LPN. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 214–243. Springer, Heidelberg, August 2016. 2, 16, 28, 29, 30, 31, 32

YZW+19. Yu Yu, Jiang Zhang, Jian Weng, Chun Guo, and Xiangxue Li. Collision resistant hashing from sub-exponential learning parity with noise. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part II*, volume 11922 of *LNCS*, pages 3–24. Springer, Heidelberg, December 2019. 8

Zha07. Rui Zhang. Tweaking TBE/IBE to PKE transforms with chameleon hash functions. In Jonathan Katz and Moti Yung, editors, *ACNS 07*, volume 4521 of *LNCS*, pages 323–339. Springer, Heidelberg, June 2007. 2, 4

A Learning Parity with Noise

Ber_p denotes the Bernoulli distribution with parameter $p \in (0, 1/2)$, that is, $\Pr[x = 1 \mid x \leftarrow \mathsf{Ber}_p] = p$ and $\Pr[x = 0 \mid x \leftarrow \mathsf{Ber}_p] = 1 - p$.

Learning Parity with Noise: We review the LPN assumption [BFKL94] and its variations.

LPN: The $\mathsf{LPN}[n, m, p]$ assumption states that for any efficient adversary $\mathcal{A}$ its advantage $\mathsf{Adv}_{\mathsf{LPN}[n,m,p],\mathcal{A}}(\kappa)$ is negligible in κ, where

$$\mathsf{Adv}_{\mathsf{LPN}[n,m,p],\mathcal{A}}(\kappa) := \left| \begin{array}{r} \Pr[\boldsymbol{A} \leftarrow \mathbb{F}_2^{m\times n}, s \leftarrow \mathbb{F}_2^n, \boldsymbol{e} \leftarrow \mathsf{Ber}_p^m : \mathcal{A}(\boldsymbol{A}, \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}) = 1] \\ - \Pr[\boldsymbol{A} \leftarrow \mathbb{F}_2^{m\times n}, \boldsymbol{b} \leftarrow \mathbb{F}_2^m : \mathcal{A}(\boldsymbol{A}, \boldsymbol{b}) = 1] \end{array} \right|.$$

Knapsack LPN: The knapsack LPN distribution is considered in Micciancio and Mol [MM11] as the dual of the LPN distribution. The $\mathsf{KLPN}[n, m, p]^m$ assumption states that for any efficient adversary $\mathcal{A}$ its advantage $\mathsf{Adv}_{\mathsf{KLPN}[n,m,p]^m,\mathcal{A}}(\kappa)$ is negligible in κ, where

$$\mathsf{Adv}_{\mathsf{KLPN}[n,m,p]^m,\mathcal{A}}(\kappa) := \left| \begin{array}{l} \Pr[\boldsymbol{A} \leftarrow \mathbb{F}_2^{m\times(m-n)}, \boldsymbol{E} \leftarrow \mathsf{Ber}_p^{m\times m} : \mathcal{A}(\boldsymbol{A}, \boldsymbol{E}\boldsymbol{A}) = 1] \\ - \Pr[\boldsymbol{A} \leftarrow \mathbb{F}_2^{m\times(m-n)}, \boldsymbol{B} \leftarrow \mathbb{F}_2^{m\times(m-n)} : \mathcal{A}(\boldsymbol{A}, \boldsymbol{B}) = 1] \end{array} \right|.$$

For any algorithm $\mathcal{A}$, there exists an algorithm $\mathcal{A}'$ that runs in roughly the same time as $\mathcal{A}$ and

$$\mathsf{Adv}_{\mathsf{LPN}[n,m,p]^m,\mathcal{A}'}(\kappa) \geq \frac{1}{m}\mathsf{Adv}_{\mathsf{KLPN}[n,m,p]^m,\mathcal{A}'}(\kappa)$$

See [MM11].

Extended Knapsack LPN: The extended knapsack LPN assumption states that for any efficient adversary $\mathcal{A}$ its advantage $\mathsf{Adv}_{\mathsf{EKLPN}[n,m,p]^m,\mathcal{A}}(\kappa)$ is negligible in κ, where

$$\begin{aligned} &\mathsf{Adv}_{\mathsf{EKLPN}[n,m,p]^m,\mathcal{A}}(\kappa) := |p_0 - p_1| \\ &p_0 := \Pr[\boldsymbol{A} \leftarrow \mathbb{F}_2^{m\times(m-n)}, \boldsymbol{E} \leftarrow \mathsf{Ber}_p^{m\times m}, \boldsymbol{z} \leftarrow \mathsf{Ber}_p^m : \mathcal{A}(\boldsymbol{A}, \boldsymbol{E}\boldsymbol{A}, \boldsymbol{z}, \boldsymbol{E}\boldsymbol{z}) = 1] \\ &p_1 := \Pr[\boldsymbol{A} \leftarrow \mathbb{F}_2^{m\times(m-n)}, \boldsymbol{B} \leftarrow \mathbb{F}_2^{m\times(m-n)}, \boldsymbol{E} \leftarrow \mathsf{Ber}_p^{m\times m}, \boldsymbol{z} \leftarrow \mathsf{Ber}_p^m : \mathcal{A}(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{z}, \boldsymbol{E}\boldsymbol{z}) = 1]. \end{aligned}$$

For any algorithm $\mathcal{A}$, there exists an algorithm $\mathcal{A}'$ that runs in roughly the same time as $\mathcal{A}$ and

$$\mathsf{Adv}_{\mathsf{LPN}[n,m,p]^m,\mathcal{A}'}(\kappa) \geq \frac{1}{2m}\mathsf{Adv}_{\mathsf{EKLPN}[n,m,p]^m,\mathcal{A}'}(\kappa)$$

See [AP12, KMP14].

1-Knapsack LPN: We additionally introduce the 1-knapsack LPN assumption, in which we replace the last *column* of $\boldsymbol{E}\boldsymbol{A}$ of the KLPN distribution with a random one. The 1-knapsack LPN assumption states that for any efficient adversary $\mathcal{A}$ its advantage $\mathsf{Adv}_{\mathsf{1KLPN}[n,m,p]^m,\mathcal{A}}(\kappa)$ is negligible in κ, where

$$\begin{aligned} &\mathsf{Adv}_{\mathsf{1KLPN}[n,m,p]^m,\mathcal{A}}(\kappa) := |p_0 - p_1| \\ &p_0 := \Pr[[\boldsymbol{A}, \boldsymbol{c}] \leftarrow \mathbb{F}_2^{m\times(m-n)}, \boldsymbol{E} \leftarrow \mathsf{Ber}_p^{m\times m}, \boldsymbol{u} \leftarrow \mathbb{F}_2^m : \mathcal{A}(\boldsymbol{A}, \boldsymbol{c}, \boldsymbol{E}\boldsymbol{A}, \boldsymbol{u}) = 1] \\ &p_1 := \Pr[[\boldsymbol{A}, \boldsymbol{c}] \leftarrow \mathbb{F}_2^{m\times(m-n)}, \boldsymbol{E} \leftarrow \mathsf{Ber}_p^{m\times m} : \mathcal{A}(\boldsymbol{A}, \boldsymbol{c}, \boldsymbol{E}\boldsymbol{A}, \boldsymbol{E}\boldsymbol{c}) = 1]. \end{aligned}$$

We consider the following intermediate probability:

$$p_u := \Pr[[\boldsymbol{A}, \boldsymbol{c}] \leftarrow \mathbb{F}_2^{m\times(m-n)}, \boldsymbol{E} \leftarrow \mathsf{Ber}_p^{m\times m}, \boldsymbol{U} \leftarrow \mathbb{F}_2^{m\times(n-1)}, \boldsymbol{u} \leftarrow \mathbb{F}_2^m : \mathcal{A}(\boldsymbol{A}, \boldsymbol{c}, \boldsymbol{U}, \boldsymbol{u}) = 1]$$

We have two adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$ such that

$$\begin{aligned} \mathsf{Adv}_{\mathsf{1KLPN}[n,m,p]^m,\mathcal{A}}(\kappa) &= |p_0 - p_1| \leq |p_0 - p_u| + |p_u - p_1| \\ &\leq \mathsf{Adv}_{\mathsf{KLPN}[n-1,m,p]^m,\mathcal{A}_1}(\kappa) + \mathsf{Adv}_{\mathsf{KLPN}[n,m,p]^m,\mathcal{A}_2}(\kappa). \end{aligned}$$

It is easy to see that

$$\mathsf{Adv}_{\mathsf{1KLPN}[n,m,p]^1,\mathcal{A}}(\kappa) = \left| \begin{array}{r} \Pr[[\boldsymbol{A}, \boldsymbol{c}] \leftarrow \mathbb{F}_2^{m\times(m-n)}, \boldsymbol{e} \leftarrow \mathsf{Ber}_p^{1\times m}, u \leftarrow \mathbb{F}_2^1 : \mathcal{A}(\boldsymbol{A}, \boldsymbol{c}, \boldsymbol{e}\boldsymbol{A}, \boldsymbol{u}) = 1] \\ - \Pr[[\boldsymbol{A}, \boldsymbol{c}] \leftarrow \mathbb{F}_2^{m\times(m-n)}, \boldsymbol{e} \leftarrow \mathsf{Ber}_p^{1\times m} : \mathcal{A}(\boldsymbol{A}, \boldsymbol{c}, \boldsymbol{e}\boldsymbol{A}, \boldsymbol{e}\boldsymbol{c}) = 1] \end{array} \right|$$

is related to $\mathsf{Adv}_{\mathsf{1KLPN}[n,m,p]^m,\mathcal{A}}(\kappa)$ by the hybrid argument.

$\mathsf{Expt}^{\mathrm{real}}_{\mathsf{Gen_{td}},\mathcal{A}}(\kappa)$	$\mathsf{Expt}^{\mathrm{corr}}_{\mathsf{Gen_{td}},\mathcal{A}}(\kappa)$
$(t,\tau_0,\tau_1,\tau',\mathit{state}) \leftarrow \mathcal{A}(1^\kappa)$	$(t,\tau_0,\tau_1,\tau',\mathit{state}) \leftarrow \mathcal{A}(1^\kappa)$
	$\tau'_t := \tau_t; \tau'_{1-t} := \tau'$
$(\boldsymbol{T}_0,\boldsymbol{T}_1,(\boldsymbol{A},\boldsymbol{B}_0,\boldsymbol{B}_1)) \leftarrow \mathsf{Gen_{td}}(1^\kappa,\tau_0,\tau_1)$	$(\boldsymbol{T}_0,\boldsymbol{T}_1,(\boldsymbol{A},\boldsymbol{B}_0,\boldsymbol{B}_1)) \leftarrow \mathsf{Gen_{td}}(1^\kappa,\tau'_0,\tau'_1)$
$\boldsymbol{z} \leftarrow \mathsf{Ber}^m_p; \boldsymbol{T} \leftarrow \mathsf{Ber}^{m\times m}_p$	$\boldsymbol{z} \leftarrow \mathsf{Ber}^m_p; \boldsymbol{T} := \boldsymbol{T}_{1-t}$
$d \leftarrow \mathcal{A}(\boldsymbol{T}_t,(\boldsymbol{A},\boldsymbol{B}_0,\boldsymbol{B}_1),\boldsymbol{z},\boldsymbol{T}\boldsymbol{z},\mathit{state})$	$d \leftarrow \mathcal{A}(\boldsymbol{T}_t,(\boldsymbol{A},\boldsymbol{B}_0,\boldsymbol{B}_1),\boldsymbol{z},\boldsymbol{T}\boldsymbol{z},\mathit{state})$
return d	**return** d

Fig. 5. Games for Trapdoor Generation Algorithm

B The Kiltz-Masny-Pieprzak TBE

Before introduce the KMP TBE itself, we first review the trapdoor generation algorithm in [KMP14, Section 3]. We have field injective homomorphism from $\mathrm{GF}(2^n)$ into $\mathbb{F}_2^{n\times n}$. For finite field elements $\tau \in \mathrm{GF}(2^n)$, we use its companion matrix $\boldsymbol{H}_\tau \in \mathbb{F}_2^{n\times n}$. Let $\boldsymbol{G} \in \mathbb{F}_2^{m\times n}$ be a generator matrix for an efficiently decodable linear code. The trapdoor generation algorithm is defined as follows:

- $\mathsf{Gen_{td}}(1^\kappa,\tau_0,\tau_1) \to (\boldsymbol{T}_0,\boldsymbol{T}_1,(\boldsymbol{A},\boldsymbol{B}_0,\boldsymbol{B}_1))$: Sample $\boldsymbol{T}_0,\boldsymbol{T}_1 \leftarrow \mathsf{Ber}^{m\times m}_p$ and $\boldsymbol{A} \leftarrow \mathbb{F}_2^{m\times n}$. Compute $\boldsymbol{B}_0 := \boldsymbol{T}_0\boldsymbol{A} - \boldsymbol{G}\boldsymbol{H}_{\tau_0}$ and $\boldsymbol{B}_1 := \boldsymbol{T}_1\boldsymbol{A} - \boldsymbol{G}\boldsymbol{H}_{\tau_1}$. Output $(\boldsymbol{T}_0,\boldsymbol{T}_1,(\boldsymbol{A},\boldsymbol{B}_0,\boldsymbol{B}_1))$.

Kiltz et al. [KMP14] showed the following lemma. We will use this lemma in the security proof.

Lemma B.1 ([KMP14, Lemma 4]). *For every adversary $\mathcal{A}$, there exists another adversary $\mathcal{A}_{\mathsf{LPN}}$ such that*

$$\left|\Pr[\mathsf{Expt}^{\mathrm{real}}_{\mathsf{Gen_{td}},\mathcal{A}}(\kappa)=1] - \Pr[\mathsf{Expt}^{\mathrm{corr}}_{\mathsf{Gen_{td}},\mathcal{A}}(\kappa)=1]\right| \leq 3m\mathsf{Adv}_{\mathsf{LPN}[m-n,m,p],\mathcal{A}_{\mathsf{LPN}}}(\kappa),$$

where $\mathsf{Expt}^{\mathrm{real}}_{\mathsf{Gen_{td}},\mathcal{A}}(\kappa)$ and $\mathsf{Expt}^{\mathrm{corr}}_{\mathsf{Gen_{td}},\mathcal{A}}(\kappa)$ are defined in Figure 5.

B.1 The KMP TBE

Let us review the parameter setting:

- A dimension $n = \Theta(\kappa^2)$ and $m \geq 2n$.
- a constant $c \in (0, 1/4)$: We set $p = \sqrt{c/m}$ and $\beta = 2\sqrt{cm}$, and a binary linear error correcting code $\boldsymbol{G}\colon \mathbb{F}_2^n \to \mathbb{F}_2^m$, which corrects up to αm errors for some $\alpha \in (4c, 1)$.
- An efficient error correcting code with generator matrix $\boldsymbol{G}_2\colon \mathcal{M} \to \mathbb{F}_2^\ell$, where the parameter $\ell \geq m$ is adjusted as we can correct up to $2\ell\sqrt{c}/\sqrt{m} = 2\ell p$ errors.

We consider a tag space $\mathcal{T} = \mathrm{GF}(2^n) \setminus \{0\}$. Now, we review the KMP TBE scheme $(\mathsf{Gen_{KMP}},\mathsf{Enc_{KMP}},\mathsf{Dec_{KMP}})$:

- $\mathsf{Gen_{KMP}}(1^\kappa) \to (\mathit{ek},\mathit{dk})$: Generate $(\boldsymbol{T}_0,\boldsymbol{T}_1,(\boldsymbol{A},\boldsymbol{B}_0,\boldsymbol{B}_1)) \leftarrow \mathsf{Gen_{td}}(1^\kappa,0,0)$ and choose $\boldsymbol{C} \leftarrow \mathbb{F}_2^{\ell\times n}$. Output

$$\begin{aligned}\mathit{dk} &= (0,\boldsymbol{T}_0) \in \mathrm{GF}(2^n)\times\mathbb{F}_2^{m\times m},\\ \mathit{ek} &= (\boldsymbol{A},\boldsymbol{B}_0,\boldsymbol{B}_1,\boldsymbol{C}) \in (\mathbb{F}_2^{m\times n})^3\times\mathbb{F}_2^{\ell\times n}.\end{aligned}$$

- $\mathsf{Enc_{KMP}}(\mathit{ek},\tau,\mu) \to \mathit{ct} = (\boldsymbol{c},\boldsymbol{c}_0,\boldsymbol{c}_1,\boldsymbol{c}_2)$: Sample $\boldsymbol{e}_1 \leftarrow \mathsf{Ber}^m_p$, $\boldsymbol{e}_2 \leftarrow \mathsf{Ber}^\ell_p$, $\boldsymbol{T}'_0,\boldsymbol{T}'_1 \leftarrow \mathsf{Ber}^{m\times m}_p$, and $\boldsymbol{s} \leftarrow \mathbb{F}_2^n$. Compute

$$\begin{aligned}\boldsymbol{c} &:= \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}_1\\ \boldsymbol{c}_0 &:= (\boldsymbol{G}\boldsymbol{H}_\tau + \boldsymbol{B}_0)\boldsymbol{s} + \boldsymbol{T}'_0\boldsymbol{e}_1\\ \boldsymbol{c}_1 &:= (\boldsymbol{G}\boldsymbol{H}_\tau + \boldsymbol{B}_1)\boldsymbol{s} + \boldsymbol{T}'_1\boldsymbol{e}_1\\ \boldsymbol{c}_2 &:= \boldsymbol{C}\boldsymbol{s} + \boldsymbol{e}_2 + \boldsymbol{G}_2(\mu)\end{aligned}$$

  and output $\mathit{ct} = (\boldsymbol{c},\boldsymbol{c}_0,\boldsymbol{c}_1,\boldsymbol{c}_2) \in \mathbb{F}_2^m\times\mathbb{F}_2^m\times\mathbb{F}_2^m\times\mathbb{F}_2^\ell$.

Table 2. Summary of Games for the Proof of Theorem B.1:

Game	$\mathsf{Gen}_{\mathsf{td}}$	dk	$\boldsymbol{c}^*$	$\boldsymbol{c}_0^*$	$\boldsymbol{c}_1^*$	$\boldsymbol{c}_2^*$
Game_0	$(0,0)$	$(0,\boldsymbol{T}_0)$	$\boldsymbol{A}\boldsymbol{s}^*+\boldsymbol{e}^*$	$(\boldsymbol{G}\boldsymbol{H}_{\tau^*}+\boldsymbol{B}_0)\boldsymbol{s}^*+\boldsymbol{T}_0^*\boldsymbol{e}^*$	$(\boldsymbol{G}\boldsymbol{H}_{\tau^*}+\boldsymbol{B}_1)\boldsymbol{s}^*+\boldsymbol{T}_1^*\boldsymbol{e}^*$	$\boldsymbol{C}\boldsymbol{s}^*+\boldsymbol{e}_2^*\boldsymbol{G}_2(\mu)$
Game_1	$(0,\tau^*)$	$(0,\boldsymbol{T}_0)$	$\boldsymbol{A}\boldsymbol{s}^*+\boldsymbol{e}^*$	$(\boldsymbol{G}\boldsymbol{H}_{\tau^*}+\boldsymbol{B}_0)\boldsymbol{s}^*+\boldsymbol{T}_0^*\boldsymbol{e}^*$	$\boldsymbol{T}_1\boldsymbol{c}^*$	$\boldsymbol{C}\boldsymbol{s}^*+\boldsymbol{e}_2^*\boldsymbol{G}_2(\mu)$
Game_2	$(0,\tau^*)$	$(\tau^*,\boldsymbol{T}_1)$	$\boldsymbol{A}\boldsymbol{s}^*+\boldsymbol{e}^*$	$(\boldsymbol{G}\boldsymbol{H}_{\tau^*}+\boldsymbol{B}_0)\boldsymbol{s}^*+\boldsymbol{T}_0^*\boldsymbol{e}^*$	$\boldsymbol{T}_1\boldsymbol{c}^*$	$\boldsymbol{C}\boldsymbol{s}^*+\boldsymbol{e}_2^*\boldsymbol{G}_2(\mu)$
Game_3	(τ^*,τ^*)	$(\tau^*,\boldsymbol{T}_1)$	$\boldsymbol{A}\boldsymbol{s}^*+\boldsymbol{e}^*$	$\boldsymbol{T}_0\boldsymbol{c}^*$	$\boldsymbol{T}_1\boldsymbol{c}^*$	$\boldsymbol{C}\boldsymbol{s}^*+\boldsymbol{e}_2^*\boldsymbol{G}_2(\mu)$
Game_4	(τ^*,τ^*)	$(\tau^*,\boldsymbol{T}_1)$	$U(\mathbb{F}_2^m)$	$\boldsymbol{T}_0\boldsymbol{c}^*$	$\boldsymbol{T}_1\boldsymbol{c}^*$	$U(\mathbb{F}_2^\ell)$
Game_5	(τ^*,τ^*)	$(\tau^*,\boldsymbol{T}_1)$	$U(\mathbb{F}_2^m)$	$U(\mathbb{F}_2^m)$	$\boldsymbol{T}_1\boldsymbol{c}^*$	$U(\mathbb{F}_2^\ell)$
Game_6	$(0,\tau^*)$	$(\tau^*,\boldsymbol{T}_1)$	$U(\mathbb{F}_2^m)$	$U(\mathbb{F}_2^m)$	$\boldsymbol{T}_1\boldsymbol{c}^*$	$U(\mathbb{F}_2^\ell)$
Game_7	$(0,\tau^*)$	$(0,\boldsymbol{T}_0)$	$U(\mathbb{F}_2^m)$	$U(\mathbb{F}_2^m)$	$\boldsymbol{T}_1\boldsymbol{c}^*$	$U(\mathbb{F}_2^\ell)$
Game_8	$(0,\tau^*)$	$(0,\boldsymbol{T}_0)$	$U(\mathbb{F}_2^m)$	$U(\mathbb{F}_2^m)$	$U(\mathbb{F}_2^m)$	$U(\mathbb{F}_2^\ell)$
Game_9	$(0,0)$	$(0,\boldsymbol{T}_0)$	$U(\mathbb{F}_2^m)$	$U(\mathbb{F}_2^m)$	$U(\mathbb{F}_2^m)$	$U(\mathbb{F}_2^\ell)$

– $\mathsf{Dec}_{\mathsf{KMP}}(dk,\tau,ct)\to\mu/\bot$: Parse $dk=(\tau_b,\boldsymbol{T}_b)$ for $b=0$ or 1 and compute

$$\tilde{c}_b := (\boldsymbol{T}_b\ \boldsymbol{I})\cdot\begin{pmatrix}-\boldsymbol{c}\\ \boldsymbol{c}_b\end{pmatrix}\quad (=\boldsymbol{G}\boldsymbol{H}_{\tau-\tau_b}\boldsymbol{s}+(\boldsymbol{T}_b'-\boldsymbol{T}_b)\boldsymbol{e}_1).$$

Reconstruct $\boldsymbol{H}_{\tau-\tau_b}\boldsymbol{s}$ with error $(\boldsymbol{T}_b'-\boldsymbol{T}_b)\boldsymbol{e}_1$ by using the decoding algorithm of $\boldsymbol{G}$. Compute $\boldsymbol{s}=\boldsymbol{H}_{\tau-\tau_b}^{-1}\cdot\boldsymbol{H}_{\tau-\tau_b}\boldsymbol{s}$. If

$$\mathsf{HW}(\boldsymbol{c}-\boldsymbol{A}\boldsymbol{s})\le\beta\wedge\mathsf{HW}(\boldsymbol{c}_0-(\boldsymbol{G}\boldsymbol{H}_\tau+\boldsymbol{B}_0)\boldsymbol{s})\le\alpha m/2\wedge\mathsf{HW}(\boldsymbol{c}_1-(\boldsymbol{G}\boldsymbol{H}_\tau+\boldsymbol{B}_1)\boldsymbol{s})\le\alpha m/2$$

hold, then compute $\boldsymbol{c}_2-\boldsymbol{C}\boldsymbol{s}=\boldsymbol{G}_2(\mu)+\boldsymbol{e}_2$ and reconstruct μ by using the decoding algorithm of $\boldsymbol{G}_2$ and output it. Otherwise, output $\bot$.

This scheme is statistically correct. Kiltz et al. showed the next lemma, which states that we cannot distinguish the decryption oracles implemented with $\boldsymbol{T}_0$ or $\boldsymbol{T}_1$.

Lemma B.2 ([KMP14, Lemma 5]). *s. Let* $(\boldsymbol{T}_0,\boldsymbol{T}_1,(\boldsymbol{A},\boldsymbol{B}_0,\boldsymbol{B}_1))\leftarrow\mathsf{Gen}_{\mathsf{td}}(1^\kappa,\tau_0,\tau_1)$*,* $dk_0=(\tau_0,\boldsymbol{T}_0)$*,* $dk_1=(\tau_1,\boldsymbol{T}_1)$*, and* $ek:=(\boldsymbol{A},\boldsymbol{B}_0,\boldsymbol{B}_1,\boldsymbol{C})$ *with* $\boldsymbol{C}\leftarrow\mathbb{F}_2^{\ell\times n}$*. With overwhelming probability over the choice of the encryption and decryption keys,* $\mathsf{Dec}_{\mathsf{KMP}}$ *with* dk_0 *and* dk_1 *and* $\mathsf{Dec}_{\mathsf{KMP}1}$ *have the same output distribution; that is, we have*

$$\Pr_{ek,dk_0,dk_1}[\forall\tau_0,\tau_1,\tau\notin\{\tau_0,\tau_1\},ct,[\mathsf{Dec}_{\mathsf{KMP}}(dk_0,\tau,ct)=\mathsf{Dec}_{\mathsf{KMP}}(dk_1,\tau,ct)]]\ge 1-2^{-\Theta(m)}.$$

Kiltz et al. showed that their TBE is IND-ST-wCCA-secure assuming $\mathsf{LPN}[m-n,m,p]$ and $\mathsf{LPN}[n,m+\ell,p]$ is hard [KMP14, Theorem 2]. In the final game of their proof, the key is generated as $(\boldsymbol{T}_0^*,\boldsymbol{T}_1^*,(\boldsymbol{A},\boldsymbol{B}_0,\boldsymbol{B}_1))\leftarrow\mathsf{Gen}(1^n,\tau^*,\tau^*)$, the decryption key is $(\tau^*,\boldsymbol{T}_1^*)$, and the challenge ciphertext is generated as $\boldsymbol{c}^*\leftarrow\mathbb{F}_2^m$, $\boldsymbol{c}_0^*\leftarrow\boldsymbol{T}_0^*\boldsymbol{c}^*$, $\boldsymbol{c}_1^*\leftarrow\boldsymbol{T}_1^*\boldsymbol{c}^*$ and $\boldsymbol{c}_2^*\leftarrow\mathbb{F}_2^\ell$. We notice that $\boldsymbol{c}_0^*$ and $\boldsymbol{c}_1^*$ are still correlated to $\boldsymbol{B}_0=\boldsymbol{T}_0^*\boldsymbol{A}$ and $\boldsymbol{B}_1=\boldsymbol{T}_1^*\boldsymbol{A}$. Thus, we should continue to modify the security game in order to cut off the correlation between keys and ciphertexts. In order to do so, we have introduced 1KLPN assumption, which hold if KLPN holds.

Theorem B.1. $\mathsf{TBE}_{\mathsf{KMP}}$ *is* OS-ST-wCCA-*secure if the LPN/KLPN/EKLPN/1KLPN assumptions hold.*

We mainly follow the definitions of games in the original paper. We summarize games in Table 2.

Game_0*:* This is the original game with $b=0$ expanded as follows:

1. The challenger runs the adversary on input 1^κ.
2. The adversary outputs τ^*. The challenger generates keys by $(\boldsymbol{T}_0,\boldsymbol{T}_1,(\boldsymbol{A},\boldsymbol{B}_0,\boldsymbol{B}_1))\leftarrow\mathsf{Gen}_{\mathsf{td}}(1^n,0,0)$ and $\boldsymbol{C}\leftarrow\mathbb{F}_2^{\ell\times n}$. It sets $ek=(\boldsymbol{A},\boldsymbol{B}_0,\boldsymbol{B}_1,\boldsymbol{C})$ and $dk=(0,\boldsymbol{T}_0,ek)$. It runs the adversary on input ek and simulates the decryption oracle by using dk.
3. The adversary outputs μ. The challenger generates the challenge ciphertext as follows: It computes
   - $\boldsymbol{s}^*\leftarrow\mathbb{F}_2^n$, $\boldsymbol{e}^*\leftarrow\mathsf{Ber}_p^m$, $\boldsymbol{e}_2^*\leftarrow\mathsf{Ber}_p^\ell$, $\boldsymbol{T}_0^*\leftarrow\mathsf{Ber}_p^{m\times m}$, $\boldsymbol{T}_1^*\leftarrow\mathsf{Ber}_p^{m\times m}$
   - $\boldsymbol{c}^*:=\boldsymbol{A}\boldsymbol{s}^*+\boldsymbol{e}^*$
   - $\boldsymbol{c}_0^*:=(\boldsymbol{G}\boldsymbol{H}_{\tau^*}+\boldsymbol{B}_0)\boldsymbol{s}^*+\boldsymbol{T}_0^*\boldsymbol{e}^*$
   - $\boldsymbol{c}_1^*:=(\boldsymbol{G}\boldsymbol{H}_{\tau^*}+\boldsymbol{B}_1)\boldsymbol{s}^*+\boldsymbol{T}_1^*\boldsymbol{e}^*$

– $\boldsymbol{c}_2^* := \boldsymbol{C}\boldsymbol{s}^* + \boldsymbol{e}_2^* + \boldsymbol{G}_2(\mu)$

and returns $ct^* = (\boldsymbol{c}^*, \boldsymbol{c}_0^*, \boldsymbol{c}_1^*, \boldsymbol{c}_2^*)$. It runs the adversary on input ct^* and simulates the decryption oracle by using dk.

4. Finally, the adversary outputs its guess b' and the challenger outputs b'.

Apparently, we have

$$\Pr[S_0] = \Pr[\mathsf{Expt}^{\text{pr-st-wcca},0}_{\mathsf{TBE}_{\mathsf{KMP}},\mathcal{A}}(\kappa) = 1].$$

Game_1: We next change how to generate $\boldsymbol{T}_1^*$ and $\boldsymbol{c}_1^*$:

1. The challenger runs the adversary on input 1^κ.
2. The adversary outputs τ^*. The challenger generates keys by $(\boldsymbol{T}_0, \boldsymbol{T}_1, (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1)) \leftarrow \underline{\mathsf{Gen}_{\mathsf{td}}(1^n, 0, \tau^*)}$ and $\boldsymbol{C} \leftarrow \mathbb{F}_2^{\ell\times n}$. It sets $ek = (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{C})$ and $dk = (0, \boldsymbol{T}_0, ek)$. It runs the adversary on input ek and simulates the decryption oracle by using dk.
3. The adversary outputs μ. The challenger generates the challenge ciphertext as follows: It computes
   - $\boldsymbol{s}^* \leftarrow \mathbb{F}_2^n$, $\boldsymbol{e}^* \leftarrow \mathsf{Ber}_p^m$, $\boldsymbol{e}_2^* \leftarrow \mathsf{Ber}_p^\ell$, $\boldsymbol{T}_0^* \leftarrow \mathsf{Ber}_p^{m\times m}$
   - $\boldsymbol{c}^* := \boldsymbol{A}\boldsymbol{s}^* + \boldsymbol{e}^*$
   - $\boldsymbol{c}_0^* := (\boldsymbol{G}\boldsymbol{H}_{\tau^*} + \boldsymbol{B}_0)\boldsymbol{s}^* + \boldsymbol{T}_0^*\boldsymbol{e}^*$
   - $\underline{\boldsymbol{c}_1^* := \boldsymbol{T}_1\boldsymbol{c}^*}$ $(= (\boldsymbol{G}\boldsymbol{H}_{\tau^*} + \boldsymbol{B}_1)\boldsymbol{s}^* + \boldsymbol{T}_1\boldsymbol{e}^*)$
   - $\boldsymbol{c}_2^* := \boldsymbol{C}\boldsymbol{s}^* + \boldsymbol{e}_2^* + \boldsymbol{G}_2(\mu)$

   and returns $ct^* = (\boldsymbol{c}^*, \boldsymbol{c}_0^*, \boldsymbol{c}_1^*, \boldsymbol{c}_2^*)$. It runs the adversary on input ct^* and simulates the decryption oracle by using dk.
4. Finally, the adversary outputs its guess b' and the challenger outputs b'.

Lemma B.3 ([KMP14, Lemma 6]). *There exists an adversary $\mathcal{A}_{01}$ satisfying*

$$|\Pr[S_0] - \Pr[S_1]| \leq \left|\Pr[\mathsf{Expt}^{\text{real}}_{\mathsf{Gen}_{\mathsf{td}},\mathcal{A}_{01}}(\kappa) = 1] - \Pr[\mathsf{Expt}^{\text{corr}}_{\mathsf{Gen}_{\mathsf{td}},\mathcal{A}_{01}}(\kappa) = 1]\right|.$$

The proof of lemma invokes Lemma B.1.

Game_2: We change how to generate $\boldsymbol{T}_1^*$ and $\boldsymbol{c}_1^*$:

1. The challenger runs the adversary on input 1^κ.
2. The adversary outputs τ^*. The challenger generates keys by $(\boldsymbol{T}_0, \boldsymbol{T}_1, (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1)) \leftarrow \mathsf{Gen}_{\mathsf{td}}(1^n, 0, \tau^*)$ and $\boldsymbol{C} \leftarrow \mathbb{F}_2^{\ell\times n}$. It sets $ek = (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{C})$ and $\underline{dk = (\tau^*, \boldsymbol{T}_1, ek)}$. It runs the adversary on input ek and simulates the decryption oracle by using dk.
3. The adversary outputs μ. The challenger generates the challenge ciphertext as follows: It computes
   - $\boldsymbol{s}^* \leftarrow \mathbb{F}_2^n$, $\boldsymbol{e}^* \leftarrow \mathsf{Ber}_p^m$, $\boldsymbol{e}_2^* \leftarrow \mathsf{Ber}_p^\ell$, $\boldsymbol{T}_0^* \leftarrow \mathsf{Ber}_p^{m\times m}$
   - $\boldsymbol{c}^* := \boldsymbol{A}\boldsymbol{s}^* + \boldsymbol{e}^*$
   - $\boldsymbol{c}_0^* := (\boldsymbol{G}\boldsymbol{H}_{\tau^*} + \boldsymbol{B}_0)\boldsymbol{s}^* + \boldsymbol{T}_0^*\boldsymbol{e}^*$
   - $\boldsymbol{c}_1^* := \boldsymbol{T}_1\boldsymbol{c}^*$
   - $\boldsymbol{c}_2^* := \boldsymbol{C}\boldsymbol{s}^* + \boldsymbol{e}_2^* + \boldsymbol{G}_2(\mu)$

   and returns $ct^* = (\boldsymbol{c}^*, \boldsymbol{c}_0^*, \boldsymbol{c}_1^*, \boldsymbol{c}_2^*)$. It runs the adversary on input ct^* and simulates the decryption oracle by using dk.
4. Finally, the adversary outputs its guess b' and the challenger outputs b'.

Lemma B.4 ([KMP14, Lemma 7]).

$$|\Pr[S_1] - \Pr[S_2]| \leq \mathsf{negl}(\kappa).$$

This lemma follows from Lemma B.2.

Game_3: We change how to generate $\boldsymbol{T}_0^*$ and $\boldsymbol{c}_0^*$:

1. The challenger runs the adversary on input 1^κ.
2. The adversary outputs τ^*. The challenger generates keys by $(\boldsymbol{T}_0, \boldsymbol{T}_1, (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1)) \leftarrow \underline{\mathsf{Gen}_{\mathsf{td}}(1^n, \tau^*, \tau^*)}$ and $\boldsymbol{C} \leftarrow \mathbb{F}_2^{\ell\times n}$. It sets $ek = (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{C})$ and $dk = (\tau^*, \boldsymbol{T}_1, ek)$. It runs the adversary on input ek and simulates the decryption oracle by using dk.
3. The adversary outputs μ. The challenger generates the challenge ciphertext as follows: It computes
   - $\boldsymbol{s}^* \leftarrow \mathbb{F}_2^n$, $\boldsymbol{e}^* \leftarrow \mathsf{Ber}_p^m$, $\boldsymbol{e}_2^* \leftarrow \mathsf{Ber}_p^\ell$
   - $\boldsymbol{c}^* := \boldsymbol{A}\boldsymbol{s}^* + \boldsymbol{e}^*$

– $\underline{\boldsymbol{c}_0^* := \boldsymbol{T}_0\boldsymbol{c}^*}$ $(= (\boldsymbol{G}\boldsymbol{H}_{\tau^*} + \boldsymbol{B}_0)\boldsymbol{s}^* + \boldsymbol{T}_0\boldsymbol{e}^*)$
– $\underline{\boldsymbol{c}_1^* := \boldsymbol{T}_1\boldsymbol{c}^*}$
– $\boldsymbol{c}_2^* := \boldsymbol{C}\boldsymbol{s}^* + \boldsymbol{e}_2^* + \boldsymbol{G}_2(\mu)$

and returns $ct^* = (\boldsymbol{c}^*, \boldsymbol{c}_0^*, \boldsymbol{c}_1^*, \boldsymbol{c}_2^*)$. It runs the adversary on input ct^* and simulates the decryption oracle by using dk.

4. Finally, the adversary outputs its guess b' and the challenger outputs b'.

Lemma B.5 ([KMP14, Lemma 8]). *There exists an adversary $\mathcal{A}_{23}$ satisfying*

$$|\Pr[S_2] - \Pr[S_3]| \leq \left|\Pr[\mathsf{Expt}^{\mathrm{real}}_{\mathsf{Gen}_{\mathsf{td}},\mathcal{A}_{23}}(\kappa) = 1] - \Pr[\mathsf{Expt}^{\mathrm{corr}}_{\mathsf{Gen}_{\mathsf{td}},\mathcal{A}_{23}}(\kappa) = 1]\right|.$$

This lemma invokes Lemma B.1.

Game_4: We change how to generate $\boldsymbol{c}^*$ and $\boldsymbol{c}_2^*$:

1. The challenger runs the adversary on input 1^κ.
2. The adversary outputs τ^*. The challenger generates keys by $(\boldsymbol{T}_0, \boldsymbol{T}_1, (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1)) \leftarrow \mathsf{Gen}_{\mathsf{td}}(1^n, \tau^*, \tau^*)$ and $\boldsymbol{C} \leftarrow \mathbb{F}_2^{\ell\times n}$. It sets $ek = (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{C})$ and $dk = (\tau^*, \boldsymbol{T}_1, ek)$. It runs the adversary on input ek and simulates the decryption oracle by using dk.
3. The adversary outputs μ. The challenger generates the challenge ciphertext as follows: It computes
   – $\underline{\boldsymbol{c}^* \leftarrow \mathbb{F}_2^m}$
   – $\boldsymbol{c}_0^* := \boldsymbol{T}_0\boldsymbol{c}^*$
   – $\boldsymbol{c}_1^* := \boldsymbol{T}_1\boldsymbol{c}^*$
   – $\underline{\boldsymbol{c}_2^* \leftarrow \mathbb{F}_2^\ell}$

   and returns $ct^* = (\boldsymbol{c}^*, \boldsymbol{c}_0^*, \boldsymbol{c}_1^*, \boldsymbol{c}_2^*)$. It runs the adversary on input ct^* and simulates the decryption oracle by using dk.
4. Finally, the adversary outputs its guess b' and the challenger outputs b'.

Lemma B.6 ([KMP14, Lemma 9]). *We have an adversary $\mathcal{A}_{34}$ satisfying*

$$|\Pr[S_3] - \Pr[S_4]| \leq \mathsf{Adv}_{\mathsf{LPN}[n,m+\ell,p],\mathcal{A}_{34}}(\kappa).$$

In the original IND-security proof, this is the final game. We continue the modification of games, since we want to modify $\boldsymbol{c}_0^*$ and $\boldsymbol{c}_1^*$ further.

Game_5: We modify the game to make $\boldsymbol{c}_0^*$ random.

1. The challenger runs the adversary on input 1^κ.
2. The adversary outputs τ^*. The challenger generates keys by $(\boldsymbol{T}_0, \boldsymbol{T}_1, (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1)) \leftarrow \mathsf{Gen}_{\mathsf{td}}(1^n, \tau^*, \tau^*)$ and $\boldsymbol{C} \leftarrow \mathbb{F}_2^{\ell\times n}$. It sets $ek = (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{C})$ and $dk = (\tau^*, \boldsymbol{T}_1, ek)$. It runs the adversary on input ek and simulates the decryption oracle by using dk.
3. The adversary outputs μ. The challenger generates the challenge ciphertext as follows: It computes
   – $\boldsymbol{c}^* \leftarrow \mathbb{F}_2^m$
   – $\underline{\boldsymbol{c}_0^* \leftarrow \mathbb{F}_2^m}$
   – $\boldsymbol{c}_1^* := \boldsymbol{T}_1\boldsymbol{c}^*$
   – $\boldsymbol{c}_2^* \leftarrow \mathbb{F}_2^\ell$

   and returns $ct^* = (\boldsymbol{c}^*, \boldsymbol{c}_0^*, \boldsymbol{c}_1^*, \boldsymbol{c}_2^*)$. It runs the adversary on input ct^* and simulates the decryption oracle by using dk.
4. Finally, the adversary outputs its guess b' and the challenger outputs b'.

In this game, the adversary is given

$$\boldsymbol{A}, \boldsymbol{B}_0 = \boldsymbol{T}_0\boldsymbol{A} - \boldsymbol{G}\boldsymbol{H}_{\tau^*}, \boldsymbol{c}^*, \text{ and } \boldsymbol{c}_0^* = \boldsymbol{T}_0\boldsymbol{c}^* \text{ or random.}$$

We use the 1KLPN assumption here.

Lemma B.7. *There exists a PPT adversary $\mathcal{A}_{45}$ satisfying*

$$|\Pr[S_4] - \Pr[S_5]| \leq \mathsf{Adv}_{\mathsf{1KLPN}[n-1,m,p]^m,\mathcal{A}_{45}}(\kappa).$$

Proof. We construct $\mathcal{A}_{45}$ as follows:

1. $\mathcal{A}_{45}$ is given $(\boldsymbol{A}, \boldsymbol{c}^*, \boldsymbol{T}_0\boldsymbol{A}, \boldsymbol{x})$, where $\boldsymbol{x}$ is $\boldsymbol{T}_0\boldsymbol{c}^*$ or random $\boldsymbol{u}$.
2. $\mathcal{A}_{45}$ runs $\mathcal{A}$ on input 1^κ and receives τ^*.
3. $\mathcal{A}_{45}$ generates keys as follows: $\boldsymbol{T}_1 \leftarrow \mathsf{Ber}_p^{m\times m}$, $\boldsymbol{B}_0 := \boldsymbol{T}_0\boldsymbol{A} - \boldsymbol{G}\boldsymbol{H}_{\tau^*}$, $\boldsymbol{B}_1 := \boldsymbol{T}_1\boldsymbol{A} - \boldsymbol{G}\boldsymbol{H}_{\tau^*}$, and $\boldsymbol{C} \leftarrow \mathbb{F}_2^{\ell\times n}$. It sets $dk = (\tau^*, \boldsymbol{T}_1)$ and $ek = (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{C})$. It runs $\mathcal{A}$ on input ek.
4. $\mathcal{A}_{45}$ simulates the decryption oracle using dk.
5. $\mathcal{A}_{45}$ generates the challenge on a query μ from $\mathcal{A}$ as follows: It generates $\boldsymbol{c}_0^* := \boldsymbol{x}$ and $\boldsymbol{c}_1^* := \boldsymbol{T}_1\boldsymbol{c}^*$. It chooses $\boldsymbol{c}_2^* \leftarrow \mathbb{F}_2^\ell$ and returns $ct^* := (\boldsymbol{c}^*, \boldsymbol{c}_0^*, \boldsymbol{c}_1^*, \boldsymbol{c}_2^*)$.
6. $\mathcal{A}_{45}$ outputs b' if $\mathcal{A}$ finally outputs b'.

If $\boldsymbol{x} = \boldsymbol{T}_0\boldsymbol{c}^*$, then $\mathcal{A}_{45}$ perfectly simulates Game_4. On the other hand, if $\boldsymbol{x}$ is uniformly at random, then $\mathcal{A}_{45}$ perfectly simulates Game_5. Thus, the lemma holds. □

Game_6*:* We change how to generate keys:
1. The challenger runs the adversary on input 1^κ.
2. The adversary outputs τ^*. The challenger generates keys by $(\boldsymbol{T}_0, \boldsymbol{T}_1, (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1)) \leftarrow \underline{\mathsf{Gen}_{\mathsf{td}}(1^n, 0, \tau^*)}$ and $\boldsymbol{C} \leftarrow \mathbb{F}_2^{\ell\times n}$. It sets $ek = (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{C})$ and $dk = (\tau^*, \boldsymbol{T}_1, ek)$. It runs the adversary on input ek and simulates the decryption oracle by using dk.
3. The adversary outputs μ. The challenger generates the challenge ciphertext as follows: It computes
   - $\boldsymbol{c}^* \leftarrow \mathbb{F}_2^m$
   - $\boldsymbol{c}_0^* \leftarrow \mathbb{F}_2^m$
   - $\boldsymbol{c}_1^* := \boldsymbol{T}_1\boldsymbol{c}^*$
   - $\boldsymbol{c}_2^* \leftarrow \mathbb{F}_2^\ell$

   and returns $ct^* = (\boldsymbol{c}^*, \boldsymbol{c}_0^*, \boldsymbol{c}_1^*, \boldsymbol{c}_2^*)$. It runs the adversary on input ct^* and simulates the decryption oracle by using dk.
4. Finally, the adversary outputs its guess b' and the challenger outputs b'.

Lemma B.8. *There exists an adversary $\mathcal{A}_{56}$ satisfying*

$$|\Pr[S_5] - \Pr[S_6]| \leq \left|\Pr[\mathsf{Expt}^{\mathsf{real}}_{\mathsf{Gen}_{\mathsf{td}},\mathcal{A}_{56}}(\kappa) = 1] - \Pr[\mathsf{Expt}^{\mathsf{corr}}_{\mathsf{Gen}_{\mathsf{td}},\mathcal{A}_{56}}(\kappa) = 1]\right|.$$

Proof. We construct $\mathcal{A}_{56}$ that distinguishes real and corr games as follows:
1. Given 1^κ, $\mathcal{A}_{56}$ runs $\mathcal{A}$ on input 1^κ and receives τ^*.
2. It sends $(1, \tau^*, \tau^*, 0)$ to its challenger and receives $(\boldsymbol{T}_1, \boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1, z, \boldsymbol{T}z)$. It chooses $\boldsymbol{C} \leftarrow \mathbb{F}_2^{\ell\times n}$. It sets $ek = (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{C})$ and $dk = (\tau^*, \boldsymbol{T}_1, ek)$. It runs $\mathcal{A}$ on input ek.
3. $\mathcal{A}_{56}$ simulates the decryption oracle using dk.
4. $\mathcal{A}_{56}$ generates the challenge on a query μ from $\mathcal{A}$ as follows: It generates $\boldsymbol{c}^* \leftarrow \mathbb{F}_2^m$, $\boldsymbol{c}_0^* \leftarrow \mathbb{F}_2^m$, and $\boldsymbol{c}_1^* := \boldsymbol{T}_1\boldsymbol{c}^*$. It also chooses $\boldsymbol{c}_2^* \leftarrow \mathbb{F}_2^\ell$ and returns $ct^* := (\boldsymbol{c}^*, \boldsymbol{c}_0^*, \boldsymbol{c}_1^*, \boldsymbol{c}_2^*)$.
5. $\mathcal{A}_{56}$ outputs b' if $\mathcal{A}$ finally outputs b'.

Notice that if the game is real and corr, then the keys are generated by $\mathsf{Gen}_{\mathsf{td}}(1^\kappa, \tau^*, \tau^*)$ and $\mathsf{Gen}_{\mathsf{td}}(1^\kappa, 0, \tau^*)$, respectively. Thus, if the game is real and the keys are generated by $\mathsf{Gen}_{\mathsf{td}}(1^\kappa, \tau^*, \tau^*)$, then $\mathcal{A}_{56}$ perfectly simulates Game_5. If the game is corr and keys are generated by $\mathsf{Gen}_{\mathsf{td}}(1^\kappa, 0, \tau^*)$, then $\mathcal{A}_{56}$ perfectly simulates Game_6. This completes the proof. □

Game_7*:* We change the decryption key.
1. The challenger runs the adversary on input 1^κ.
2. The adversary outputs τ^*. The challenger generates keys by $(\boldsymbol{T}_0, \boldsymbol{T}_1, (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1)) \leftarrow \mathsf{Gen}_{\mathsf{td}}(1^n, 0, \tau^*)$ and $\boldsymbol{C} \leftarrow \mathbb{F}_2^{\ell\times n}$. It sets $ek = (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{C})$ and $\underline{dk = (0, \boldsymbol{T}_0, dk)}$. It runs the adversary on input ek and simulates the decryption oracle by using dk.
3. The adversary outputs μ. The challenger generates the challenge ciphertext as follows: It computes
   - $\boldsymbol{c}^* \leftarrow \mathbb{F}_2^m$
   - $\boldsymbol{c}_0^* \leftarrow \mathbb{F}_2^m$
   - $\boldsymbol{c}_1^* := \boldsymbol{T}_1\boldsymbol{c}^*$
   - $\boldsymbol{c}_2^* \leftarrow \mathbb{F}_2^\ell$

   and returns $ct^* = (\boldsymbol{c}^*, \boldsymbol{c}_0^*, \boldsymbol{c}_1^*, \boldsymbol{c}_2^*)$. It runs the adversary on input ct^* and simulates the decryption oracle by using dk.
4. Finally, the adversary outputs its guess b' and the challenger outputs b'.

Following Lemma B.4, we can switch decryption key:

Lemma B.9. *We have*

$$|\Pr[S_6] - \Pr[S_7]| \leq \mathsf{negl}(\kappa).$$

Game_8*:* We change how to generate $\boldsymbol{c}_2^*$:

1. The challenger runs the adversary on input 1^κ.
2. The adversary outputs τ^*. The challenger generates keys by $(\boldsymbol{T}_0, \boldsymbol{T}_1, (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1)) \leftarrow \mathsf{Gen}_{\mathsf{td}}(1^n, 0, \tau^*)$ and $\boldsymbol{C} \leftarrow \mathbb{F}_2^{\ell \times n}$. It sets $ek = (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{C})$ and $dk = (0, \boldsymbol{T}_0, ek)$. It runs the adversary on input ek and simulates the decryption oracle by using dk.
3. The adversary outputs μ. The challenger generates the challenge ciphertext as follows: It computes
   - $\boldsymbol{c}^* \leftarrow \mathbb{F}_2^m$
   - $\boldsymbol{c}_0^* \leftarrow \mathbb{F}_2^m$
   - $\boldsymbol{c}_1^* \leftarrow \mathbb{F}_2^m$
   - $\underline{\boldsymbol{c}_2^* \leftarrow \mathbb{F}_2^\ell}$

   and returns $ct^* = (\boldsymbol{c}^*, \boldsymbol{c}_0^*, \boldsymbol{c}_1^*, \boldsymbol{c}_2^*)$. It runs the adversary on input ct^* and simulates the decryption oracle by using dk.
4. Finally, the adversary outputs its guess b' and the challenger outputs b'.

Lemma B.10. *There exists a PPT adversary $\mathcal{A}_{78}$ satisfying*

$$|\Pr[S_7] - \Pr[S_8]| \leq \mathsf{Adv}_{\mathsf{1KLPN}[n-1,m,p]^m, \mathcal{A}_{78}}(\kappa).$$

Proof. We construct $\mathcal{A}_{78}$ as follows:

1. $\mathcal{A}_{78}$ is given $(\boldsymbol{A}, \boldsymbol{c}^*, \boldsymbol{T}_1\boldsymbol{A}, \boldsymbol{x})$, where $\boldsymbol{x}$ is $\boldsymbol{T}_1\boldsymbol{c}^*$ or random $\boldsymbol{u}$.
2. $\mathcal{A}_{78}$ runs $\mathcal{A}$ on input 1^κ and receives τ^*.
3. $\mathcal{A}_{78}$ generates keys as follows: $\boldsymbol{T}_0 \leftarrow \mathsf{Ber}_p^{m \times m}$, $\boldsymbol{B}_0 := \boldsymbol{T}_0\boldsymbol{A} - \boldsymbol{G}\boldsymbol{H}_{\tau^*}$, $\boldsymbol{B}_1 := \boldsymbol{T}_1\boldsymbol{A} - \boldsymbol{G}\boldsymbol{H}_{\tau^*}$, and $\boldsymbol{C} \leftarrow \mathbb{F}_2^{\ell \times n}$. It sets $ek = (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{C})$ and $dk = (0, \boldsymbol{T}_0, ek)$. It runs $\mathcal{A}$ on input ek.
4. $\mathcal{A}_{78}$ simulates the decryption oracle using dk.
5. $\mathcal{A}_{78}$ generates the challenge on a query μ from $\mathcal{A}$ as follows: It generates $\boldsymbol{c}_0^* \leftarrow \mathbb{F}_2^m$ and $\boldsymbol{c}_1^* := \boldsymbol{x}$. It also chooses $\boldsymbol{c}_2^* \leftarrow \mathbb{F}_2^\ell$ and returns $ct^* := (\boldsymbol{c}^*, \boldsymbol{c}_0^*, \boldsymbol{c}_1^*, \boldsymbol{c}_2^*)$.
6. $\mathcal{A}_{78}$ outputs b' if $\mathcal{A}$ finally outputs b'.

If $\boldsymbol{x} = \boldsymbol{T}_1\boldsymbol{c}^*$, then $\mathcal{A}_{78}$ perfectly simulates Game_7. On the other hand, if $\boldsymbol{x}$ is uniformly at random, then $\mathcal{A}_{78}$ perfectly simulates Game_8. Thus, the lemma holds. ◻

Game_9*:* We modify how to generate keys. This is the original game with $b = 1$:

1. The challenger runs the adversary on input 1^κ.
2. The adversary outputs τ^*. The challenger generates keys by $(\boldsymbol{T}_0, \boldsymbol{T}_1, (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1)) \leftarrow \underline{\mathsf{Gen}_{\mathsf{td}}(1^n, 0, 0)}$ and $\boldsymbol{C} \leftarrow \mathbb{F}_2^{\ell \times n}$. It sets $ek = (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{C})$ and $dk = (0, \boldsymbol{T}_0, ek)$. It runs the adversary on input ek and simulates the decryption oracle by using dk.
3. The adversary outputs μ. The challenger generates the challenge ciphertext as follows: It computes
   - $\boldsymbol{c}^* \leftarrow \mathbb{F}_2^m$
   - $\boldsymbol{c}_0^* \leftarrow \mathbb{F}_2^m$
   - $\boldsymbol{c}_1^* \leftarrow \mathbb{F}_2^m$
   - $\boldsymbol{c}_2^* \leftarrow \mathbb{F}_2^\ell$

   and returns $ct^* = (\boldsymbol{c}^*, \boldsymbol{c}_0^*, \boldsymbol{c}_1^*, \boldsymbol{c}_2^*)$. It runs the adversary on input ct^* and simulates the decryption oracle by using dk.
4. Finally, the adversary outputs its guess b' and the challenger outputs b'.

Lemma B.11. *There exists an adversary $\mathcal{A}_{89}$ satisfying*

$$|\Pr[S_8] - \Pr[S_9]| \leq \left| \Pr[\mathsf{Expt}^{\mathrm{real}}_{\mathsf{Gen}_{\mathsf{td}}, \mathcal{A}_{89}}(\kappa) = 1] - \Pr[\mathsf{Expt}^{\mathrm{corr}}_{\mathsf{Gen}_{\mathsf{td}}, \mathcal{A}_{89}}(\kappa) = 1] \right|.$$

Proof. We construct $\mathcal{A}_{89}$ that distinguishes real and corr games as follows:

1. Given 1^κ, $\mathcal{A}_{89}$ runs $\mathcal{A}$ on input 1^κ and receives τ^*.
2. It sends $(0, 0, \tau^*, 0)$ to its challenger and receives $(\boldsymbol{T}_0, \boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1, z, \boldsymbol{T}z)$. It chooses $\boldsymbol{C} \leftarrow \mathbb{F}_2^{\ell \times n}$. It sets $dk = (0, \boldsymbol{T}_0)$ and $ek = (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{C})$. It runs $\mathcal{A}$ on input ek.
3. $\mathcal{A}_{89}$ simulates the decryption oracle using dk.
4. $\mathcal{A}_{89}$ generates the challenge on a query μ from $\mathcal{A}$ as follows: It generates $\boldsymbol{c}^* \leftarrow \mathbb{F}_2^m$, $\boldsymbol{c}_0^* \leftarrow \mathbb{F}_2^m$, $\boldsymbol{c}_1^* \leftarrow \mathbb{F}_2^m$. It chooses $\boldsymbol{c}_2^* \leftarrow \mathbb{F}_2^\ell$ and returns $ct^* := (\boldsymbol{c}^*, \boldsymbol{c}_0^*, \boldsymbol{c}_1^*, \boldsymbol{c}_2^*)$.
5. $\mathcal{A}_{89}$ outputs b' if $\mathcal{A}$ finally outputs b'.

Notice that if the game is real and corr, then the keys are generated by $\mathsf{Gen}_{\mathsf{td}}(1^{\kappa}, 0, \tau^*)$ and $\mathsf{Gen}_{\mathsf{td}}(1^{\kappa}, 0, 0)$, respectively. Thus, if the game is real and the keys are generated by $\mathsf{Gen}_{\mathsf{td}}(1^{\kappa}, 0, \tau^*)$, then $\mathcal{A}_{89}$ perfectly simulates Game_8. If the game is corr and keys are generated by $\mathsf{Gen}_{\mathsf{td}}(1^{\kappa}, 0, 0)$, then $\mathcal{A}_{89}$ perfectly simulates Game_9. This completes the proof. □

Apparently, we have

$$\Pr[S_9] = \Pr[\mathsf{Expt}^{\text{pr-st-wcca},1}_{\mathsf{TBE}_{\mathsf{KMP}},\mathcal{A}}(\kappa) = 1].$$

This completes the proof. □

C The Yu-Zhang TBE

Yu and Zhang [YZ16] also proposed tag-based encryption whose IND-ST-wCCA security is based on the sub-exponential hardness of constant-rate LPN. We here show its PR-ST-wCCA security without changing the assumptions.

Preliminaries: $\mathcal{D}_{\lambda}^{n_1 \times n}$ denotes a matrix distribution induced by multiplying two random matrices chosen from $U(\mathbb{F}_2^{n_1 \times \lambda})$ and $U(\mathbb{F}_2^{\lambda \times n})$. $\widetilde{\mathsf{Ber}}_{\mu_1}^n$ is a distribution $\mathsf{Ber}_{\mu_1}^n$ conditioned on $(1 - \sqrt{6}/3)\mu_1 n \leq \mathsf{HW}(\mathsf{Ber}_{\mu_1}^n) \leq 2\mu_1 n$. This is efficiently samplable, because $\Pr[(1 - \sqrt{6}/3)\mu_1 n \leq \mathsf{HW}(e) \leq 2\mu_1 n \mid e \leftarrow \widetilde{\mathsf{Ber}}_{\mu_1}^n]$ is noticeable. Yu and Zhang showed that for $\mu_1 = \Omega(\lg(n)/n)$, $\widetilde{\mathsf{Ber}}_{\mu_1}^n$ has the min-entropy $\Omega(\lg^2(n))$. $\widetilde{\mathsf{Ber}}_{\mu_1}^{q \times n}$ denotes a matrix distribution whose each row is chosen from $\widetilde{\mathsf{Ber}}_{\mu_1}^n$.
Yu and Zhang showed the following lemma, which states that if constant-rate LPN is sub-exponentially hard, then 'leaky' LPN is computationally hard.

Lemma C.1 ([YZ16, Corollary .5.1]). *Let n be a security parameter and let $\mu \in (0, 1/2)$ be any constant. Suppose that $\mathsf{LPN}_{\mu,n}$ problem is $2^{\omega(n^{1/2})}$-hard (for any super-constant hidden by $\omega(\cdot)$). Then, for every $\mu_1 = \Omega(\lg n/n)$ and $\lambda = \Theta(\lg^2 n)$ such that $2\lambda \leq H_\infty(\widetilde{\mathsf{Ber}}_{\mu_1}^n)$, and every $q = \mathsf{poly}(n)$, we have*

$$((\boldsymbol{S}_0\boldsymbol{e}, \boldsymbol{E}_0\boldsymbol{s}), \boldsymbol{e}, \boldsymbol{s}, \boldsymbol{A}, \boldsymbol{S}_0\boldsymbol{A} + \boldsymbol{E}_0) \approx_c ((\boldsymbol{S}_0\boldsymbol{e}, \boldsymbol{E}_0\boldsymbol{s}), \boldsymbol{e}, \boldsymbol{s}, \boldsymbol{A}, \boldsymbol{B}),$$

where the probability is take over $\boldsymbol{S}_0 \leftarrow \widetilde{\mathsf{Ber}}_{\mu_1}^{q\times n}$, $\boldsymbol{E}_0 \leftarrow \mathsf{Ber}_{\mu}^{q\times n}$, $\boldsymbol{A} \leftarrow \mathcal{D}_{\lambda}^{n\times n}$, $\boldsymbol{B} \leftarrow U_{q\times n}$, $\boldsymbol{s} \leftarrow \widetilde{\mathsf{Ber}}_{\mu_1}^n$, $\boldsymbol{e} \leftarrow \mathsf{Ber}_{\mu}^n$ and internal coins of the distinguisher.

As the 1KLPN assumption in the KMP-TBE case, we need 1-leaky LPN version of the above lemma.

Lemma C.2. *Let n be a security parameter and let $\mu \in (0, 1/2)$ be any constant. Suppose that $\mathsf{LPN}_{\mu,n}$ problem is $2^{\omega(n^{1/2})}$-hard (for any super-constant hidden by $\omega(\cdot)$.). Then, for every $\mu_1 = \Omega(\lg n/n)$ and $\lambda = \Theta(\lg^2 n)$ such that $2\lambda \leq H_\infty(\widetilde{\mathsf{Ber}}_{\mu_1}^n)$, and every $q = \mathsf{poly}(n)$, we have*

$$(\boldsymbol{S}_0\boldsymbol{c}, \boldsymbol{c}, \boldsymbol{A}, \boldsymbol{S}_0\boldsymbol{A} + \boldsymbol{E}_0) \approx_c (\boldsymbol{r}, \boldsymbol{c}, \boldsymbol{A}, \boldsymbol{S}_0\boldsymbol{A} + \boldsymbol{E}_0),$$

where the probability is take over $\boldsymbol{S}_0 \leftarrow \widetilde{\mathsf{Ber}}_{\mu_1}^{q\times n}$, $\boldsymbol{E}_0 \leftarrow \mathsf{Ber}_{\mu}^{q\times n}$, $\boldsymbol{A} \leftarrow \mathcal{D}_{\lambda}^{n\times n}$, $\boldsymbol{c} \leftarrow \mathbb{F}_2^n$, $\boldsymbol{r} \leftarrow \mathbb{F}_2^q$ and internal coins of the distinguisher.

Proof. We have

$$\begin{aligned}(\boldsymbol{S}_0\boldsymbol{c}, \boldsymbol{c}, \boldsymbol{A}, \boldsymbol{S}_0\boldsymbol{A} + \boldsymbol{E}_0) &\approx_c (\boldsymbol{S}_0\boldsymbol{c}, \boldsymbol{c}, \boldsymbol{A}, \boldsymbol{B}) \\ &\approx_c (\boldsymbol{r}, \boldsymbol{c}, \boldsymbol{A}, \boldsymbol{B}) \\ &\approx_c (\boldsymbol{r}, \boldsymbol{c}, \boldsymbol{A}, \boldsymbol{S}_0\boldsymbol{A} + \boldsymbol{E}_0).\end{aligned}$$

The first transition follows from the proof of Lemma C.1 (Please see the original proof.) The third transition is justified by ignoring leaky part $((\boldsymbol{S}_0\boldsymbol{e}, \boldsymbol{E}_0\boldsymbol{s}), \boldsymbol{e}, \boldsymbol{s})$ in Lemma C.1. In order to show the second one, we consider $(\boldsymbol{S}_0\boldsymbol{c}, \boldsymbol{c})$ and $(\boldsymbol{r}, \boldsymbol{c})$. Recall that each row of $\boldsymbol{S}_0$ is chosen from $\widetilde{\mathsf{Ber}}_{\mu_1}^n$ whose minimum entropy is at least 2λ. Notice that $\mathcal{H} := \{h_{\boldsymbol{c}} : \mathbb{F}_2^n \to \mathbb{F}_2 \mid \boldsymbol{c} \in \mathbb{F}_2, h_{\boldsymbol{c}}(\boldsymbol{x}) = \boldsymbol{x} \cdot \boldsymbol{c}\}$ is universal. Thus, the leftover hash lemma shows that the statistical distance between $(h_{\boldsymbol{c}}(\boldsymbol{s}), \boldsymbol{c})$ and $(u, \boldsymbol{c})$ is $2^{-\Omega(\lambda)}$, which is negligible in κ. Since the leftover hash lemma close the composition, the statistical distance between $(\boldsymbol{S}_0\boldsymbol{c}, \boldsymbol{c})$ and $(\boldsymbol{r}, \boldsymbol{c})$ is still negligible in κ. This completes the proof. □

C.1 The YZ TBE

Let us review the parameter setting:

- A dimension n and $m \geq 2n$.
- A constant $\mu \in (0, 1/10]$.
- A constant $\alpha > 0$.
- Let $\mu_1 = \alpha \lg(n)/n$, $\beta = 1/2 = 1/n^{3\alpha}$, and $\gamma = 1/2 - 1/(2n^{3\alpha/2})$ and choose $\lambda = \Theta(\lg^2 n)$ such that $2\lambda \leq H_\infty(\widetilde{\mathsf{Ber}}^n_{\mu_1})$.
- Two efficient error-correcting codes with generator matrices $\boldsymbol{G} \in \mathbb{F}_2^{q\times n}$ and $\boldsymbol{G}_2 \in \mathbb{F}_2^{\ell\times n}$, where the parameters $q = O(n^{6\alpha+1})$ and $\ell = O(n)$ are adjusted as we can correct up to βq and $2\mu\ell$ errors, respectively.
- a tag space $\mathcal{T} = \mathrm{GF}(2^n) \setminus \{0\}$.

Before giving the YZ TBE scheme, we review its trapdoor generation algorithm and discuss their property, which is similar to that of the KMP TBE scheme. We have field injective homomorphism from $\mathrm{GF}(2^n)$ into $\mathbb{F}_2^{n\times n}$. For finite field elements $\tau \in \mathrm{GF}(2^n)$, we use its companion matrix $\boldsymbol{H}_\tau \in \mathbb{F}_2^{n\times n}$. Let $\boldsymbol{G} \in \mathbb{F}_2^{m\times n}$ be a generator matrix for an efficiently decodable linear code. The trapdoor generation algorithm is defined as follows:

- $\mathsf{Gen_{td}}'(1^\kappa, \tau_0, \tau_1) \to (\boldsymbol{S}_0, \boldsymbol{E}_0, \boldsymbol{S}_1, \boldsymbol{E}_1, (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{C}))$: $\boldsymbol{A} \leftarrow \mathcal{D}_\lambda^{n\times n}$, $\boldsymbol{S}_0, \boldsymbol{S}_1 \leftarrow \widetilde{\mathsf{Ber}}^{q\times n}_{\mu_1}$, and $\boldsymbol{E}_0, \boldsymbol{E}_1 \leftarrow \mathsf{Ber}^{q\times n}_{\mu}$. Compute $\boldsymbol{B}_0 = \boldsymbol{S}_0\boldsymbol{A} + \boldsymbol{E}_0 - \boldsymbol{G}\boldsymbol{H}_{\tau_0} \in \mathbb{F}_2^{q\times n}$ and $\boldsymbol{B}_1 = \boldsymbol{S}_1\boldsymbol{A} + \boldsymbol{E}_1 - \boldsymbol{G}\boldsymbol{H}_{\tau_1} \in \mathbb{F}_2^{q\times n}$. Output $(\boldsymbol{S}_0, \boldsymbol{E}_0, \boldsymbol{S}_1, \boldsymbol{E}_1, (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1))$

$\mathsf{Expt}^{\mathrm{real}}_{\mathsf{Gen_{td}}',\mathcal{A}}(\kappa)$	$\mathsf{Expt}^{\mathrm{corr}}_{\mathsf{Gen_{td}}',\mathcal{A}}(\kappa)$
$(t, \tau_0, \tau_1, \tau', state) \leftarrow \mathcal{A}(1^\kappa)$	$(t, \tau_0, \tau_1, \tau', state) \leftarrow \mathcal{A}(1^\kappa)$
	$\tau'_t := \tau_t; \tau'_{1-t} := \tau'$
$(\boldsymbol{S}_0, \boldsymbol{E}_0, \boldsymbol{S}_1, \boldsymbol{E}_1, (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1)) \leftarrow \mathsf{Gen_{td}}'(1^\kappa, \tau_0, \tau_1)$	$(\boldsymbol{S}_0, \boldsymbol{E}_0, \boldsymbol{S}_1, \boldsymbol{E}_1, (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1)) \leftarrow \mathsf{Gen_{td}}'(1^\kappa, \tau'_0, \tau'_1)$
$\boldsymbol{e} \leftarrow \mathsf{Ber}^n_p; \boldsymbol{S} \leftarrow \widetilde{\mathsf{Ber}}^{q\times n}_{\mu_1}$	$\boldsymbol{e} \leftarrow \mathsf{Ber}^n_p; \boldsymbol{S} \leftarrow \boldsymbol{S}_{1-t}$
$\boldsymbol{s} \leftarrow \widetilde{\mathsf{Ber}}^n_{\mu_1}; \boldsymbol{E} \leftarrow \mathsf{Ber}^{q\times n}_p$	$\boldsymbol{s} \leftarrow \widetilde{\mathsf{Ber}}^n_{\mu_1}; \boldsymbol{E} \leftarrow \boldsymbol{E}_{1-t}$
$d \leftarrow \mathcal{A}(\boldsymbol{S}_t, \boldsymbol{E}_t, (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1), \boldsymbol{e}, \boldsymbol{S}\boldsymbol{e}, \boldsymbol{s}, \boldsymbol{E}\boldsymbol{s}, state)$	$d \leftarrow \mathcal{A}(\boldsymbol{S}_t, \boldsymbol{E}_t, (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1), \boldsymbol{e}, \boldsymbol{S}\boldsymbol{e}, \boldsymbol{s}, \boldsymbol{E}\boldsymbol{s}, state)$
return d	**return** d

Fig. 6. Games for Trapdoor Generation Algorithm

Yu and Zhang [YZ16] showed the following lemma which is similar to Lemma B.1 by invoking Lemma C.1 twice. We will use this lemma in the security proof.

Lemma C.3 (Adapted, [YZ16, Lemmas 5.3, 5.4, and 5.5]). *For every adversary $\mathcal{A}$, there exists two adversaries $\mathcal{A}_0$ and $\mathcal{A}_1$ such that*

$$\left|\Pr[\mathsf{Expt}^{\mathrm{real}}_{\mathsf{Gen_{td}}',\mathcal{A}}(\kappa) = 1] - \Pr[\mathsf{Expt}^{\mathrm{corr}}_{\mathsf{Gen_{td}}',\mathcal{A}}(\kappa) = 1]\right| \leq \mathsf{Adv}_{\mathsf{leakyLPN},\mathcal{A}_0}(\kappa) + \mathsf{Adv}_{\mathsf{leakyLPN},\mathcal{A}_1}(\kappa),$$

where $\mathsf{Expt}^{\mathrm{real}}_{\mathsf{Gen_{td}}',\mathcal{A}}(\kappa)$ *and* $\mathsf{Expt}^{\mathrm{corr}}_{\mathsf{Gen_{td}}',\mathcal{A}}(\kappa)$ *are defined in Figure 6.*

The YZ TBE scheme $(\mathsf{Gen_{YZ}}, \mathsf{Enc_{YZ}}, \mathsf{Dec_{YZ}})$ is defined as follows:

- $\mathsf{Gen_{YZ}}(1^\kappa) \to (ek, dk)$: $(\boldsymbol{S}_0, \boldsymbol{E}_0, \boldsymbol{S}_1, \boldsymbol{E}_1, (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1)) \leftarrow \mathsf{Gen_{td}}'(1^\kappa, 0, 0)$. (We note that $\boldsymbol{B}_i = \boldsymbol{S}_i\boldsymbol{A} + \boldsymbol{E}_i \in \mathbb{F}_2^{q\times n}$.) $\boldsymbol{C} \leftarrow \mathcal{D}_\lambda^{\ell\times n}$. Output

$$ek = (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{C}) \in (\mathbb{F}_2^{m\times n})^3 \times \mathbb{F}_2^{\ell\times n},$$
$$dk = (0, \boldsymbol{S}_0, ek) \in \mathrm{GF}(2^n) \times \mathbb{F}_2^{q\times n} \times \{0,1\}^*.$$

Table 3. Summary of Games for the Proof of Theorem C.1:

Game	$\mathsf{Gen}_{\mathsf{td}}$	dk	$\boldsymbol{c}^*$	$\boldsymbol{c}_0^*$	$\boldsymbol{c}_1^*$	$\boldsymbol{c}_2^*$
Game_0	$(0,0)$	$(0,\boldsymbol{S}_0)$	$\boldsymbol{A}\boldsymbol{s}^*+\boldsymbol{e}^*$	$(\boldsymbol{G}\boldsymbol{H}_{\tau^*}+\boldsymbol{B}_0)\boldsymbol{s}^*+\boldsymbol{S}_0^*\boldsymbol{e}_1^*-\boldsymbol{E}_0^*\boldsymbol{s}^*$	$(\boldsymbol{G}\boldsymbol{H}_{\tau^*}+\boldsymbol{B}_1)\boldsymbol{s}^*+\boldsymbol{S}_1^*\boldsymbol{e}_1^*-\boldsymbol{E}_1^*\boldsymbol{s}^*$	$\boldsymbol{C}\boldsymbol{s}^*+\boldsymbol{e}_2^*\boldsymbol{G}_2(\mu)$
Game_1	$(0,\tau^*)$	$(0,\boldsymbol{S}_0)$	$\boldsymbol{A}\boldsymbol{s}^*+\boldsymbol{e}^*$	$(\boldsymbol{G}\boldsymbol{H}_{\tau^*}+\boldsymbol{B}_0)\boldsymbol{s}^*+\boldsymbol{S}_0^*\boldsymbol{e}_1^*-\boldsymbol{E}_0^*\boldsymbol{s}^*$	$\boldsymbol{S}_1\boldsymbol{c}^*$	$\boldsymbol{C}\boldsymbol{s}^*+\boldsymbol{e}_2^*\boldsymbol{G}_2(\mu)$
Game_2	$(0,\tau^*)$	$(\tau^*,\boldsymbol{S}_1)$	$\boldsymbol{A}\boldsymbol{s}^*+\boldsymbol{e}^*$	$(\boldsymbol{G}\boldsymbol{H}_{\tau^*}+\boldsymbol{B}_0)\boldsymbol{s}^*+\boldsymbol{S}_0^*\boldsymbol{e}_1^*-\boldsymbol{E}_0^*\boldsymbol{s}^*$	$\boldsymbol{S}_1\boldsymbol{c}^*$	$\boldsymbol{C}\boldsymbol{s}^*+\boldsymbol{e}_2^*\boldsymbol{G}_2(\mu)$
Game_3	(τ^*,τ^*)	$(\tau^*,\boldsymbol{S}_1)$	$\boldsymbol{A}\boldsymbol{s}^*+\boldsymbol{e}^*$	$\boldsymbol{S}_0\boldsymbol{c}^*$	$\boldsymbol{S}_1\boldsymbol{c}^*$	$\boldsymbol{C}\boldsymbol{s}^*+\boldsymbol{e}_2^*\boldsymbol{G}_2(\mu)$
Game_4	(τ^*,τ^*)	$(\tau^*,\boldsymbol{S}_1)$	$U(\mathbb{F}_2^n)$	$\boldsymbol{S}_0\boldsymbol{c}^*$	$\boldsymbol{S}_1\boldsymbol{c}^*$	$U(\mathbb{F}_2^\ell)$
Game_5	(τ^*,τ^*)	$(\tau^*,\boldsymbol{S}_1)$	$U(\mathbb{F}_2^n)$	$U(\mathbb{F}_2^q)$	$\boldsymbol{S}_1\boldsymbol{c}^*$	$U(\mathbb{F}_2^\ell)$
Game_6	$(0,\tau^*)$	$(\tau^*,\boldsymbol{S}_1)$	$U(\mathbb{F}_2^n)$	$U(\mathbb{F}_2^q)$	$\boldsymbol{S}_1\boldsymbol{c}^*$	$U(\mathbb{F}_2^\ell)$
Game_7	$(0,\tau^*)$	$(0,\boldsymbol{S}_0)$	$U(\mathbb{F}_2^n)$	$U(\mathbb{F}_2^q)$	$\boldsymbol{S}_1\boldsymbol{c}^*$	$U(\mathbb{F}_2^\ell)$
Game_8	$(0,\tau^*)$	$(0,\boldsymbol{S}_0)$	$U(\mathbb{F}_2^n)$	$U(\mathbb{F}_2^q)$	$U(\mathbb{F}_2^q)$	$U(\mathbb{F}_2^\ell)$
Game_9	$(0,0)$	$(0,\boldsymbol{S}_0)$	$U(\mathbb{F}_2^n)$	$U(\mathbb{F}_2^q)$	$U(\mathbb{F}_2^q)$	$U(\mathbb{F}_2^\ell)$

- $\mathsf{Enc}_{\mathsf{YZ}}(ek,\tau,\mu)\to ct=(\boldsymbol{c},\boldsymbol{c}_0,\boldsymbol{c}_1,\boldsymbol{c}_2)$: Generate $\boldsymbol{s}\leftarrow\widetilde{\mathsf{Ber}}_{\mu_1}^n$, $\boldsymbol{e}_1\leftarrow\mathsf{Ber}_\mu^n$, $\boldsymbol{e}_2\leftarrow\mathsf{Ber}_\mu^\ell$, $\boldsymbol{S}_0',\boldsymbol{S}_1'\leftarrow\widetilde{\mathsf{Ber}}_{\mu_1}^{q\times n}$, and $\boldsymbol{E}_0',\boldsymbol{E}_1'\leftarrow\mathsf{Ber}_\mu^{q\times n}$. Compute
$$\begin{aligned}\boldsymbol{c}&:=\boldsymbol{A}\boldsymbol{s}+\boldsymbol{e}_1\\ \boldsymbol{c}_0&:=(\boldsymbol{G}\boldsymbol{H}_\tau+\boldsymbol{B}_0)\boldsymbol{s}+\boldsymbol{S}_0'\boldsymbol{e}_1-\boldsymbol{E}_0'\boldsymbol{s}\\ \boldsymbol{c}_1&:=(\boldsymbol{G}\boldsymbol{H}_\tau+\boldsymbol{B}_1)\boldsymbol{s}+\boldsymbol{S}_1'\boldsymbol{e}_1-\boldsymbol{E}_1'\boldsymbol{s}\\ \boldsymbol{c}_2&:=\boldsymbol{C}\boldsymbol{s}+\boldsymbol{e}_2+\boldsymbol{G}_2(\mu)\end{aligned}$$
and output $ct=(\boldsymbol{c},\boldsymbol{c}_0,\boldsymbol{c}_1,\boldsymbol{c}_2)\in\mathbb{F}_2^m\times\mathbb{F}_2^m\times\mathbb{F}_2^m\times\mathbb{F}_2^\ell$.
- $\mathsf{Dec}_{\mathsf{YZ}}(dk,\tau,ct)\to\mu/\bot$: Parse $dk=(\tau_b,\boldsymbol{S}_b,ek)$ and compute
$$\tilde{c}_b:=\boldsymbol{c}_b-\boldsymbol{S}_b\boldsymbol{c},$$
which is $\boldsymbol{G}\boldsymbol{H}_{\tau-\tau_b}\boldsymbol{s}+(\boldsymbol{S}_b'-\boldsymbol{S}_b)\boldsymbol{e}_1+(\boldsymbol{E}_b-\boldsymbol{E}_b')\boldsymbol{s}$ if the ciphertext is correctly computed. Reconstruct $\boldsymbol{b}=\boldsymbol{H}_{\tau-\tau_b}\boldsymbol{s}$ from $\tilde{c}_b$ with error $(\boldsymbol{S}_b'-\boldsymbol{S}_b)\boldsymbol{e}_1+(\boldsymbol{E}_b-\boldsymbol{E}_b)\boldsymbol{s}$ by using the decoding algorithm of $\boldsymbol{G}$. Compute $\boldsymbol{s}=\boldsymbol{H}_{\tau-\tau_b}^{-1}\cdot\boldsymbol{b}$. If
$$\mathsf{HW}(\boldsymbol{c}-\boldsymbol{A}\boldsymbol{s})\le 2\mu n\wedge\mathsf{HW}(\boldsymbol{c}_0-(\boldsymbol{G}\boldsymbol{H}_\tau+\boldsymbol{B}_0)\boldsymbol{s})\le\gamma q\wedge\mathsf{HW}(\boldsymbol{c}_1-(\boldsymbol{G}\boldsymbol{H}_\tau+\boldsymbol{B}_1)\boldsymbol{s})\le\gamma q$$
hold, then compute $\boldsymbol{c}_2-\boldsymbol{C}\boldsymbol{s}=\boldsymbol{G}_2(\mu)+\boldsymbol{e}_2$ and reconstruct μ by using the decoding algorithm of $\boldsymbol{G}_2$ and output it. Otherwise, output $\bot$.

As Kiltz et al. showed the key-switching lemma, Yu and Zhang also showed their key-switching lemma as follows:

Lemma C.4 ([YZ16, Section 5.2.1]). *Let $\mathsf{Dec}_{\mathsf{YZ}0}$ be $\mathsf{Dec}_{\mathsf{YZ}}$ that uses $\boldsymbol{c}_0$ to extract $\boldsymbol{s}$ and let and $\mathsf{Dec}_{\mathsf{YZ}1}$ be $\mathsf{Dec}_{\mathsf{YZ}}$ that uses $\boldsymbol{c}_1$ to extract $\boldsymbol{s}$. Then, with overwhelming probability over the choice of the encryption and decryption keys, $\mathsf{Dec}_{\mathsf{YZ}0}$ and $\mathsf{Dec}_{\mathsf{YZ}1}$ have the same output distribution.*

Now, we are ready to show that $\mathsf{TBE}_{\mathsf{YZ}}$ is OS-ST-wCCA-secure as follows:

Theorem C.1. *$\mathsf{TBE}_{\mathsf{YZ}}$ is OS-ST-wCCA-secure if $\mathsf{LPN}_{\mu,n}$ is $2^{\omega(2^{1/2})}$-hard.*

We mainly follow the definitions of games in the original paper but we adopt the notions in KMP. We summarize the games in Table 3

Game_0*:* This is the original game with $b=0$ expanded as follows:

1. The challenger runs the adversary on input 1^κ.
2. The adversary outputs τ^*. The challenger generates keys by $(\boldsymbol{S}_0,\boldsymbol{E}_0,\boldsymbol{S}_1,\boldsymbol{E}_1,(\boldsymbol{A},\boldsymbol{B}_0,\boldsymbol{B}_1))\leftarrow\mathsf{Gen}_{\mathsf{td}}'(1^\kappa,0,0)$ and $\boldsymbol{C}\leftarrow\mathcal{D}_\lambda^{\ell\times n}$. It sets $ek=(\boldsymbol{A},\boldsymbol{B}_0,\boldsymbol{B}_1,\boldsymbol{C})$ and $dk=(0,\boldsymbol{S}_0,ek)$. It runs the adversary on input ek and simulates the decryption oracle by using dk.
3. The adversary outputs μ. The challenger generates the challenge ciphertext as follows: It generates
   - $\boldsymbol{s}^*\leftarrow\widetilde{\mathsf{Ber}}_{\mu_1}^n$, $\boldsymbol{e}_1^*\leftarrow\mathsf{Ber}_\mu^n$, $\boldsymbol{e}_2^*\leftarrow\mathsf{Ber}_\mu^\ell$, $\boldsymbol{S}_0^*\leftarrow\widetilde{\mathsf{Ber}}_{\mu_1}^{q\times n}$, $\boldsymbol{E}_0^*\leftarrow\mathsf{Ber}_\mu^{q\times n}$, $\boldsymbol{S}_1^*\leftarrow\widetilde{\mathsf{Ber}}_{\mu_1}^{q\times n}$, $\boldsymbol{E}_1^*\leftarrow\mathsf{Ber}_\mu^{q\times n}$

- $\boldsymbol{c}^* := \boldsymbol{A}\boldsymbol{s}^* + \boldsymbol{e}_1^*$
- $\boldsymbol{c}_0^* := (\boldsymbol{G}\boldsymbol{H}_{\tau^*} + \boldsymbol{B}_0)\boldsymbol{s}^* + \boldsymbol{S}_0^*\boldsymbol{e}_1^* - \boldsymbol{E}_0^*\boldsymbol{s}^*$
- $\boldsymbol{c}_1^* := (\boldsymbol{G}\boldsymbol{H}_{\tau^*} + \boldsymbol{B}_1)\boldsymbol{s}^* + \boldsymbol{S}_1^*\boldsymbol{e}_1^* - \boldsymbol{E}_1^*\boldsymbol{s}^*$
- $\boldsymbol{c}_2^* := \boldsymbol{C}\boldsymbol{s}^* + \boldsymbol{e}_2^* + \boldsymbol{G}_2(\mu)$.

It returns $ct^* = (\boldsymbol{c}^*, \boldsymbol{c}_0^*, \boldsymbol{c}_1^*, \boldsymbol{c}_2^*)$. It runs the adversary on input ct^* and simulates the decryption oracle by using dk.

4. Finally, the adversary outputs its guess b' and the challenger outputs b'.

Apparently, we have

$$\Pr[S_0] = \Pr[\mathsf{Expt}^{\text{pr-st-wcca},0}_{\mathsf{TBE}_{\mathsf{YZ}},\mathcal{A}}(\kappa) = 1].$$

Game_1*: This is the original game with $b = 0$ expanded as follows:*

1. The challenger runs the adversary on input 1^κ.
2. The adversary outputs τ^*. The challenger generates keys by $(\boldsymbol{S}_0, \boldsymbol{E}_0, \boldsymbol{S}_1, \boldsymbol{E}_1, (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1)) \leftarrow \underline{\mathsf{Gen}_{\mathsf{td}}{}'(1^\kappa, 0, \tau^*)}$ and $\boldsymbol{C} \leftarrow \mathcal{D}_\lambda^{\ell\times n}$. It sets $ek = (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{C})$ and $dk = (0, \boldsymbol{S}_0, ek)$. It runs the adversary on input ek and simulates the decryption oracle by using dk.
3. The adversary outputs μ. The challenger generates the challenge ciphertext as follows: It generates
   - $\boldsymbol{s}^* \leftarrow \widetilde{\mathsf{Ber}}^n_{\mu_1}$, $\boldsymbol{e}_1^* \leftarrow \mathsf{Ber}^n_\mu$, $\boldsymbol{e}_2^* \leftarrow \mathsf{Ber}^\ell_\mu$, $\boldsymbol{S}_0^* \leftarrow \widetilde{\mathsf{Ber}}^{q\times n}_{\mu_1}$, $\boldsymbol{E}_0^* \leftarrow \mathsf{Ber}^{q\times n}_\mu$
   - $\boldsymbol{c}^* := \boldsymbol{A}\boldsymbol{s}^* + \boldsymbol{e}_1^*$
   - $\boldsymbol{c}_0^* := (\boldsymbol{G}\boldsymbol{H}_{\tau^*} + \boldsymbol{B}_0)\boldsymbol{s}^* + \boldsymbol{S}_0^*\boldsymbol{e}_1^* - \boldsymbol{E}_0^*\boldsymbol{s}^*$
   - $\underline{\boldsymbol{c}_1^* := \boldsymbol{S}_1\boldsymbol{c}^*}$ $(= (\boldsymbol{G}\boldsymbol{H}_{\tau^*} + \boldsymbol{B}_1')\boldsymbol{s}^* + \boldsymbol{S}_1\boldsymbol{e}_1^* - \boldsymbol{E}_1\boldsymbol{s}^*)$
   - $\boldsymbol{c}_2^* := \boldsymbol{C}\boldsymbol{s}^* + \boldsymbol{e}_2^* + \boldsymbol{G}_2(\mu)$.

   It returns $ct^* = (\boldsymbol{c}^*, \boldsymbol{c}_0^*, \boldsymbol{c}_1^*, \boldsymbol{c}_2^*)$. It runs the adversary on input ct^* and simulates the decryption oracle by using dk.
4. Finally, the adversary outputs its guess b' and the challenger outputs b'.

We can use Lemma C.3 to bound the distance between Game_0 and Game_1.

Lemma C.5 (Adapted, [YZ16, Lemmas 5.3, 5.4, and 5.5]). *There exists an adversary $\mathcal{A}_{01}$ satisfying*

$$|\Pr[S_0] - \Pr[S_1]| \le \left|\Pr[\mathsf{Expt}^{\mathrm{real}}_{\mathsf{Gen}_{\mathsf{td}}{}',\mathcal{A}_{01}}(\kappa) = 1] - \Pr[\mathsf{Expt}^{\mathrm{corr}}_{\mathsf{Gen}_{\mathsf{td}}{}',\mathcal{A}_{01}}(\kappa) = 1]\right|.$$

Game_2*: We next switch the decryption key from $(0, \boldsymbol{S}_0)$ to $(\tau^*, \boldsymbol{S}_1)$.*

1. The challenger runs the adversary on input 1^κ.
2. The adversary outputs τ^*. The challenger generates keys by $(\boldsymbol{S}_0, \boldsymbol{E}_0, \boldsymbol{S}_1, \boldsymbol{E}_1, (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1)) \leftarrow \mathsf{Gen}_{\mathsf{td}}{}'(1^\kappa, 0, \tau^*)$ and $\boldsymbol{C} \leftarrow \mathcal{D}_\lambda^{\ell\times n}$. It sets $ek = (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{C})$ and $\underline{dk = (\tau^*, \boldsymbol{S}_1, ek)}$. It runs the adversary on input ek and simulates the decryption oracle by using dk.
3. The adversary outputs μ. The challenger generates the challenge ciphertext as follows: It generates
   - $\boldsymbol{s}^* \leftarrow \widetilde{\mathsf{Ber}}^n_{\mu_1}$, $\boldsymbol{e}_1^* \leftarrow \mathsf{Ber}^n_\mu$, $\boldsymbol{e}_2^* \leftarrow \mathsf{Ber}^\ell_\mu$, $\boldsymbol{S}_0^* \leftarrow \widetilde{\mathsf{Ber}}^{q\times n}_{\mu_1}$, $\boldsymbol{E}_0^* \leftarrow \mathsf{Ber}^{q\times n}_\mu$
   - $\boldsymbol{c}^* := \boldsymbol{A}\boldsymbol{s}^* + \boldsymbol{e}_1^*$
   - $\boldsymbol{c}_0^* := (\boldsymbol{G}\boldsymbol{H}_{\tau^*} + \boldsymbol{B}_0)\boldsymbol{s}^* + \boldsymbol{S}_0^*\boldsymbol{e}_1^* - \boldsymbol{E}_0^*\boldsymbol{s}^*$
   - $\boldsymbol{c}_1^* := \boldsymbol{S}_1\boldsymbol{c}^*$
   - $\boldsymbol{c}_2^* := \boldsymbol{C}\boldsymbol{s}^* + \boldsymbol{e}_2^* + \boldsymbol{G}_2(\mu)$.

   It returns $ct^* = (\boldsymbol{c}^*, \boldsymbol{c}_0^*, \boldsymbol{c}_1^*, \boldsymbol{c}_2^*)$. It runs the adversary on input ct^* and simulates the decryption oracle by using dk.
4. Finally, the adversary outputs its guess b' and the challenger outputs b'.

Yu and Zhang showed the following lemma by using the key-switching lemma Lemma C.4:

Lemma C.6 (Adapted, [YZ16, Lemma 5.6]). *We have*

$$|\Pr[S_1] - \Pr[S_2]| \le \mathsf{negl}(\kappa).$$

Game_3: We next modify $\boldsymbol{c}_0^*$:

1. The challenger runs the adversary on input 1^κ.
2. The adversary outputs τ^*. The challenger generates keys by $(\boldsymbol{S}_0, \boldsymbol{E}_0, \boldsymbol{S}_1, \boldsymbol{E}_1, (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1)) \leftarrow \underline{\mathsf{Gen}_{\mathsf{td}}'(1^\kappa, \tau^*, \tau^*)}$ and $\boldsymbol{C} \leftarrow \mathcal{D}_\lambda^{\ell \times n}$. It sets $ek = (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{C})$ and $dk = (\tau^*, \boldsymbol{S}_1, ek)$. It runs the adversary on input ek and simulates the decryption oracle by using dk.
3. The adversary outputs μ. The challenger generates the challenge ciphertext as follows: It generates
   - $\boldsymbol{s}^* \leftarrow \widetilde{\mathsf{Ber}}_{\mu_1}^n$, $\boldsymbol{e}_1^* \leftarrow \mathsf{Ber}_\mu^n$, $\boldsymbol{e}_2^* \leftarrow \mathsf{Ber}_\mu^\ell$
   - $\boldsymbol{c}^* := \boldsymbol{A}\boldsymbol{s}^* + \boldsymbol{e}_1^*$
   - $\underline{\boldsymbol{c}_0^* := \boldsymbol{S}_0\boldsymbol{c}^*}$ $(=(\boldsymbol{G}\boldsymbol{H}_{\tau^*} + \boldsymbol{B}_0)\boldsymbol{s}^* + \boldsymbol{S}_0^*\boldsymbol{e}_1^* - \boldsymbol{E}_0^*\boldsymbol{s}^*)$
   - $\boldsymbol{c}_1^* := \boldsymbol{S}_1\boldsymbol{c}^*$
   - $\boldsymbol{c}_2^* := \boldsymbol{C}\boldsymbol{s}^* + \boldsymbol{e}_2^* + \boldsymbol{G}_2(\mu)$.

   It returns $ct^* = (\boldsymbol{c}^*, \boldsymbol{c}_0^*, \boldsymbol{c}_1^*, \boldsymbol{c}_2^*)$. It runs the adversary on input ct^* and simulates the decryption oracle by using dk.
4. Finally, the adversary outputs its guess b' and the challenger outputs b'.

We can use Lemma C.3 to bound the distance between Game_2 and Game_3.

Lemma C.7 (Adapted, [YZ16, Lemma 5.7]). *There exists an adversary $\mathcal{A}_{23}$ satisfying*

$$|\Pr[S_2] - \Pr[S_3]| \leq \left|\Pr[\mathsf{Expt}^{\mathsf{real}}_{\mathsf{Gen}_{\mathsf{td}}', \mathcal{A}_{23}}(\kappa) = 1] - \Pr[\mathsf{Expt}^{\mathsf{corr}}_{\mathsf{Gen}_{\mathsf{td}}', \mathcal{A}_{23}}(\kappa) = 1]\right|.$$

Game_4: We next replace two components $\boldsymbol{c}^*$ and $\boldsymbol{c}_2^*$ of the challenge ciphertext with random ones:

1. The challenger runs the adversary on input 1^κ.
2. The adversary outputs τ^*. The challenger generates keys by $(\boldsymbol{S}_0, \boldsymbol{E}_0, \boldsymbol{S}_1, \boldsymbol{E}_1, (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1)) \leftarrow \mathsf{Gen}_{\mathsf{td}}'(1^\kappa, \tau^*, \tau^*)$ and $\boldsymbol{C} \leftarrow \mathcal{D}_\lambda^{\ell \times n}$. It sets $ek = (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{C})$ and $dk = (\tau^*, \boldsymbol{S}_1, ek)$. It runs the adversary on input ek and simulates the decryption oracle by using dk.
3. The adversary outputs μ. The challenger generates the challenge ciphertext as follows: It generates
   - $\underline{\boldsymbol{c}^* \leftarrow \mathbb{F}_2^n}$
   - $\boldsymbol{c}_0^* := \boldsymbol{S}_0\boldsymbol{c}^*$
   - $\boldsymbol{c}_1^* := \boldsymbol{S}_1\boldsymbol{c}^*$
   - $\underline{\boldsymbol{c}_2^* \leftarrow \mathbb{F}_2^\ell}$.

   It returns $ct^* = (\boldsymbol{c}^*, \boldsymbol{c}_0^*, \boldsymbol{c}_1^*, \boldsymbol{c}_2^*)$. It runs the adversary on input ct^* and simulates the decryption oracle by using dk.
4. Finally, the adversary outputs its guess b' and the challenger outputs b'.

Using Lemma C.1, Yu and Zhang showed the following lemma.

Lemma C.8 (Adapted, [YZ16, Lemma 5.8]). *There exists an adversary $\mathcal{A}_{34}$ satisfying*

$$|\Pr[S_3] - \Pr[S_4]| \leq \mathsf{Adv}_{\mathsf{leakyLPN}, \mathcal{A}_{34}}(\kappa).$$

This is the final game of the original proof. We continue modifying the games in order to make $\boldsymbol{c}_0^*$ and $\boldsymbol{c}_1^*$ random.

Game_5: We next replace $\boldsymbol{c}_0^*$ with random one:

1. The challenger runs the adversary on input 1^κ.
2. The adversary outputs τ^*. The challenger generates keys by $(\boldsymbol{S}_0, \boldsymbol{E}_0, \boldsymbol{S}_1, \boldsymbol{E}_1, (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1)) \leftarrow \mathsf{Gen}_{\mathsf{td}}'(1^\kappa, \tau^*, \tau^*)$ and $\boldsymbol{C} \leftarrow \mathcal{D}_\lambda^{\ell \times n}$. It sets $ek = (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{C})$ and $dk = (\tau^*, \boldsymbol{S}_1, ek)$. It runs the adversary on input ek and simulates the decryption oracle by using dk.
3. The adversary outputs μ. The challenger generates the challenge ciphertext as follows: It generates
   - $\boldsymbol{c}^* \leftarrow \mathbb{F}_2^n$
   - $\underline{\boldsymbol{c}_0 \leftarrow \mathbb{F}_2^q}$
   - $\boldsymbol{c}_1^* := \boldsymbol{S}_1\boldsymbol{c}^*$
   - $\boldsymbol{c}_2^* \leftarrow \mathbb{F}_2^\ell$.

   It returns $ct^* = (\boldsymbol{c}^*, \boldsymbol{c}_0^*, \boldsymbol{c}_1^*, \boldsymbol{c}_2^*)$. It runs the adversary on input ct^* and simulates the decryption oracle by using $dk = (\tau^*, \boldsymbol{S}_1, ek)$.
4. Finally, the adversary outputs its guess b' and the challenger outputs b'.

Lemma C.9. *There exists an adversary $\mathcal{A}_{45}$ satisfying*

$$|\Pr[S_4] - \Pr[S_5]| \leq \mathsf{Adv}_{1\mathsf{leakyLPN}, \mathcal{A}_{45}}(\kappa).$$

The proof is very similar to Lemma B.8.

Proof. We construct $\mathcal{A}_{45}$ as follows:

1. $\mathcal{A}_{45}$ is given $(z, \boldsymbol{c}, \boldsymbol{A}, \boldsymbol{S}_0\boldsymbol{A} + \boldsymbol{E}_0)$, where $z = \boldsymbol{S}_0\boldsymbol{c}$ or $\boldsymbol{r} \leftarrow \mathbb{F}_2^q$. It then invokes $\mathcal{A}$ on input 1^κ and receives τ^*. It generates
   - $\boldsymbol{B}_0 := \boldsymbol{S}_0\boldsymbol{A} + \boldsymbol{E}_0 - \boldsymbol{G}\boldsymbol{H}_{\tau^*}$
   - $\boldsymbol{S}_1 \leftarrow \widetilde{\mathsf{Ber}}_{\mu_1}^{q\times n}$, $\boldsymbol{E}_1 \leftarrow \mathsf{Ber}_{\mu}^{q\times n}$, $\boldsymbol{B}_1 := \boldsymbol{S}_1\boldsymbol{A} + \boldsymbol{E}_1 - \boldsymbol{G}\boldsymbol{H}_{\tau^*}$
   - $\boldsymbol{C} \leftarrow \mathcal{D}_\lambda^{\ell\times n}$.

   It sets $dk = (\tau^*, \boldsymbol{S}_1, ek)$ and $ek = (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{C})$. It runs the adversary on input ek and simulates the decryption oracle by using dk.
2. The adversary outputs μ. $\mathcal{A}_{45}$ generates the challenge ciphertext as follows: It generates
   - $\boldsymbol{c}^* := \boldsymbol{c}$, $\boldsymbol{c}_0^* := z$, $\boldsymbol{c}_1^* := \boldsymbol{S}_1\boldsymbol{c}^*$, and $\boldsymbol{c}_2^* \leftarrow \mathbb{F}_2^\ell$.

   It returns $ct^* = (\boldsymbol{c}^*, \boldsymbol{c}_0^*, \boldsymbol{c}_1^*, \boldsymbol{c}_2^*)$. It runs the adversary on input ct^* and simulates the decryption oracle by using $dk = (\tau^*, \boldsymbol{S}_1, ek)$.
3. Finally, the adversary outputs its guess b' and $\mathcal{A}_{45}$ outputs b'.

If $z = \boldsymbol{S}_0\boldsymbol{c}$, then $\mathcal{A}_{45}$ perfectly simulates Game_4. If $z = \boldsymbol{r} \leftarrow \mathbb{F}_2^q$, then $\mathcal{A}_{45}$ perfectly simulates Game_5. Thus, the lemma follows. □

Game_6*:* We next change how to generate trapdoor:

1. The challenger runs the adversary on input 1^κ.
2. The adversary outputs τ^*. The challenger generates keys by $(\boldsymbol{S}_0, \boldsymbol{E}_0, \boldsymbol{S}_1, \boldsymbol{E}_1, (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1)) \leftarrow \underline{\mathsf{Gen}_{\mathsf{td}}{}'(1^\kappa, 0, \tau^*)}$ and $\boldsymbol{C} \leftarrow \mathcal{D}_\lambda^{\ell\times n}$. It sets $ek = (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{C})$ and $dk = (\tau^*, \boldsymbol{S}_1, ek)$. It runs the adversary on input ek and simulates the decryption oracle by using dk.
3. The adversary outputs μ. The challenger generates the challenge ciphertext as follows: It generates
   - $\boldsymbol{c}^* \leftarrow \mathbb{F}_2^n$
   - $\boldsymbol{c}_0 \leftarrow \mathbb{F}_2^q$
   - $\boldsymbol{c}_1^* := \boldsymbol{S}_1\boldsymbol{c}^*$
   - $\boldsymbol{c}_2^* \leftarrow \mathbb{F}_2^\ell$.

   It returns $ct^* = (\boldsymbol{c}^*, \boldsymbol{c}_0^*, \boldsymbol{c}_1^*, \boldsymbol{c}_2^*)$. It runs the adversary on input ct^* and simulates the decryption oracle by using dk.
4. Finally, the adversary outputs its guess b' and the challenger outputs b'.

Lemma C.10. *There exists an adversary $\mathcal{A}_{56}$ satisfying*

$$|\Pr[S_5] - \Pr[S_6]| \le \left|\Pr[\mathsf{Expt}^{\mathsf{real}}_{\mathsf{Gen}_{\mathsf{td}}{}', \mathcal{A}_{56}}(\kappa) = 1] - \Pr[\mathsf{Expt}^{\mathsf{corr}}_{\mathsf{Gen}_{\mathsf{td}}{}', \mathcal{A}_{56}}(\kappa) = 1]\right|.$$

Proof. We construct $\mathcal{A}_{56}$ that distinguishes real and corr games as follows:

1. Given 1^κ, $\mathcal{A}_{56}$ runs $\mathcal{A}$ on input 1^κ and receives τ^*.
2. It sends $(1, \tau^*, \tau^*, 0)$ to its challenger and receives $(\boldsymbol{S}_1, \boldsymbol{E}_1, \boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{e}, \boldsymbol{S}\boldsymbol{e}, \boldsymbol{s}, \boldsymbol{E}\boldsymbol{s})$. It chooses $\boldsymbol{C} \leftarrow \mathcal{D}_\lambda^{\ell\times n}$. It sets $dk = (\tau^*, \boldsymbol{S}_1)$ and $ek = (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{C})$. It runs $\mathcal{A}$ on input ek.
3. $\mathcal{A}_{56}$ simulates the decryption oracle using dk.
4. $\mathcal{A}_{56}$ generates the challenge on a query μ from $\mathcal{A}$ as follows: It generates $\boldsymbol{c}^* \leftarrow \mathbb{F}_2^n$, $\boldsymbol{c}_0^* \leftarrow \mathbb{F}_2^q$, and $\boldsymbol{c}_1^* := \boldsymbol{S}_1\boldsymbol{c}^*$. It chooses $\boldsymbol{c}_2^* \leftarrow \mathbb{F}_2^\ell$ and returns $ct^* := (\boldsymbol{c}^*, \boldsymbol{c}_0^*, \boldsymbol{c}_1^*, \boldsymbol{c}_2^*)$.
5. $\mathcal{A}'_{\mathsf{Gen}_{\mathsf{td}}}$ outputs b' if $\mathcal{A}$ finally outputs b'.

Notice that if the game is real and corr, then the keys are generated by $\mathsf{Gen}_{\mathsf{td}}(1^\kappa, \tau^*, \tau^*)$ and $\mathsf{Gen}_{\mathsf{td}}(1^\kappa, 0, \tau^*)$, respectively. Thus, if the game is real and the keys are generated by $\mathsf{Gen}_{\mathsf{td}}(1^\kappa, \tau^*, \tau^*)$, then $\mathcal{A}_{56}$ perfectly simulates Game_5. If the game is corr and keys are generated by $\mathsf{Gen}_{\mathsf{td}}(1^\kappa, 0, \tau^*)$, then $\mathcal{A}_{56}$ perfectly simulates Game_6. This completes the proof. □

Game_7*:* We then switch the decapsulation key.

1. The challenger runs the adversary on input 1^κ.
2. The adversary outputs τ^*. The challenger generates keys by $(\boldsymbol{S}_0, \boldsymbol{E}_0, \boldsymbol{S}_1, \boldsymbol{E}_1, (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1)) \leftarrow \mathsf{Gen}_{\mathsf{td}}{}'(1^\kappa, 0, \tau^*)$ and $\boldsymbol{C} \leftarrow \mathcal{D}_\lambda^{\ell\times n}$. It sets $ek = (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{C})$ and $\underline{dk = (0, \boldsymbol{S}_0, ek)}$. It runs the adversary on input ek and simulates the decryption oracle by using dk.
3. The adversary outputs μ. The challenger generates the challenge ciphertext as follows: It generates
   - $\boldsymbol{c}^* \leftarrow \mathbb{F}_2^n$

- $\boldsymbol{c}_0 \leftarrow \mathbb{F}_2^q$
- $\boldsymbol{c}_1^* := \boldsymbol{S}_1\boldsymbol{c}^*$
- $\boldsymbol{c}_2^* \leftarrow \mathbb{F}_2^\ell$.

It returns $ct^* = (\boldsymbol{c}^*, \boldsymbol{c}_0^*, \boldsymbol{c}_1^*, \boldsymbol{c}_2^*)$. It runs the adversary on input ct^* and simulates the decryption oracle by using dk.

4. Finally, the adversary outputs its guess b' and the challenger outputs b'.

Following the key-switching lemma Lemma C.4, we have the following lemma as Lemma C.6:

Lemma C.11. *We have*

$$|\Pr[S_6] - \Pr[S_7]| \le \mathsf{negl}(\kappa).$$

Game$_8$*:* We then replace $\boldsymbol{c}_1^* := \boldsymbol{S}_1\boldsymbol{c}^*$ with $\boldsymbol{c}_1^* \leftarrow \mathbb{F}_2^q$:

1. The challenger runs the adversary on input 1^κ.
2. The adversary outputs τ^*. The challenger generates keys by $(\boldsymbol{S}_0, \boldsymbol{E}_0, \boldsymbol{S}_1, \boldsymbol{E}_1, (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1)) \leftarrow \mathsf{Gen}_{\mathsf{td}}'(1^\kappa, 0, \tau^*)$ and $\boldsymbol{C} \leftarrow \mathcal{D}_\lambda^{\ell\times n}$. It sets $ek = (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{C})$ and $dk = (0, \boldsymbol{S}_0, ek)$. It runs the adversary on input ek and simulates the decryption oracle by using dk.
3. The adversary outputs μ. The challenger generates the challenge ciphertext as follows: It generates
   - $\boldsymbol{c}^* \leftarrow \mathbb{F}_2^n$
   - $\boldsymbol{c}_0^* \leftarrow \mathbb{F}_2^q$
   - $\underline{\boldsymbol{c}_1^* \leftarrow \mathbb{F}_2^q}$
   - $\boldsymbol{c}_2^* \leftarrow \mathbb{F}_2^\ell$.

   It returns $ct^* = (\boldsymbol{c}^*, \boldsymbol{c}_0^*, \boldsymbol{c}_1^*, \boldsymbol{c}_2^*)$. It runs the adversary on input ct^* and simulates the decryption oracle by using dk.
4. Finally, the adversary outputs its guess b' and the challenger outputs b'.

Lemma C.12. *There exists an adversary $\mathcal{A}_{78}$ satisfying*

$$|\Pr[S_7] - \Pr[S_8]| \le \mathsf{Adv}_{\mathsf{1leakyLPN}, \mathcal{A}_{78}}(\kappa).$$

The proof is similar to Lemma C.9.

Proof. We construct $\mathcal{A}_{78}$ as follows:

1. $\mathcal{A}_{78}$ is given $(z, \boldsymbol{c}, \boldsymbol{A}, \boldsymbol{S}_1\boldsymbol{A} + \boldsymbol{E}_1)$, where $z = \boldsymbol{S}_1\boldsymbol{c}$ or $\boldsymbol{r} \leftarrow \mathbb{F}_2^q$. It then invokes $\mathcal{A}$ on input 1^κ and receives τ^*. It generates
   - $\boldsymbol{S}_0 \leftarrow \widetilde{\mathsf{Ber}}_{\mu_1}^{q\times n}$, $\boldsymbol{E}_0 \leftarrow \mathsf{Ber}_\mu^{q\times n}$, $\boldsymbol{B}_0 := \boldsymbol{S}_0\boldsymbol{A} + \boldsymbol{E}_0$
   - $\boldsymbol{B}_1 := \boldsymbol{S}_1\boldsymbol{A} + \boldsymbol{E}_1 - \boldsymbol{G}\boldsymbol{H}_{\tau^*}$
   - $\boldsymbol{C} \leftarrow \mathcal{D}_\lambda^{\ell\times n}$.

   It sets $dk = (0, \boldsymbol{S}_0, ek)$ and $ek = (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{C})$. It runs the adversary on input ek and simulates the decryption oracle by using dk.
2. The adversary outputs μ. $\mathcal{A}_{45}$ generates the challenge ciphertext as follows: It generates
   - $\boldsymbol{c}^* := \boldsymbol{c}$, $\boldsymbol{c}_0^* :\leftarrow \mathbb{F}_2^q$, $\boldsymbol{c}_1^* := z$, and $\boldsymbol{c}_2^* \leftarrow \mathbb{F}_2^\ell$.

   It returns $ct^* = (\boldsymbol{c}^*, \boldsymbol{c}_0^*, \boldsymbol{c}_1^*, \boldsymbol{c}_2^*)$. It runs the adversary on input ct^* and simulates the decryption oracle by using $dk = (0, \boldsymbol{S}_0, ek)$.
3. Finally, the adversary outputs its guess b' and $\mathcal{A}_{78}$ outputs b'.

If $z = \boldsymbol{S}_1\boldsymbol{c}$, then $\mathcal{A}_{78}$ perfectly simulates Game$_7$. If $z = \boldsymbol{r} \leftarrow \mathbb{F}_2^q$, then $\mathcal{A}_{78}$ perfectly simulates Game$_8$. Thus, the lemma follows. □

Game$_9$*:* We again modify how to generate key:

1. The challenger runs the adversary on input 1^κ.
2. The adversary outputs τ^*. The challenger generates keys by $(\boldsymbol{S}_0, \boldsymbol{E}_0, \boldsymbol{S}_1, \boldsymbol{E}_1, (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1)) \leftarrow \underline{\mathsf{Gen}_{\mathsf{td}}'(1^\kappa, 0, 0)}$ and $\boldsymbol{C} \leftarrow \mathcal{D}_\lambda^{\ell\times n}$. It sets $ek = (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{C})$ and $dk = (0, \boldsymbol{S}_0, ek)$. It runs the adversary on input ek and simulates the decryption oracle by using dk.
3. The adversary outputs μ. The challenger generates the challenge ciphertext as follows: It generates
   - $\boldsymbol{c}^* \leftarrow \mathbb{F}_2^n$
   - $\boldsymbol{c}_0^* \leftarrow \mathbb{F}_2^q$
   - $\boldsymbol{c}_1^* \leftarrow \mathbb{F}_2^q$
   - $\boldsymbol{c}_2^* \leftarrow \mathbb{F}_2^\ell$.

It returns $ct^* = (\boldsymbol{c}^*, \boldsymbol{c}_0^*, \boldsymbol{c}_1^*, \boldsymbol{c}_2^*)$. It runs the adversary on input ct^* and simulates the decryption oracle by using dk.

4. Finally, the adversary outputs its guess b' and the challenger outputs b'.

Apparently, we have

$$\Pr[S_9] = \Pr[\mathsf{Expt}^{\text{pr-st-wcca},1}_{\mathsf{TBE}_{\mathsf{YZ}},\mathcal{A}}(\kappa) = 1].$$

Lemma C.13. *There exists an adversary $\mathcal{A}_{89}$ satisfying*

$$|\Pr[S_8] - \Pr[S_9]| \leq \left|\Pr[\mathsf{Expt}^{\text{real}}_{\mathsf{Gen}_{\mathsf{td}}',\mathcal{A}_{89}}(\kappa) = 1] - \Pr[\mathsf{Expt}^{\text{corr}}_{\mathsf{Gen}_{\mathsf{td}}',\mathcal{A}_{89}}(\kappa) = 1]\right|.$$

Proof. We construct $\mathcal{A}_{89}$ that distinguishes real and corr games as follows:

1. Given 1^κ, $\mathcal{A}_{89}$ runs $\mathcal{A}$ on input 1^κ and receives τ^*.
2. It sends $(0, 0, \tau^*, 0)$ to its challenger and receives $(\boldsymbol{S}_0, \boldsymbol{E}_0, \boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{e}, \boldsymbol{Se}, \boldsymbol{s}, \boldsymbol{Es})$. It chooses $\boldsymbol{C} \leftarrow \mathcal{D}_\lambda^{\ell\times n}$. It sets $dk = (0, \boldsymbol{S}_0)$ and $ek = (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{C})$. It runs $\mathcal{A}$ on input ek.
3. $\mathcal{A}_{89}$ simulates the decryption oracle using dk.
4. $\mathcal{A}_{89}$ generates the challenge on a query μ from $\mathcal{A}$ as follows: It generates $\boldsymbol{c}^* \leftarrow \mathbb{F}_2^n$, $\boldsymbol{c}_0^* \leftarrow \mathbb{F}_2^q$, and $\boldsymbol{c}_1^* \leftarrow \mathbb{F}_2^q$. It chooses $\boldsymbol{c}_2^* \leftarrow \mathbb{F}_2^\ell$ and returns $ct^* := (\boldsymbol{c}^*, \boldsymbol{c}_0^*, \boldsymbol{c}_1^*, \boldsymbol{c}_2^*)$.
5. $\mathcal{A}'_{\mathsf{Gen}_{\mathsf{td}}}$ outputs b' if $\mathcal{A}$ finally outputs b'.

Notice that if the game is real and corr, then the keys are generated by $\mathsf{Gen}_{\mathsf{td}}(1^\kappa, 0, \tau^*)$ and $\mathsf{Gen}_{\mathsf{td}}(1^\kappa, 0, 0)$, respectively. Thus, if the game is real and the keys are generated by $\mathsf{Gen}_{\mathsf{td}}(1^\kappa, 0, \tau^*)$, then $\mathcal{A}_{89}$ perfectly simulates Game_8. If the game is corr and keys are generated by $\mathsf{Gen}_{\mathsf{td}}(1^\kappa, 0, 0)$, then $\mathcal{A}_{89}$ perfectly simulates Game_9. This completes the proof. ◻