Chosen Ciphertext and Trapdoor Secure Public Key Authenticated Encryption with Keyword Search

Tomoki Mori[1] and Keita Emura*[1,2]

[1.]Kanazawa University, Japan.
[2.]AIST, Japan.

November 23, 2026

Abstract

For public-key encryption with keyword search (PEKS), chosen-ciphertext attack (CCA) security has been studied in settings where an adversary is allowed to access the test oracle. Notably, the CHK transformation (Canetti, Halevi, and Katz, EUROCRYPT 2004) does not yield CCA-secure PEKS. To address this limitation, Abdalla, Bellare, and Neven (TCC 2010/JoC 2018) proposed a generic construction of CCA-secure PEKS from CCA-secure anonymous and weakly robust identity-based encryption. However, CCA security alone is insufficient to resist keyword-guessing attacks (KGAs) arising from trapdoor leakage in PEKS. To mitigate KGAs, public-key authenticated encryption with keyword search (PAEKS) has been studied. In particular, Li and Boyen (IACR CiC 2024) proposed a generic construction of PAEKS. In this paper, we propose a generic transformation that converts a PAEKS scheme into a CCTA-secure one, where CCTA denotes security against chosen-ciphertext and trapdoor attacks. Our transformation can be viewed as a variant of the CHK transformation. The key observation is that, unlike in PEKS, the receiver in PAEKS identifies and specifies the sender when generating a trapdoor, enabling keywords to be used as tags. The transformation directly upgrades PAEKS instantiations obtained via the Li-Boyen generic construction under various assumptions to CCTA-secure schemes. As a result, we obtain the first fully CCTA-secure PAEKS schemes from both pairings and lattices. As an additional benefit, the transformation improves search efficiency. By leveraging verification keys embedded in ciphertexts and trapdoors, sender-receiver matching can be performed without cryptographic computation, for example, by using hash tables. Consequently, for a trapdoor associated with a particular sender, the search complexity measured in cryptographic operations becomes independent of the number of senders and is essentially the same as that of PEKS. Although improving efficiency is not the primary goal of the transformation, this benefit is of independent interest.

1 Introduction

1.1 PAEKS: Public-Key Authenticated Encryption with Keyword Search

It is well known that public-key encryption with keyword search (PEKS) [8] is vulnerable to keyword guessing attacks (KGAs), in which information about keywords is revealed through trapdoors. In other words, information about the keyword being searched leaks from search queries. There

*Corresponding Author: k-emura@se.kanazawa-u.ac.jp

is a concern that leakage of information about the searched keywords may allow the content of a document to be inferred from the keywords it contains, even if the document itself is encrypted. Moreover, since such keywords may be related to personal interests or preferences, it is strongly required that no keyword information be revealed from search queries. Preventing KGA is impossible in the PEKS setting because anyone can generate a ciphertext for an arbitrary keyword kw using the receiver's public key pk_R and can test whether a trapdoor corresponds to kw via the test algorithm. For example, if the Medical Dictionary[1], which provides authoritative descriptions of medical conditions, medications, anatomical terms, notable medical personalities, and many other medical concepts, and contains more than 45,000 entries from all areas of medicine and healthcare, is used as a keyword source [22], an adversary can identify the keyword associated with a trapdoor by exhaustively testing all candidate keywords. Although additional keywords may be used in practice, the keyword space is unlikely to be exponentially large. Therefore, KGA constitutes a realistic threat in PEKS.

To address this issue, Huang and Li [26] introduced public-key authenticated encryption with keyword search (PAEKS), where encrypting a keyword kw requires not only a receiver's public key pk_R but also a sender's secret key sk_S. As a result, a ciphertext is associated with the receiver, and the keyword, namely $(\mathsf{S}, \mathsf{R}, kw)$. Similarly, to generate a trapdoor TD for a keyword kw', a sender's public key $\mathsf{pk}_{\mathsf{S}'}$ is required in addition to a receiver's secret key $\mathsf{sk}_{\mathsf{R}'}$. Thus, a trapdoor TD is also associated with a tuple of the sender, the receiver, and the keyword, namely $(\mathsf{S}', \mathsf{R}', kw')$.[2] The test algorithm outputs 1 if $(\mathsf{S}, \mathsf{R}, kw) = (\mathsf{S}', \mathsf{R}', kw')$ (correctness), and outputs 0 if $(\mathsf{S}, \mathsf{R}, kw) \neq (\mathsf{S}', \mathsf{R}', kw')$ (consistency). It is particularly worth noting that if $(\mathsf{S}, \mathsf{R}) \neq (\mathsf{S}', \mathsf{R}')$, that is, if there is a sender-receiver mismatch, the test algorithm must output 0 even when $kw = kw'$. This notion of consistency in the multi-user setting was introduced in [19].

In PEKS, the sender encrypts a keyword using the receiver's public key, thereby specifying the receiver, whereas the receiver does not specify the sender, since the trapdoor is sender independent. In contrast, in PAEKS, the receiver computes a trapdoor while explicitly specifying the sender as well. The fact that the receiver recognizes and specifies the sender constitutes the core idea of the construction proposed in this paper and is therefore worth highlighting.

Security Models. In addition to ciphertext indistinguishability (CI), which guarantees that no keyword information is leaked from ciphertexts, trapdoor indistinguishability (TI) is defined to ensure that no keyword information is leaked from trapdoors. As a stronger and more desirable security notion, *full* CI is defined, in which an adversary is allowed to send the challenge keywords to the encryption oracle. Full CI is necessary to capture scenarios in which the same keyword is encrypted multiple times, and it is mandatory in practical settings. Note that, in PEKS, an adversary can obtain ciphertexts for arbitrary keywords, including the challenge keywords. Therefore, full security should be regarded as the standard and minimal security requirement under the CI setting, even for PAEKS. Similarly, full TI is defined, where an adversary is allowed to send the challenge keywords to the trapdoor oracle. From the perspective of the primary goal of PAEKS, namely hiding keyword information from search queries, full TI is preferable from the standpoint of stronger security guarantees. However, unlike searchable ciphertexts, trapdoors are not persistently stored but are typically ephemeral objects generated for query purposes. Therefore, we interpret that even TI security without the full restriction retains a certain degree of significance in this paper.

[1]The American Heritage® Stedman's Medical Dictionary, Second Edition: `https://medical-dictionary.thefreedictionary.com/`

[2]Throughout this paper, we use a prime ($'$) to denote keys and keywords that are used as inputs to the trapdoor generation algorithm. When the context makes this clear, we omit the prime for simplicity.

Generic Construction. Li and Boyen [31] proposed a generic construction of PAEKS based on non-interactive key exchange (NIKE) and symmetric-key equality-predicate encryption (EPE), and provided several full CI- and full TI-secure PAEKS instantiations using pairings and lattices. Moreover, when only TI security is required, their generic construction enables PAEKS instantiations under a wide range of complexity assumptions, including pairing-free Diffie-Hellman assumptions, lattices, codes, and isogenies. Shinoki et al. [37] later identified a flaw in the pairing-based EPE instantiation proposed by Li and Boyen, and introduced an alternative pairing-based EPE construction.

1.2 Research Background: CCA Security in PEKS/PAEKS

In addition to the basic security notion that ensures no keyword information is leaked from ciphertexts, security under chosen-ciphertext attacks (CCA) for PEKS has also been studied [3,4]. In the CCA setting, an adversary is allowed to access a test oracle, which returns the test result for a given pair of a keyword and a ciphertext. Abdalla et al. [3, 4] showed that the generic construction of PEKS from anonymous identity-based encryption (IBE) proposed by Boneh et al. [8] achieves CCA security (resp. consistency) if the underlying IBE scheme is CCA-secure (resp. (weakly) robust). Note, however, that the generic construction of PEKS from anonymous IBE proposed by Abdalla et al. [2], which achieves consistency, does not provide CCA security even when the underlying IBE scheme is CCA-secure. Ma et al. [33] proposed another CCA-secure PEKS construction based on smooth projective hash functions. One might consider applying the CHK (Canetti-Halevi-Katz) transformation [11], a well-known technique for constructing CCA-secure public-key encryption (PKE) schemes, to obtain a CCA-secure PEKS scheme. For example, one could encrypt $\mathsf{vk}||kw$ as a keyword, where vk is a verification key of a one-time signature scheme, and then sign the resulting ciphertext. In this approach, the keyword is treated as a tag [27] required for applying the CHK transformation. However, this approach does not preserve the search functionality of PEKS. This is because a trapdoor is independent of sender information and is associated solely with the keyword.

To the best of our knowledge, CCA-secure PAEKS has not been extensively studied, with the following two exceptions. Yao et al. [42] defined CCA security for CI (Definition 7 in [42]) and proposed a lattice-based PAEKS construction. However, their CCA security model is weak in the sense that an adversary is not allowed to send the challenge keywords to the test oracle. Since the adversary is also prohibited from sending the challenge keywords to the trapdoor oracle (in order to exclude trivial attacks), the adversary cannot obtain any test results related to the challenge keywords. Moreover, they do not define TI security (they define unforgeability of trapdoors), and it is not guaranteed that no keyword information is leaked from trapdoors (Definition 8 in [42]). Actually, a trapdoor in their PAEKS scheme is a variant of the GPV (Gentry-Peikert-Vaikuntanathan) signature [25] on a keyword. While this construction provides trapdoor unforgeability, the trapdoor inherently leaks information about the underlying keyword. Zhang et al. [48] also defined CCA security for CI (Definition 7 in [48]) and for TI (Definition 6 in [48]), and proposed a pairing-based PAEKS construction which they call lightweight and fast dynamic PEKS (LFD-PEKS). We clarify that their encryption algorithm takes the data owner's secret key as input and considers KGA. Therefore, although the authors denote their scheme as PEKS, we regard it as a variant of PAEKS. As a drawback, their CCA security model restricts the input to the test oracle to pairs consisting of the challenge ciphertext and a trapdoor $(\mathsf{CT}^*, \mathsf{TD})$. Since this input coincides with that of the test algorithm itself, an adversary can locally execute the test algorithm without accessing the test oracle; that is, oracle access does not provide any additional information. Beyond the weak form of their CCA security definition, we further demonstrate in the Appendix A that the LFD-PEKS scheme [48] is insecure. Specifically, keyword information is leaked from both ciphertexts

and trapdoors.

1.3 Our Motivation

Here, we revisit the original notion of CCA security in PKE. In this security model, an adversary does not have access to the secret (decryption) key itself. Nevertheless, under CCA security, the adversary is allowed to observe the outputs of the decryption algorithm that uses the secret key. As mentioned in the previous subsection, existing CCA security definitions for PAEKS [42, 48] are insufficient, as they fail to capture the threat model intended by the original notion of CCA security for PKE. Therefore, we refine these definitions accordingly. From this perspective, we informally describe the desirable security models for CI and TI as follows. The following informal definitions may be skipped by the reader, as the formal definitions are given in Section 3.

CCA Full Ciphertext Indistinguishability: An adversary is allowed to send a pair consisting of a keyword and a ciphertext (kw, CT) to the test oracle and obtain the corresponding test result.

- To rule out trivial attacks, a restriction is imposed on queries involving the pair formed by one of the challenge keywords (kw_0^*, kw_1^*) and the challenge ciphertext CT^*, i.e., $(kw, \mathsf{CT}) \notin \{(kw_0^*, \mathsf{CT}^*), (kw_1^*, \mathsf{CT}^*)\}$. In particular, the adversary is still allowed to obtain test results related to the challenge keywords.
- Because the adversary is prohibited from obtaining the trapdoor for a challenge keyword (in order to exclude trivial attacks), access to test results related to the challenge keyword through the test oracle strictly increases the information available to the adversary. This accurately captures the scenario originally intended by CCA-secure PKE.

CTA Full Trapdoor Indistinguishability: An adversary is allowed to send a pair consisting of a keyword and a trapdoor $(kw, \mathsf{TD}) \notin \{(kw_0^*, \mathsf{TD}^*), (kw_1^*, \mathsf{TD}^*)\}$ to the test oracle and obtain the corresponding test result, where TD^* is the challenge trapdoor. Since the adversary is allowed to choose the trapdoor, this notion is more appropriately referred to as *CTA*, where CTA stands for chosen-trapdoor attack.

- Since TI is defined to guarantee that no information about the keyword is leaked from trapdoors, the input to the test oracle should be a trapdoor rather than a ciphertext.
- Because the adversary is prohibited from obtaining the ciphertext corresponding to a challenge keyword (in order to exclude trivial attacks), access to test results related to the challenge keyword through the test oracle strictly increases the information available to the adversary. This again accurately reflects the scenario originally intended by CCA-secure PKE.

We say that a PAEKS scheme is *fully CCTA-secure* if it provides both CCA full CI security and CTA full TI security. We also consider a weaker variant in which the scheme achieves CCA full CI security and CTA TI security. Then, we say that a PAEKS scheme is *CCTA-secure*. Note that CTA security can be defined irrespective of whether it is full or not. As a natural question, we ask:

How can one construct a (fully) CCTA-secure PAEKS scheme?

Since Li and Boyen [31] proposed a generic construction of PAEKS that can be instantiated under a wide range of complexity assumptions, our motivation is as follows:

Table 1: (Fully) CCTA-secure PAEKS Instantiations. (Q)ROM stands for (Quantum) Random Oracle Model, STD stands for Standard Model, and BSW stands for Boneh-Shen-Waters.

Building Blocks (NIKE, EPE, Sig)	Full TI	Note
Diffie-Hellman [16], HMAC [31], Schnorr [15]		ROM, Pairing-free
Pairing-based [23], Pairing-based [37], BSW [9]	✓	STD
Swoosh [24], HMAC [31], Hawk [17]		(Q)ROM, Post-Quantum
Swoosh [24], Lattice-based [31], Hawk [17]	✓	(Q)ROM, Post-Quantum
CSIDE [13], HMAC [31], FAEST [6]		(Q)ROM, Post-Quantum
MODRED [36], HMAC [31], SDitH [35]		(Q)ROM, Post-Quantum

To propose a generic transformation of (fully) CCTA-secure PAEKS that takes PAEKS as a building block.

This enables us to directly transform PAEKS instantiations obtained from the Li-Boyen generic construction under various complexity assumptions into CCTA-secure ones.

1.4 Our Contribution

In this paper, we propose a generic transformation that converts a PAEKS scheme into a CCTA-secure one.

- The proposed transformation employs a strongly unforgeable signature scheme in addition to the underlying PAEKS scheme. As described below (Technical Overview), our transformation technique is an application of the CHK transformation.
  - Our main contribution lies instead in the observation that, while directly applying the CHK transformation to PEKS affects the search functionality (since the keyword representations in ciphertexts and trapdoors are modified), this is not the case for PAEKS. In other words, we observe that the fact that the receiver is aware of the sender enables the CHK transformation to be applied without impairing search functionality in the PAEKS setting.
  - We emphasize that our transformation preserves the full security of the underlying PAEKS scheme. Consequently, fully CCTA-secure PAEKS schemes can be instantiated via the Li-Boyen generic construction (see Table 1).
    - Our transformation yields a lattice-based fully CCTA-secure PAEKS scheme by combining a lattice-based PAEKS scheme (obtained from Swoosh [24] and the Li-Boyen lattice-based EPE via the Li-Boyen generic construction) with, e.g., Hawk [17].
    - PAEKS can be constructed from CSIDH (Commutative Supersingular Isogeny Diffie-Hellman) [13], as noted by Li and Boyen [31]. To the best of our knowledge, there is currently no known transformation that upgrades SQIsign from existential unforgeability to strong unforgeability while relying solely on standard isogeny assumptions [1]. From the perspective of post-quantum cryptographic instantiations, we can employ FAEST [6], whose security relies on symmetric-key cryptography and for which strong unforgeability was proved by Kulkarni and Xagawa [28]. Thus, our transformation yields CCTA-secure PAEKS schemes based on isogenies.

    - ∗ A code-based NIKE [36] has also been proposed and can be used to construct a code-based PAEKS scheme via the generic construction of Li and Boyen. Moreover, Kulkarni and Xagawa [28] proved that SDitH (Syndrome Decoding in the Head) [35] achieves strong unforgeability. Thus, our transformation yields CCTA-secure PAEKS schemes based on codes.
    - ∗ We believe that our pairing-free CCTA-secure instantiation, which relies only on standard elliptic curves such as Curve25519 [7], provides notable practical efficiency, although it is not post-quantum secure.

- Our transformation also provides an efficiency benefit where no cryptographic computation is required for sender-receiver matching. Although not the primary goal of the proposed transformation, this efficiency improvement is of independent interest. See Section 6 for detail.
    - Consequently, our transformation partially addresses the efficiency issue considered in [10, 14, 29, 30, 32, 39–41, 47].

Remark. Regarding HAWK, following the AI-assisted cryptanalysis reported by Anthropic using its Claude model in a blog post[3] (published on July 28, 2026), the HAWK team withdrew HAWK from NIST's Additional Digital Signatures standardization process. The reported attack significantly reduced the estimated security level of the scheme, and the authors noted that restoring the intended security margin would require substantially larger parameters, thereby diminishing the compactness and efficiency advantages that distinguished HAWK from other NIST candidates. In our work, we employ HAWK only as an instantiation of the proposed generic construction because it provides strong unforgeability; it was not selected for efficiency considerations. Nevertheless, we believe it is worthwhile to explicitly discuss alternative signatures that can be used in lattice-based instantiations. One natural candidate for instantiating the proposed generic construction is the construction of Eaton and Song [18], who proposed a generic transformation in the QROM that compiles any existentially unforgeable signature scheme into a strongly unforgeable one. Their construction can be viewed as the QROM counterpart of the transformation by Teranishi et al. [38], and they explicitly presented a lattice-based instantiation based on the Bonsai-tree signature scheme [12]. Therefore, this instantiation can be used as an alternative to HAWK in our framework. Furthermore, as mentioned above, FAEST [6], whose security relies on symmetric-key cryptography and whose strong unforgeability was established by Kulkarni and Xagawa [28], can also serve as a suitable instantiation.

Regarding instantiations via the Li-Boyen generic construction of PAEKS, Matsui and Emura [34] observed that key collision resistance of EPE is required to prove the consistency of PAEKS. They also stated that their analysis requires no modification to the underlying EPE construction and therefore does not affect existing instantiation results. Consequently, their observation does not affect any of the instantiations listed in Table 1.

1.5 Technical Overview

Our transformation is a variant of the CHK transformation but we do not employ a one-time signature. We pointed out that the receiver recognizes and specifies the sender to compute a trapdoor. This fact allows us to employ a keyword as a tag without affecting the search functionality

[3] `https://www.anthropic.com/news/discovering-cryptographic-weaknesses`

unlike PEKS. Let a sender have two public keys $(\mathsf{pk}_\mathsf{S}, \mathsf{vk}_\mathsf{S})$ where pk_S is the sender's public key of the underlying PAEKS scheme and vk_S is a verification key of the underlying signature scheme. Similarly, let a receiver have two public keys $(\mathsf{pk}_\mathsf{R}, \mathsf{vk}_\mathsf{R})$. The sender encrypts a keyword kw by

$$\mathsf{CT}' \leftarrow \mathsf{PAEKS.Enc}(\mathsf{pk}_\mathsf{S}, \mathsf{sk}_\mathsf{S}, \mathsf{pk}_\mathsf{R}, \mathsf{vk}_\mathsf{S}||\mathsf{vk}_\mathsf{R}||kw)$$

Here, $\mathsf{vk}_\mathsf{S}||\mathsf{vk}_\mathsf{R}||kw$ is regarded as a keyword. Similarly, a receiver computes a trapdoor for a keyword kw' by

$$\mathsf{TD}' \leftarrow \mathsf{PAEKS.Enc}(\mathsf{pk}_{\mathsf{R}'}, \mathsf{sk}_{\mathsf{R}'}, \mathsf{pk}_{\mathsf{S}'}, \mathsf{vk}_{\mathsf{S}'}||\mathsf{vk}_{\mathsf{R}'}||kw')$$

Here, if $\mathsf{vk}_\mathsf{S}||\mathsf{vk}_\mathsf{R}||kw = \mathsf{vk}_{\mathsf{S}'}||\mathsf{vk}_{\mathsf{R}'}||kw'$ holds, then $(\mathsf{S}, \mathsf{R}, kw) = (\mathsf{S}', \mathsf{R}', kw')$ holds, and if $\mathsf{vk}_\mathsf{S}||\mathsf{vk}_\mathsf{R}||kw \neq \mathsf{vk}_{\mathsf{S}'}||\mathsf{vk}_{\mathsf{R}'}||kw'$ holds, then $(\mathsf{S}, \mathsf{R}, kw) \neq (\mathsf{S}', \mathsf{R}', kw')$ holds. Thus, the modification does not affect the search functionality (i.e., correctness and consistency) of the underlying PAEKS scheme. The form of a ciphertext and a trapdoor are

$$\mathsf{CT} = (\mathsf{CT}', \mathsf{vk}^c_\mathsf{S}, \mathsf{vk}^c_\mathsf{R}, \sigma_\mathsf{S}) \text{ and } \mathsf{TD} = (\mathsf{TD}', \mathsf{vk}^t_\mathsf{S}, \mathsf{vk}^t_\mathsf{R}, \sigma_\mathsf{R})$$

where σ_S is a signature on CT' using the sender's signing key $\mathsf{sigk}^c_\mathsf{S}$ and vk^c_R is for declaring the receiver's verification key to be used for encryption, and σ_R is a signature on TD' using the receiver's signing key $\mathsf{sigk}^t_\mathsf{R}$ and vk^t_S is for declaring the sender's verification key to be used for generating the trapdoor. Here, the superscripts c and t explicitly indicate that the corresponding values are specified at the time of encryption and trapdoor generation, respectively. The Test algorithm, that takes CT and TD as input, checks whether $(\mathsf{vk}^c_\mathsf{S}, \mathsf{vk}^c_\mathsf{R}) = (\mathsf{vk}^t_\mathsf{S}, \mathsf{vk}^t_\mathsf{R})$, and the validity of σ_S and σ_R, and then runs the Test algorithm of the underlying PAEKS scheme for $(\mathsf{CT}', \mathsf{TD}')$.

Matching of Verification Keys. While it is clear that signature verification is inherited from the CHK transformation, we explain why it is necessary to additionally check the matching of verification keys $(\mathsf{vk}^c_\mathsf{S}, \mathsf{vk}^c_\mathsf{R}) = (\mathsf{vk}^t_\mathsf{S}, \mathsf{vk}^t_\mathsf{R})$. At first glance, since these verification keys can be easily replaced, both ciphertexts and trapdoors appear to be malleable, which may lead one to suspect that CCTA security cannot be achieved under such circumstances. Let us consider the case in which an adversary chooses a verification key different from the specified one as the keyword. For example, for $\mathsf{CT} \leftarrow \mathsf{PAEKS.Enc}(\mathsf{pk}_\mathsf{S}, \mathsf{sk}_\mathsf{S}, \mathsf{pk}_\mathsf{R}, \mathsf{vk}_\mathsf{S}||\mathsf{vk}_\mathsf{R}||kw)$, a CCA full CI adversary generates $(\mathsf{vk}'_\mathsf{S}, \mathsf{sigk}'_\mathsf{S})$, computes σ'_S on CT, and sends $(kw, (\mathsf{CT}, \mathsf{vk}'_\mathsf{S}, \mathsf{vk}_\mathsf{R}, \sigma'_\mathsf{S}))$ to the test oracle. Of course, σ'_S is a valid signature on CT under vk'_S. Then, the adversary can modify the challenge ciphertext by re-computing a signature. Here, we clarify that the trapdoor used for running the Test algorithm on the test oracle is *honestly* generated by kw, pk^*_R, sk^*_R, and pk^*_S (this is analogous to the decryption oracle in CCA security for PKE, which returns the output of the decryption algorithm under the honestly generated secret key). Thus, if $(\mathsf{vk}'_\mathsf{S}, \mathsf{vk}_\mathsf{R}) \neq (\mathsf{vk}^*_\mathsf{S}, \mathsf{vk}^*_\mathsf{R})$, then the oracle outputs 0 that does not give any useful information to the adversary. If $(\mathsf{vk}'_\mathsf{S}, \mathsf{vk}_\mathsf{R}) = (\mathsf{vk}^*_\mathsf{S}, \mathsf{vk}^*_\mathsf{R})$, then the adversary needs to forge a signature. An analogous argument applies when proving CTA full TI security.

Handling Challenge Keyword-Related Test Queries. Next, we explicitly consider the case where the adversary sends $(kw^*_0, \cdot)$ or $(kw^*_1, \cdot)$ to the test oracle. In the security proof, the simulator is not allowed to obtain a trapdoor (resp. a ciphertext) corresponding to kw^*_0 or kw^*_1 in the CCA full CI (resp. CTA full TI) experiment from the PAEKS challenger, in order to rule out trivial attacks. To answer such test queries, again, we rely on the unforgeability of the underlying signature scheme. Specifically, if the adversary sends to the test oracle a ciphertext (resp. a trapdoor) that was not generated by the encryption (resp. trapdoor) oracle in the CCA full CI (resp. CTA full TI) experiment, then the simulator can break the unforgeability of the underlying signature scheme. After performing the sequence of game hops, the simulator is able to obtain knowledge of

all ciphertexts (resp. all trapdoors) appearing in the security proof. Consequently, upon receiving a test query (kw, CT) (resp. (kw, TD)), the simulator already knows whether CT (resp. TD) corresponds to kw, without invoking the test algorithm. Strong unforgeability is crucial to exclude the case where the adversary sends $(\mathsf{CT}^*, \mathsf{vk}_\mathsf{S}^*, \mathsf{vk}_\mathsf{R}^*, \sigma_\mathsf{S})$ (resp. $(\mathsf{TD}^*, \mathsf{vk}_\mathsf{S}^*, \mathsf{vk}_\mathsf{R}^*, \sigma_\mathsf{R})$) to the test oracle, where σ_S (resp. σ_R) does not appear in the challenge ciphertext (resp. the challenge trapdoor), as in the CHK transformation. Finally, we construct an algorithm that breaks full CI (resp. full TI) security of the underlying PAEKS scheme. We remark that our security proof remains valid even if the underlying PAEKS scheme satisfies only TI security.

2 Preliminaries

2.1 Signatures

Let $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a signature scheme. The key generation algorithm KeyGen takes a security parameter $\lambda \in \mathbb{N}$ as input and outputs a verification and signing key pair $(\mathsf{vk}, \mathsf{sigk})$. The signing algorithm Sign takes sigk and a message to be signed M as input and outputs a signature σ. The verification algorithm Verify takes vk, M, and σ as input and outputs 0 or 1. Correctness requires that for any $\lambda \in \mathbb{N}$, any $(\mathsf{vk}, \mathsf{sigk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and any M, $\mathsf{Verify}(\mathsf{vk}, M, \mathsf{Sign}(\mathsf{sigk}, M)) = 1$ holds with overwhelming probability in the security parameter λ. Strong existential unforgeability under chosen-message attacks (strong EUF-CMA) is defined as follows.

Definition 1 (Strong EUF-CMA)**.** *Let $\mathcal{A}$ be an adversary and $\mathcal{C}$ be the challenger. $\mathcal{C}$ generates $(\mathsf{vk}, \mathsf{sigk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and gives vk to $\mathcal{A}$. $\mathcal{C}$ initiates $\mathsf{SigSet} = \emptyset$. $\mathcal{A}$ is allowed to issue signing queries M. $\mathcal{C}$ generates $\sigma \leftarrow \mathsf{Sign}(\mathsf{sigk}, M)$ and returns σ to $\mathcal{A}$. $\mathcal{C}$ stores (M, σ) to SigSet. Finally, $\mathcal{A}$ outputs (M^*, σ^*). We say that $\mathcal{A}$ wins if*

$$\mathsf{Verify}(\mathsf{vk}, M^*, \sigma^*) = 1 \wedge (M^*, \sigma^*) \notin \mathsf{SigSet}$$

The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^{\mathsf{sEUF\text{-}CMA}}_{\mathsf{Sig},\mathcal{A}}(\lambda) = \Pr[\mathcal{A}\ wins]$. We say that Sig provides strong EUF-CMA security if $\mathsf{Adv}^{\mathsf{sEUF\text{-}CMA}}_{\mathsf{Sig},\mathcal{A}}(\lambda)$ is negligible for all probabilistic polynomial-time (PPT) adversaries $\mathcal{A}$. If $\mathcal{A}$ is not allowed to send M^* as a signing query, then we say that Sig is EUF-CMA secure, and the advantage $\mathcal{A}$ is defined as $\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{Sig},\mathcal{A}}(\lambda)$.

2.2 PAEKS

Let $\mathsf{PAEKS} = (\mathsf{Setup}, \mathsf{KeyGen}_\mathsf{S}, \mathsf{KeyGen}_\mathsf{R}, \mathsf{Enc}, \mathsf{Trapdoor}, \mathsf{Test})$ be a PAEKS scheme which is defined as follows. Let $\mathcal{KS}$ be a keyword space. The following definitions are the same as those of Li and Boyen [31], except that we explicitly define consistency that Li and Boyen omitted.

Setup: The setup algorithm takes a security parameter $\lambda \in \mathbb{N}$ and outputs a common parameter pp.

$\mathsf{KeyGen}_\mathsf{S}$: The sender key generation algorithm takes pp as input, and outputs a sender public/private key pair$(\mathsf{pk}_\mathsf{S}, \mathsf{sk}_\mathsf{S})$.

$\mathsf{KeyGen}_\mathsf{R}$: The receiver key generation algorithm takes pp as input, and outputs a receiver public/private key pair $(\mathsf{pk}_\mathsf{R}, \mathsf{sk}_\mathsf{R})$.

Enc: The keyword encryption algorithm takes pk_S, sk_S, pk_R, and a keyword $kw \in \mathcal{KS}$, and outputs a ciphertext CT.

Trapdoor: The trapdoor generation algorithm takes $\mathsf{pk_R}$, $\mathsf{sk_R}$, $\mathsf{pk_S}$, and a keyword $kw \in \mathcal{KS}$, and outputs a trapdoor TD.

Test: The test algorithm takes CT, and TD, and outputs 0 or 1.

Correctness: For all $\lambda \in \mathbb{N}$, $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{pk_S}, \mathsf{sk_S}) \leftarrow \mathsf{KeyGen_S}(\mathsf{pp})$, $(\mathsf{pk_R}, \mathsf{sk_R}) \leftarrow \mathsf{KeyGen_R}(\mathsf{pp})$, and $kw \in \mathcal{KS}$, it is required that $\mathsf{Test}(\mathsf{Enc}(\mathsf{pk_S}, \mathsf{sk_S}, \mathsf{pk_R}, kw), \mathsf{Trapdoor}(\mathsf{pk_R}, \mathsf{sk_R}, \mathsf{pk_S}, kw)) = 1$ holds with overwhelming probability.

Consistency: It guarantees that the Test algorithm outputs 0 if $(\mathsf{S}, \mathsf{R}, kw) \neq (\mathsf{S}', \mathsf{R}', kw')$ where a ciphertext is generated by $(\mathsf{pk_S}, \mathsf{sk_S}, \mathsf{pk_R}, kw)$ and a trapdoor is generated by $(\mathsf{pk_{R'}}, \mathsf{sk_{R'}}, \mathsf{pk_{S'}}, kw')$.

Definition 2 (Consistency)**.** *We define the following experiment.*

$\mathsf{Exp}^{\mathsf{Consist}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda)$:
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$(\mathsf{pk_S}, \mathsf{sk_S}) \leftarrow \mathsf{KeyGen_S}(\mathsf{pp})$; $(\mathsf{pk_{S'}}, \mathsf{sk_{S'}}) \leftarrow \mathsf{KeyGen_S}(\mathsf{pp})$
$(\mathsf{pk_R}, \mathsf{sk_R}) \leftarrow \mathsf{KeyGen_R}(\mathsf{pp})$; $(\mathsf{pk_{R'}}, \mathsf{sk_{R'}}) \leftarrow \mathsf{KeyGen_R}(\mathsf{pp})$
$(kw, kw', c_s, c_r, t_s, t_r) \leftarrow \mathcal{A}(\mathsf{pk_S}, \mathsf{pk_{S'}}, \mathsf{pk_R}, \mathsf{pk_{R'}})$
s.t. $kw, kw' \in \mathcal{KS}$, $c_s, t_s \in \{\mathsf{S}, \mathsf{S}'\}$ *and* $c_r, t_r \in \{\mathsf{R}, \mathsf{R}'\}$
$\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{pk}_{c_s}, \mathsf{sk}_{c_s}, \mathsf{pk}_{c_r}, kw)$; $\mathsf{TD} \leftarrow \mathsf{Trapdoor}(\mathsf{pk}_{t_r}, \mathsf{sk}_{t_r}, \mathsf{pk}_{t_s}, kw')$
If $\mathsf{Test}(\mathsf{CT}, \mathsf{TD}) = 1 \wedge (c_s, c_r, kw) \neq (t_s, t_r, kw')$ *then output* 1 *and* 0 *otherwise.*

We say that PAEKS *provides consistency if*

$$\mathsf{Adv}^{\mathsf{Consist}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda) := \Pr[\mathsf{Exp}^{\mathsf{Consist}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda) = 1]$$

is negligible in the security parameter for all probabilistic polynomial-time (PPT) adversaries $\mathcal{A}$*.*

Full CI: It guarantees that no keyword information is revealed from ciphertexts. An adversary is allowed to send any keyword including the challenge ones to the encryption oracle. In order to exclude trivial attacks, the adversary is not allowed to send the challenge keywords (with $\mathsf{pk^*_S}$) to the trapdoor oracle.

Definition 3 (Full CI)**.** *We define the following experiment.*

$\mathsf{Exp}^{\mathsf{Full\text{-}CI}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda)$:
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$(\mathsf{pk^*_S}, \mathsf{sk^*_S}) \leftarrow \mathsf{KeyGen_S}(\mathsf{pp})$; $(\mathsf{pk^*_R}, \mathsf{sk^*_R}) \leftarrow \mathsf{KeyGen_R}(\mathsf{pp})$
$(kw^*_0, kw^*_1, \mathsf{state}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Enc}}(\mathsf{pk^*_S}, \mathsf{sk^*_S}, \cdot, \cdot), \mathcal{O}_{\mathsf{Trapdoor}}(\mathsf{pk^*_R}, \mathsf{sk^*_R}, \cdot, \cdot)}(\mathsf{pk^*_S}, \mathsf{pk^*_R})$
s.t. $kw^*_0, kw^*_1 \in \mathcal{KS}$ *and* $kw^*_0 \neq kw^*_1$
$\mathsf{coin} \xleftarrow{\$} \{0, 1\}$; $\mathsf{CT}^* \leftarrow \mathsf{Enc}(\mathsf{pk^*_S}, \mathsf{sk^*_S}, \mathsf{pk^*_R}, kw^*_{\mathsf{coin}})$
$\mathsf{coin}' \leftarrow \mathcal{A}^{\mathsf{Enc}(\mathsf{pk^*_S}, \mathsf{sk^*_S}, \cdot, \cdot), \mathsf{Trapdoor}(\mathsf{pk^*_R}, \mathsf{sk^*_R}, \cdot)}(\mathsf{CT}^*, \mathsf{state})$
If $\mathsf{coin} = \mathsf{coin}'$ *then output* 1 *and* 0 *otherwise.*

The encryption oracle $\mathcal{O}_{\mathsf{Enc}}$ *takes a receiver's public key* $\mathsf{pk_R}$ *and a keyword* $kw \in \mathcal{KS}$ *and returns a ciphertext* $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{pk^*_S}, \mathsf{sk^*_S}, \mathsf{pk_R}, kw)$*. The trapdoor oracle* $\mathcal{O}_{\mathsf{Trapdoor}}$ *takes a sender's public key*

$\mathsf{pk_S}$ *and a keyword* $kw \in \mathcal{KS}$*, where* $(\mathsf{pk_S}, kw) \notin \{(\mathsf{pk}^*_\mathsf{S}, kw^*_0), (\mathsf{pk}^*_\mathsf{S}, kw^*_1)\}$*, and returns a trapdoor* $\mathsf{TD} \leftarrow \mathsf{Trapdoor}(\mathsf{pk}^*_\mathsf{R}, \mathsf{sk}^*_\mathsf{R}, \mathsf{pk_S}, kw)$*. We say that* PAEKS *is fully CI secure if*

$$\mathsf{Adv}^{\mathsf{Full\text{-}CI}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{Full\text{-}CI}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}|$$

is negligible in the security parameter for all PPT adversaries $\mathcal{A}$*.*

Full TI: It guarantees that no keyword information is revealed from trapdoors. An adversary is allowed to send any keyword including the challenge ones to the trapdoor oracle. In order to exclude trivial attacks, the adversary is not allowed to send the challenge keywords (with pk^*_R) to the encryption oracle.

Definition 4 (Full TI)**.** *We define the following experiment.*

$\mathsf{Exp}^{\mathsf{Full\text{-}TI}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda)$:
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$(\mathsf{pk}^*_\mathsf{S}, \mathsf{sk}^*_\mathsf{S}) \leftarrow \mathsf{KeyGen_S}(\mathsf{pp})$; $(\mathsf{pk}^*_\mathsf{R}, \mathsf{sk}^*_\mathsf{R}) \leftarrow \mathsf{KeyGen_R}(\mathsf{pp})$
$(kw^*_0, kw^*_1, \mathsf{state}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Enc}}(\mathsf{pk}^*_\mathsf{S}, \mathsf{sk}^*_\mathsf{S}, \cdot, \cdot), \mathcal{O}_{\mathsf{Trapdoor}}(\mathsf{pk}^*_\mathsf{R}, \mathsf{sk}^*_\mathsf{R}, \cdot, \cdot)}(\mathsf{pk}^*_\mathsf{S}, \mathsf{pk}^*_\mathsf{R})$
  s.t. $kw^*_0, kw^*_1 \in \mathcal{KS}$ *and* $kw^*_0 \neq kw^*_1$
$\mathsf{coin} \xleftarrow{\$} \{0,1\}$; $\mathsf{TD}^* \leftarrow \mathsf{Trapdoor}(\mathsf{pk}^*_\mathsf{R}, \mathsf{sk}^*_\mathsf{R}, \mathsf{pk}^*_\mathsf{S}, kw^*_{\mathsf{coin}})$
$\mathsf{coin}' \leftarrow \mathcal{A}^{\mathsf{Enc}(\mathsf{pk}^*_\mathsf{S}, \mathsf{sk}^*_\mathsf{S}, \cdot, \cdot), \mathsf{Trapdoor}(\mathsf{pk}^*_\mathsf{R}, \mathsf{sk}^*_\mathsf{R}, \cdot, \cdot)}(\mathsf{TD}^*, \mathsf{state})$
If $\mathsf{coin} = \mathsf{coin}'$ *then output* 1 *and* 0 *otherwise.*

The encryption oracle $\mathcal{O}_{\mathsf{Enc}}$ *takes a receiver's public key* $\mathsf{pk_R}$ *and a keyword* $kw \in \mathcal{KS}$*, where* $(\mathsf{pk_R}, kw) \notin \{(\mathsf{pk}^*_\mathsf{R}, kw^*_0), (pkr^*, kw^*_1)\}$*, and returns a ciphertext* $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{pk}^*_\mathsf{S}, \mathsf{sk}^*_\mathsf{S}, \mathsf{pk_R}, kw)$*. The trapdoor oracle* $\mathcal{O}_{\mathsf{Trapdoor}}$ *takes a sender's public key* $\mathsf{pk_S}$ *and a keyword* $kw \in \mathcal{KS}$*, and returns a trapdoor* $\mathsf{TD} \leftarrow \mathsf{Trapdoor}(\mathsf{pk}^*_\mathsf{R}, \mathsf{sk}^*_\mathsf{R}, \mathsf{pk_S}, kw)$*. We say that* PAEKS *is fully TI secure if*

$$\mathsf{Adv}^{\mathsf{Full\text{-}TI}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{Full\text{-}TI}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}|$$

is negligible in the security parameter for all PPT adversaries $\mathcal{A}$*.*

If $\mathcal{A}$ is not allowed to send the challenge keywords (with pk^*_S) to the trapdoor oracle, then this security notion is referred to as TI (then the trapdoor oracle $\mathcal{O}_{\mathsf{Trapdoor}}$ takes $(\mathsf{pk_S}, kw) \notin \{(\mathsf{pk}^*_\mathsf{S}, kw^*_0), (\mathsf{pk}^*_\mathsf{S}, kw^*_1)\}$), and the advantage $\mathcal{A}$ is defined as $\mathsf{Adv}^{\mathsf{TI}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda)$.

3 Definition of CCTA Security

In this section, we provide formal definitions of the informal security notions described in Section 1.3.

CCA Full CI: In addition to full CI, an adversary is allowed to send a pair consisting of a keyword and a ciphertext (kw, CT) to the test oracle where $(kw, \mathsf{CT}) \notin \{(kw^*_0, \mathsf{CT}^*), (kw^*_1, \mathsf{CT}^*)\}$, and obtains the corresponding test result. We note that $\mathcal{A}$ can send $(\mathsf{pk_S}, kw) \notin \{(\mathsf{pk}^*_\mathsf{S}, kw^*_0), (\mathsf{pk}^*_\mathsf{S}, kw^*_1)\}$ to the trapdoor oracle $\mathcal{O}_{\mathsf{Trapdoor}}$. Thus, $\mathcal{A}$ can run the Test algorithm with a trapdoor $\mathsf{TD} \leftarrow \mathsf{Trapdoor}(\mathsf{pk}^*_\mathsf{R}, \mathsf{sk}^*_\mathsf{R}, \mathsf{pk_S}, kw)$ without accessing the test oracle $\mathcal{O}_{\mathsf{Test}}$. To increase the information that $\mathcal{A}$ obtains, in our definition, $\mathcal{A}$ is allowed to obtain the test result with $\mathsf{CT} \neq \mathsf{CT}^*$ and with a trapdoor $\mathsf{TD} \leftarrow \mathsf{Trapdoor}(\mathsf{pk}^*_\mathsf{R}, \mathsf{sk}^*_\mathsf{R}, \mathsf{pk}^*_\mathsf{S}, kw)$ where $kw \in \{kw^*_0, kw^*_1\}$. Consequently, $\mathcal{A}$ can obtain information that is unavailable to it in the full CI model.

Definition 5 (CCA Full CI)**.** *We define the following experiment.*

$$
\begin{aligned}
&\mathsf{Exp}^{\mathsf{CCA\text{-}Full\text{-}CI}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda):\\
&\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)\\
&(\mathsf{pk}^*_\mathsf{S}, \mathsf{sk}^*_\mathsf{S}) \leftarrow \mathsf{KeyGen}_\mathsf{S}(\mathsf{pp});\ (\mathsf{pk}^*_\mathsf{R}, \mathsf{sk}^*_\mathsf{R}) \leftarrow \mathsf{KeyGen}_\mathsf{R}(\mathsf{pp})\\
&(kw^*_0, kw^*_1, \mathsf{state}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Enc}}(\mathsf{pk}^*_\mathsf{S}, \mathsf{sk}^*_\mathsf{S}, \cdot, \cdot), \mathcal{O}_{\mathsf{Trapdoor}}(\mathsf{pk}^*_\mathsf{R}, \mathsf{sk}^*_\mathsf{R}, \cdot, \cdot), \mathcal{O}_{\mathsf{Test}}(\cdot, \cdot)}(\mathsf{pk}^*_\mathsf{S}, \mathsf{pk}^*_\mathsf{R})\\
&\quad \textit{s.t. } kw^*_0, kw^*_1 \in \mathcal{KS} \textit{ and } kw^*_0 \neq kw^*_1\\
&\mathsf{coin} \xleftarrow{\$} \{0,1\};\ \mathsf{CT}^* \leftarrow \mathsf{Enc}(\mathsf{pk}^*_\mathsf{S}, \mathsf{sk}^*_\mathsf{S}, \mathsf{pk}^*_\mathsf{R}, kw^*_{\mathsf{coin}})\\
&\mathsf{coin}' \leftarrow \mathcal{A}^{\mathsf{Enc}(\mathsf{pk}^*_\mathsf{S}, \mathsf{sk}^*_\mathsf{S}, \cdot, \cdot), \mathsf{Trapdoor}(\mathsf{pk}^*_\mathsf{R}, \mathsf{sk}^*_\mathsf{R}, \cdot\cdot), \mathcal{O}_{\mathsf{Test}}(\cdot, \cdot)}(\mathsf{CT}^*, \mathsf{state})\\
&\textit{If } \mathsf{coin} = \mathsf{coin}' \textit{ then output } 1 \textit{ and } 0 \textit{ otherwise.}
\end{aligned}
$$

The encryption oracle $\mathcal{O}_{\mathsf{Enc}}$ *and the trapdoor oracle* $\mathcal{O}_{\mathsf{Trapdoor}}$ *are the same as those of full CI. The test oracle* $\mathcal{O}_{\mathsf{Test}}$ *takes a keyword* $kw \in \mathcal{KS}$ *and a ciphertext* CT *where* $(kw, \mathsf{CT}) \notin \{(kw^*_0, \mathsf{CT}^*), (kw^*_1, \mathsf{CT}^*)\}$*. The oracle computes* $\mathsf{TD} \leftarrow \mathsf{Trapdoor}(\mathsf{pk}^*_\mathsf{R}, \mathsf{sk}^*_\mathsf{R}, \mathsf{pk}^*_\mathsf{S}, kw)$ *and returns the result of* $\mathsf{Test}(\mathsf{CT}, \mathsf{TD})$*. We say that* PAEKS *is CCA fully CI secure if*

$$\mathsf{Adv}^{\mathsf{CCA\text{-}Full\text{-}CI}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{CCA\text{-}Full\text{-}CI}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}|$$

is negligible in the security parameter for all PPT adversaries $\mathcal{A}$*.*

CTA Full TI: In addition to full TI, an adversary is allowed to send a pair consisting of a keyword and a trapdoor (kw, TD) to the test oracle where $(kw, \mathsf{TD}) \notin \{(kw^*_0, \mathsf{TD}^*), (kw^*_1, \mathsf{TD}^*)\}$, and obtains the corresponding test result. We note that $\mathcal{A}$ can send $(\mathsf{pk}_\mathsf{R}, kw) \notin \{(\mathsf{pk}^*_\mathsf{R}, kw^*_0), (\mathsf{pk}^*_\mathsf{R}, kw^*_1)\}$ to the encryption oracle $\mathcal{O}_{\mathsf{Enc}}$. Thus, $\mathcal{A}$ can run the Test algorithm with a ciphertext $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{pk}^*_\mathsf{S}, \mathsf{sk}^*_\mathsf{S}, \mathsf{pk}_\mathsf{R}, kw)$ without accessing the test oracle $\mathcal{O}_{\mathsf{Test}}$. To increase the information that $\mathcal{A}$ obtains, in our definition, $\mathcal{A}$ is allowed to obtain the test result with $\mathsf{TD} \neq \mathsf{TD}^*$ and with a ciphertext $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{pk}^*_\mathsf{S}, \mathsf{sk}^*_\mathsf{S}, \mathsf{pk}^*_\mathsf{R}, kw)$ where $kw \in \{kw^*_0, kw^*_1\}$. Consequently, $\mathcal{A}$ can obtain information that is unavailable to it in the full TI model.

Definition 6 (CTA Full TI)**.** *We define the following experiment.*

$$
\begin{aligned}
&\mathsf{Exp}^{\mathsf{CTA\text{-}Full\text{-}TI}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda):\\
&\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)\\
&(\mathsf{pk}^*_\mathsf{S}, \mathsf{sk}^*_\mathsf{S}) \leftarrow \mathsf{KeyGen}_\mathsf{S}(\mathsf{pp});\ (\mathsf{pk}^*_\mathsf{R}, \mathsf{sk}^*_\mathsf{R}) \leftarrow \mathsf{KeyGen}_\mathsf{R}(\mathsf{pp})\\
&(kw^*_0, kw^*_1, \mathsf{state}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Enc}}(\mathsf{pk}^*_\mathsf{S}, \mathsf{sk}^*_\mathsf{S}, \cdot, \cdot), \mathcal{O}_{\mathsf{Trapdoor}}(\mathsf{pk}^*_\mathsf{R}, \mathsf{sk}^*_\mathsf{R}, \cdot, \cdot), \mathcal{O}_{\mathsf{Test}}(\cdot, \cdot)}(\mathsf{pk}^*_\mathsf{S}, \mathsf{pk}^*_\mathsf{R})\\
&\quad \textit{s.t. } kw^*_0, kw^*_1 \in \mathcal{KS} \textit{ and } kw^*_0 \neq kw^*_1\\
&\mathsf{coin} \xleftarrow{\$} \{0,1\};\ \mathsf{TD}^* \leftarrow \mathsf{Trapdoor}(\mathsf{pk}^*_\mathsf{R}, \mathsf{sk}^*_\mathsf{R}, \mathsf{pk}^*_\mathsf{S}, kw^*_{\mathsf{coin}})\\
&\mathsf{coin}' \leftarrow \mathcal{A}^{\mathsf{Enc}(\mathsf{pk}^*_\mathsf{S}, \mathsf{sk}^*_\mathsf{S}, \cdot, \cdot), \mathsf{Trapdoor}(\mathsf{pk}^*_\mathsf{R}, \mathsf{sk}^*_\mathsf{R}, \cdot, \cdot), \mathcal{O}_{\mathsf{Test}}(\cdot, \cdot)}(\mathsf{TD}^*, \mathsf{state})\\
&\textit{If } \mathsf{coin} = \mathsf{coin}' \textit{ then output } 1 \textit{ and } 0 \textit{ otherwise.}
\end{aligned}
$$

The encryption oracle $\mathcal{O}_{\mathsf{Enc}}$ *and the trapdoor oracle* $\mathcal{O}_{\mathsf{Trapdoor}}$ *are the same as those of full CI. The test oracle* $\mathcal{O}_{\mathsf{Test}}$ *takes a keyword* $kw \in \mathcal{KS}$ *and a trapdoor* TD *where* $(kw, \mathsf{TD}) \notin \{(kw^*_0, \mathsf{TD}^*), (kw^*_1, \mathsf{TD}^*)\}$*. The oracle computes* $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{pk}^*_\mathsf{S}, \mathsf{sk}^*_\mathsf{S}, \mathsf{pk}^*_\mathsf{R}, kw)$ *and returns the result of* $\mathsf{Test}(\mathsf{CT}, \mathsf{TD})$*. We say that* PAEKS *is CTA fully TI secure if*

$$\mathsf{Adv}^{\mathsf{CTA\text{-}Full\text{-}TI}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{CTA\text{-}Full\text{-}TI}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}|$$

is negligible in the security parameter for all PPT adversaries $\mathcal{A}$*.*

If $\mathcal{A}$ is not allowed to send the challenge keywords (with $\mathsf{pk}_{\mathsf{S}}^*$) to the trapdoor oracle, then this security notion is referred to as CTA TI (then the trapdoor oracle $\mathcal{O}_{\mathsf{Trapdoor}}$ takes $(\mathsf{pk}_{\mathsf{S}}, kw) \notin \{(\mathsf{pk}_{\mathsf{S}}^*, kw_0^*), (\mathsf{pk}_{\mathsf{S}}^*, kw_1^*)\}$, and the advantage $\mathcal{A}$ is defined as $\mathsf{Adv}^{\mathsf{CTA\text{-}TI}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda)$). We note that $\mathcal{A}$ is allowed to obtain the test result with $\mathsf{TD} \neq \mathsf{TD}^*$ and with a ciphertext $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{pk}_{\mathsf{S}}^*, \mathsf{sk}_{\mathsf{S}}^*, \mathsf{pk}_{\mathsf{R}}^*, kw)$ where $kw \in \{kw_0^*, kw_1^*\}$. Consequently, $\mathcal{A}$ can obtain information that is unavailable to it in the TI model, and we claim that the CTA TI model enhances the security level realized by the TI model.

Definition 7. *We say that* PAEKS *is fully CCTA-secure if it provides both CCA full CI security and CTA full TI security.*

Definition 8. *We say that* PAEKS *is CCTA-secure if it provides both CCA full CI security and CTA TI security.*

Remark. Regarding the interfaces of $\mathcal{O}_{\mathsf{Enc}}$ (resp. $\mathcal{O}_{\mathsf{Trapdoor}}$), an adversary submits a public key pk_{R} (resp. pk_{S}) that conforms to the definitions given by Li and Boyen [31]. At first glance, allowing a public key as input may appear to capture the scenario in which a ciphertext (resp. trapdoor) is generated using an adversarially generated public key. However, in their security proof, if the public key has not been generated in advance, the simulator generates a public key (via the NIKE key generation oracle) and then uses it. Therefore, the honest key registration (HKR) setting [23] is sufficient. To make this behavior more explicit, it would be preferable to define $\mathcal{O}_{\mathsf{Enc}}$ to take (R, kw) (resp. $\mathcal{O}_{\mathsf{Trapdoor}}$ to take (S, kw)) as input, where R (resp. S) is treated as an identity for the NIKE key generation. We assume that our oracle definitions implicitly follow the above HKR setting, as in Li and Boyen [31].

4 Proposed Generic Transformation

Let $\mathsf{PAEKS}' = (\mathsf{Setup}', \mathsf{KeyGen}'_{\mathsf{S}}, \mathsf{KeyGen}'_{\mathsf{R}}, \mathsf{Enc}', \mathsf{Trapdoor}', \mathsf{Test}')$ be a PAEKS scheme and $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a signature scheme. We construct a PAEKS scheme $\mathsf{PAEKS} = (\mathsf{Setup}, \mathsf{KeyGen}_{\mathsf{S}}, \mathsf{KeyGen}_{\mathsf{R}}, \mathsf{Enc}, \mathsf{Trapdoor}, \mathsf{Test})$ from PAEKS' and Sig as follows. As mentioned in the technical overview, our transformation is a variant of the CHK transformation but we do not employ a one-time signature, and $\mathsf{vk}_{\mathsf{S}}||\mathsf{vk}_{\mathsf{R}}||kw$ is regarded as a keyword.

$\mathsf{Setup}(1^\lambda)$**:** Run $\mathsf{pp}' \leftarrow \mathsf{Setup}'(1^\lambda)$ and output $\mathsf{pp} = \mathsf{pp}'$.

$\mathsf{KeyGen}_{\mathsf{S}}(\mathsf{pp})$**:** Parse $\mathsf{pp} = \mathsf{pp}'$. Run $(\mathsf{pk}'_{\mathsf{S}}, \mathsf{sk}'_{\mathsf{S}}) \leftarrow \mathsf{KeyGen}'_{\mathsf{S}}(\mathsf{pp}')$ and $(\mathsf{vk}_{\mathsf{S}}, \mathsf{sigk}_{\mathsf{S}}) \leftarrow \mathsf{KeyGen}(1^\lambda)$. Output $\mathsf{pk}_{\mathsf{S}} = (\mathsf{pk}'_{\mathsf{S}}, \mathsf{vk}_{\mathsf{S}})$ and $\mathsf{sk}_{\mathsf{S}} = (\mathsf{sk}'_{\mathsf{S}}, \mathsf{sigk}_{\mathsf{S}})$.

$\mathsf{KeyGen}_{\mathsf{R}}$**:** Parse $\mathsf{pp} = \mathsf{pp}'$. Run $(\mathsf{pk}'_{\mathsf{R}}, \mathsf{sk}'_{\mathsf{R}}) \leftarrow \mathsf{KeyGen}'_{\mathsf{R}}(\mathsf{pp}')$ and $(\mathsf{vk}_{\mathsf{R}}, \mathsf{sigk}_{\mathsf{R}}) \leftarrow \mathsf{KeyGen}(1^\lambda)$. Output $\mathsf{pk}_{\mathsf{R}} = (\mathsf{pk}'_{\mathsf{R}}, \mathsf{vk}_{\mathsf{R}})$ and $\mathsf{sk}_{\mathsf{R}} = (\mathsf{sk}'_{\mathsf{R}}, \mathsf{sigk}_{\mathsf{R}})$.

$\mathsf{Enc}(\mathsf{pk}_{\mathsf{S}}, \mathsf{sk}_{\mathsf{S}}, \mathsf{pk}_{\mathsf{R}}, kw)$**:** Parse $\mathsf{pk}_{\mathsf{S}} = (\mathsf{pk}'_{\mathsf{S}}, \mathsf{vk}_{\mathsf{S}})$, $\mathsf{sk}_{\mathsf{S}} = (\mathsf{sk}'_{\mathsf{S}}, \mathsf{sigk}_{\mathsf{S}})$, and $\mathsf{pk}_{\mathsf{R}} = (\mathsf{pk}'_{\mathsf{R}}, \mathsf{vk}_{\mathsf{R}})$. Run $\mathsf{CT}' \leftarrow \mathsf{Enc}(\mathsf{pk}'_{\mathsf{S}}, \mathsf{sk}'_{\mathsf{S}}, \mathsf{pk}'_{\mathsf{R}}, \mathsf{vk}_{\mathsf{S}}||\mathsf{vk}_{\mathsf{R}}||kw)$ and $\sigma_{\mathsf{S}} \leftarrow \mathsf{Sign}(\mathsf{sigk}_{\mathsf{S}}, \mathsf{CT}')$. Output $\mathsf{CT} = (\mathsf{CT}', \mathsf{vk}_{\mathsf{S}}, \mathsf{vk}_{\mathsf{R}}, \sigma_{\mathsf{S}})$.

$\mathsf{Trapdoor}(\mathsf{pk}_{\mathsf{R}}, \mathsf{sk}_{\mathsf{R}}, \mathsf{pk}_{\mathsf{S}}, kw)$**:** Parse $\mathsf{pk}_{\mathsf{R}} = (\mathsf{pk}'_{\mathsf{R}}, \mathsf{vk}_{\mathsf{R}})$, $\mathsf{sk}_{\mathsf{R}} = (\mathsf{sk}'_{\mathsf{R}}, \mathsf{sigk}_{\mathsf{R}})$, and $\mathsf{pk}_{\mathsf{S}} = (\mathsf{pk}'_{\mathsf{S}}, \mathsf{vk}_{\mathsf{S}})$. Run $\mathsf{TD}' \leftarrow \mathsf{Trapdoor}(\mathsf{pk}'_{\mathsf{R}}, \mathsf{sk}'_{\mathsf{R}}, \mathsf{pk}'_{\mathsf{S}}, \mathsf{vk}_{\mathsf{S}}||\mathsf{vk}_{\mathsf{R}}||kw)$ and $\sigma_{\mathsf{R}} \leftarrow \mathsf{Sign}(\mathsf{sigk}_{\mathsf{R}}, \mathsf{TD}')$. Output $\mathsf{TD} = (\mathsf{TD}', \mathsf{vk}_{\mathsf{S}}, \mathsf{vk}_{\mathsf{R}}, \sigma_{\mathsf{R}})$.

$\mathsf{Test}(\mathsf{CT}, \mathsf{TD})$**:** Parse $\mathsf{CT} = (\mathsf{CT}', \mathsf{vk}^c_{\mathsf{S}}, \mathsf{vk}^c_{\mathsf{R}}, \sigma_{\mathsf{S}})$ and $\mathsf{TD} = (\mathsf{TD}', \mathsf{vk}^t_{\mathsf{S}}, \mathsf{vk}^t_{\mathsf{R}}, \sigma_{\mathsf{R}})$. Output 0 if $(\mathsf{vk}^c_{\mathsf{S}}, \mathsf{vk}^c_{\mathsf{R}}) \neq (\mathsf{vk}^t_{\mathsf{S}}, \mathsf{vk}^t_{\mathsf{R}})$. Otherwise, output 0 if $\mathsf{Verify}(\mathsf{vk}^c_{\mathsf{S}}, \mathsf{CT}', \sigma_{\mathsf{S}}) = 0$. Otherwise, output 0 if $\mathsf{Verify}(\mathsf{vk}^t_{\mathsf{R}}, \mathsf{TD}', \sigma_{\mathsf{R}}) = 0$. Otherwise, output the result of $\mathsf{Test}(\mathsf{CT}', \mathsf{TD}')$.

Correctness obviously holds if PAEKS' and Sig are correct.

5 Security Proofs

Theorem 1. *The proposed transformation provides consistency if* PAEKS' *provides consistency.*

Proof Sketch. If $(c_s, c_r, kw) \neq (t_s, t_r, kw')$, then $\mathsf{vk}_{c_s}||\mathsf{vk}_{c_r}||kw \neq \mathsf{vk}_{t_s}||\mathsf{vk}_{t_r}||kw'$ holds. This is sufficient to break consistency of PAEKS'.

Theorem 2. *The proposed transformation is fully CCTA-secure if* PAEKS' *is fully CI-secure and fully TI-secure, and* Sig *is strong EUF-CMA-secure.*

We prove the theorem by the following two lemmas.

Lemma 1. *The proposed transformation is fully CCA-secure if* PAEKS' *is fully CI-secure and* Sig *is strong EUF-CMA-secure.*

Proof. We use a sequence of games Game_0, Game_1, and Game_2. Let E_i be an event that $\mathcal{A}$ wins in Game_i $(i \in \{0, 1, 2\})$.

Game_0**.** This is the security game of CCA full CI. By definition,

$$\mathsf{Adv}^{\mathsf{CCA\text{-}Full\text{-}CI}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda) = |\Pr[E_0] - 1/2|$$

Game_1**.** This is the same as Game_0 except that if $\mathcal{A}$ sends a valid ciphertext to the test oracle that was not generated by the encryption oracle or was not the challenge ciphertext, then abort. Here, valid means that for a queried ciphertext $\mathsf{CT} = (\mathsf{CT}', \mathsf{vk_S}, \mathsf{vk_R}, \sigma_\mathsf{S})$, $(\mathsf{vk_S}, \mathsf{vk_R}) = (\mathsf{vk}^*_\mathsf{S}, \mathsf{vk}^*_\mathsf{R})$ and $\mathsf{Verify}(\mathsf{vk_S}, \mathsf{CT}', \sigma_\mathsf{S}) = 1$ hold. If this abort event E does not happen, Game_0 and Game_1 are identical. Therefore, $|\Pr[E_0] - \Pr[E_1]| \leq \Pr[E]$ holds by the difference lemma. We show that $\Pr[E]$ is negligible if Sig provides strong EUF-CMA security as follows. Let $\mathcal{A}$ be an adversary of CCA full CI and $\mathcal{C}_1$ be the challenger of the signature scheme Sig. We construct an algorithm $\mathcal{B}_1$ that breaks strong EUF-CMA security if the event happens as follows. $\mathcal{C}_1$ runs $(\mathsf{vk}, \mathsf{sigk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and sends vk to $\mathcal{B}_1$. $\mathcal{B}_1$ prepares $(\mathsf{pp}, \mathsf{pk}^*_\mathsf{S}, \mathsf{pk}^*_\mathsf{R})$ as follows.

- Run $\mathsf{pp}' \leftarrow \mathsf{Setup}'(1^\lambda)$ and set $\mathsf{pp} = \mathsf{pp}'$.
- Set $\mathsf{vk} = \mathsf{vk}^*_\mathsf{S}$. Run $(\mathsf{pk}^{*\prime}_\mathsf{S}, \mathsf{sk}^{*\prime}_\mathsf{S}) \leftarrow \mathsf{KeyGen}'_\mathsf{S}(\mathsf{pp}')$ and set $\mathsf{pk}^*_\mathsf{S} = (\mathsf{pk}^{*\prime}_\mathsf{S}, \mathsf{vk}^*_\mathsf{S})$ and $\mathsf{sk}^*_\mathsf{S} = (\mathsf{sk}^{*\prime}_\mathsf{S}, -)$.
- Run $(\mathsf{pk}^{*\prime}_\mathsf{R}, \mathsf{sk}^{*\prime}_\mathsf{R}) \leftarrow \mathsf{KeyGen}'_\mathsf{R}(\mathsf{pp}')$ and $(\mathsf{vk}^*_\mathsf{R}, \mathsf{sigk}^*_\mathsf{R}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and set $\mathsf{pk}^*_\mathsf{R} = (\mathsf{pk}^{*\prime}_\mathsf{R}, \mathsf{vk}^*_\mathsf{R})$ and $\mathsf{sk}^*_\mathsf{R} = (\mathsf{sk}^{*\prime}_\mathsf{R}, \mathsf{sigk}^*_\mathsf{R})$.

$\mathcal{B}_1$ sends $(\mathsf{pk}^*_\mathsf{S}, \mathsf{pk}^*_\mathsf{R})$ to $\mathcal{A}$. $\mathcal{B}_1$ answers queries issued by $\mathcal{A}$ as follows.

Enc: Upon a query $(\mathsf{pk_R}, kw)$ where $\mathsf{pk_R} = (\mathsf{pk}'_\mathsf{R}, \mathsf{vk_R})$, $\mathcal{B}_1$ runs $\mathsf{CT}' \leftarrow \mathsf{Enc}(\mathsf{pk}^{*\prime}_\mathsf{S}, \mathsf{sk}^{*\prime}_\mathsf{S}, \mathsf{pk}'_\mathsf{R}, \mathsf{vk}^*_\mathsf{S}||\mathsf{vk_R}||kw)$. $\mathcal{B}_1$ sends CT' to $\mathcal{C}_1$ as a signing query. $\mathcal{C}_1$ runs $\sigma \leftarrow \mathsf{Sign}(\mathsf{sigk}, \mathsf{CT}')$ and returns σ to $\mathcal{B}_1$. $\mathcal{B}_1$ sets $\sigma_\mathsf{S} = \sigma$ and returns $(\mathsf{CT}', \mathsf{vk}^*_\mathsf{S}, \mathsf{vk_R}, \sigma_\mathsf{S})$ to $\mathcal{A}$.

Trapdoor: Upon a query $(\mathsf{pk_S}, kw)$ where $\mathsf{pk_S} = (\mathsf{pk}'_\mathsf{S}, \mathsf{vk_S})$, $\mathcal{B}_1$ runs $\mathsf{TD}' \leftarrow \mathsf{Trapdoor}(\mathsf{pk}^{*\prime}_\mathsf{R}, \mathsf{sk}^{*\prime}_\mathsf{R}, \mathsf{pk}'_\mathsf{S}, \mathsf{vk_S}||\mathsf{vk}^*_\mathsf{R}||kw)$ and $\sigma_\mathsf{R} \leftarrow \mathsf{Sign}(\mathsf{sigk}^*_\mathsf{R}, \mathsf{TD}')$, and returns $\mathsf{TD} = (\mathsf{TD}', \mathsf{vk_S}, \mathsf{vk}^*_\mathsf{R}, \sigma_\mathsf{R})$ to $\mathcal{A}$.

Test: Upon a query (kw, CT) where $\mathsf{CT} = (\mathsf{CT}', \mathsf{vk_S}, \mathsf{vk_R}, \sigma_\mathsf{S})$, if $(\mathsf{vk_S}, \mathsf{vk_R}) \neq (\mathsf{vk}^*_\mathsf{S}, \mathsf{vk}^*_\mathsf{R})$, then $\mathcal{B}_1$ returns 0 (remark that a trapdoor, which is internally generated in $\mathcal{O}_{\mathsf{Test}}$, is generated by specifying pk^*_S and pk^*_R and thus the test result is 0 if $(\mathsf{vk_S}, \mathsf{vk_R}) \neq (\mathsf{vk}^*_\mathsf{S}, \mathsf{vk}^*_\mathsf{R})$ regardless of whether σ_S is valid or not). Let $\mathsf{CT} = (\mathsf{CT}', \mathsf{vk}^*_\mathsf{S}, \mathsf{vk}^*_\mathsf{R}, \sigma_\mathsf{S})$. $\mathcal{B}_1$ returns 0 if $\mathsf{Verify}(\mathsf{vk}^*_\mathsf{S}, \mathsf{CT}', \sigma_\mathsf{S}) = 0$. Otherwise, if CT is not a response to an encryption query (i.e., the abort event happens),

then either (1) $\mathcal{B}_1$ did not send CT' to $\mathcal{C}_1$ as a signing query, or (2) $\mathcal{B}_1$ has sent CT' to $\mathcal{C}_1$ as a signing query and has obtained σ as its response. Since CT is not a response to an encryption query, $\sigma \neq \sigma_{\mathsf{S}}$ holds. $\mathcal{B}_1$ sends $(\mathsf{CT}', \sigma_{\mathsf{S}})$ to $\mathcal{C}_1$ and breaks EUF-CMA security in (1), and breaks strong EUF-CMA security in (2). Otherwise, if CT is a response to an encryption query, then $\mathcal{B}_1$ runs $\mathsf{TD}' \leftarrow \mathsf{Trapdoor}(\mathsf{pk}_{\mathsf{R}}^{*\prime}, \mathsf{sk}_{\mathsf{R}}^{*\prime}, \mathsf{pk}_{\mathsf{S}}^{*\prime}, \mathsf{vk}_{\mathsf{S}}^{*}||\mathsf{vk}_{\mathsf{R}}^{*}||kw)$ and returns the result of $\mathsf{Test}'(\mathsf{CT}', \mathsf{TD}')$.

In the challenge phase, $\mathcal{A}$ declares kw_0^* and kw_1^*. $\mathcal{B}_1$ randomly selects $\mathsf{coin} \xleftarrow{\$} \{0,1\}$ and computes $\mathsf{CT}^{*\prime} \leftarrow \mathsf{Enc}(\mathsf{pk}_{\mathsf{S}}^{*\prime}, \mathsf{sk}_{\mathsf{S}}^{*\prime}, \mathsf{pk}_{\mathsf{R}}^{*\prime}, \mathsf{vk}_{\mathsf{S}}^{*}||\mathsf{vk}_{\mathsf{R}}^{*}||kw_{\mathsf{coin}}^{*})$. $\mathcal{B}_1$ sends $\mathsf{CT}^{*\prime}$ to $\mathcal{C}_1$ as a signing query. $\mathcal{C}_1$ runs $\sigma \leftarrow \mathsf{Sign}(\mathsf{sigk}, \mathsf{CT}^{*\prime})$ and returns σ to $\mathcal{B}_1$. $\mathcal{B}_1$ sets $\sigma_{\mathsf{S}}^* = \sigma$ and returns the challenge ciphertext $\mathsf{CT}^* = (\mathsf{CT}^{*\prime}, \mathsf{vk}_{\mathsf{S}}^*, \mathsf{vk}_{\mathsf{R}}^*, \sigma_{\mathsf{S}}^*)$ to $\mathcal{A}$.

$\mathcal{B}_1$ answers to queries issued by $\mathcal{A}$ as in the previous phase. We emphasize that $\mathcal{B}_1$ breaks strong EUF-CMA security if $\mathcal{A}$ sends $(\cdot, (\mathsf{CT}^{*\prime}, \mathsf{vk}_{\mathsf{S}}^*, \mathsf{vk}_{\mathsf{R}}^*, \sigma_{\mathsf{S}}))$ as a test query where $\sigma_{\mathsf{S}} \neq \sigma_{\mathsf{S}}^*$ and $\mathsf{Verify}(\mathsf{vk}_{\mathsf{S}}^*, \mathsf{CT}^{*\prime}, \sigma_{\mathsf{S}}) = 1$. We conclude that

$$\Pr[E] \leq \mathsf{Adv}^{\mathsf{sEUF\text{-}CMA}}_{\mathsf{Sig},\mathcal{B}_1}(\lambda)$$

Game_2**.** This is the same as Game_1 except that upon each test query (kw, CT), return 1 if CT was generated as the response to an encryption query $(\mathsf{pk}_{\mathsf{R}}^*, kw)$, and 0 otherwise. We remark that, if CT was generated as the response to an encryption query $(\mathsf{pk}_{\mathsf{R}}, kw)$ where $\mathsf{pk}_{\mathsf{R}} = (\mathsf{pk}'_{\mathsf{R}}, \mathsf{vk}_{\mathsf{R}}^*)$ and $\mathsf{pk}'_{\mathsf{R}} \neq \mathsf{pk}_{\mathsf{R}}^{*\prime}$, then $\mathcal{B}_2$ returns the same response in Game_1. Since $\mathsf{vk}_{\mathsf{R}}^*$ is contained to the query, the response is not necessarily 0. We make it clear that, if $\mathsf{Test}'(\mathsf{CT}', \mathsf{TD}') = 1$ in Game_1 where $\mathsf{CT}' \leftarrow \mathsf{Enc}'(\mathsf{pk}_{\mathsf{S}}^{*\prime}, \mathsf{sk}_{\mathsf{S}}^{*\prime}, \mathsf{pk}'_{\mathsf{R}}, \mathsf{vk}_{\mathsf{S}}^{*}||\mathsf{vk}_{\mathsf{R}}^{*}||kw)$ and $\mathsf{TD}' \leftarrow \mathsf{Trapdoor}'(\mathsf{pk}_{\mathsf{R}}^{*\prime}, \mathsf{sk}_{\mathsf{R}}^{*\prime}, \mathsf{pk}_{\mathsf{S}}^{*\prime}, \mathsf{vk}_{\mathsf{S}}^{*}||\mathsf{vk}_{\mathsf{R}}^{*}||kw)$, then $\mathcal{B}_2$ returns 1. Obviously, this contradicts consistency (i.e., a different receiver's public key $\mathsf{pk}'_{\mathsf{R}} \neq \mathsf{pk}_{\mathsf{R}}^{*\prime}$ is specified for encryption, even the same keyword $\mathsf{vk}_{\mathsf{S}}^{*}||\mathsf{vk}_{\mathsf{R}}^{*}||kw$ is specified). However, consistency is irrelevant to the privacy argument here and this does not gain information that $\mathcal{A}$ obtains. Concretely, the fact that no keyword information is leaked from ciphertexts is independent of consistency. For example, in a PAEKS scheme that does not satisfy consistency, both ciphertexts and trapdoors can be uniformly random, and the Test algorithm always outputs 1. This is evident from the fact that, by doing so, one can construct a PAEKS scheme that provides correctness, and is secure in the sense that no keyword information is leaked from either ciphertexts or trapdoors, although such a scheme is meaningless in practice. The crucial point here is that we respond to $\mathcal{A}$ with the same output as that of the Test' algorithm, without actually executing the Test' algorithm, and we do not have to consider an algorithm that breaks consistency. Since all valid ciphertexts contained in test queries are generated via the encryption oracle in Game_1 unless E happens, the simulation of the test oracle is identical. Thus,

$$\Pr[E_1 \wedge \neg E] = \Pr[E_2 \wedge \neg E]$$

We show that $|\Pr[E_2 \wedge \neg E] - \frac{1}{2}|$ is negligible if PAEKS' is fully CI secure. Let $\mathcal{C}_2$ be the challenger of the PAEKS scheme PAEKS'. We construct an algorithm $\mathcal{B}_2$ that breaks full CI security as follows. $\mathcal{C}_2$ runs $\mathsf{pp}' \leftarrow \mathsf{Setup}'(1^\lambda)$, $(\mathsf{pk}_{\mathsf{S}}^{*\prime}, \mathsf{sk}_{\mathsf{S}}^{*\prime}) \leftarrow \mathsf{KeyGen}'_{\mathsf{S}}(\mathsf{pp}')$, and $(\mathsf{pk}_{\mathsf{R}}^{*\prime}, \mathsf{sk}_{\mathsf{R}}^{*\prime}) \leftarrow \mathsf{KeyGen}'_{\mathsf{R}}(\mathsf{pp}')$, and sends $(\mathsf{pk}_{\mathsf{S}}^{*\prime}, \mathsf{pk}_{\mathsf{R}}^{*\prime})$ to $\mathcal{B}_2$. $\mathcal{B}_2$ runs $(\mathsf{vk}_{\mathsf{S}}^*, \mathsf{sigk}_{\mathsf{S}}^*) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $(\mathsf{vk}_{\mathsf{R}}^*, \mathsf{sigk}_{\mathsf{R}}^*) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and sends $\mathsf{pk}_{\mathsf{S}}^* = (\mathsf{pk}_{\mathsf{S}}^{*\prime}, \mathsf{vk}_{\mathsf{S}}^*)$ and $\mathsf{pk}_{\mathsf{R}}^* = (\mathsf{pk}_{\mathsf{R}}^{*\prime}, \mathsf{vk}_{\mathsf{R}}^*)$ to $\mathcal{A}$. $\mathcal{B}_2$ responds to queries from $\mathcal{A}$ as follows.

Enc: Upon a query $(\mathsf{pk}_{\mathsf{R}}, kw)$ where $\mathsf{pk}_{\mathsf{R}} = (\mathsf{pk}'_{\mathsf{R}}, \mathsf{vk}_{\mathsf{R}})$, $\mathcal{B}_1$ sends $(\mathsf{pk}'_{\mathsf{R}}, \mathsf{vk}_{\mathsf{S}}^*||\mathsf{vk}_{\mathsf{R}}||kw)$ to $\mathcal{C}_2$ as an encryption query. $\mathcal{C}_2$ runs $\mathsf{CT}' \leftarrow \mathsf{Enc}(\mathsf{pk}_{\mathsf{S}}^{*\prime}, \mathsf{sk}_{\mathsf{S}}^{*\prime}, \mathsf{pk}'_{\mathsf{R}}, \mathsf{vk}_{\mathsf{S}}^*||\mathsf{vk}_{\mathsf{R}}||kw)$ and returns CT' to $\mathcal{B}_2$. $\mathcal{B}_2$ computes $\sigma_{\mathsf{S}} \leftarrow \mathsf{Sign}(\mathsf{sigk}_{\mathsf{S}}^*, \mathsf{CT}')$ and returns $\mathsf{CT} = (\mathsf{CT}', \mathsf{vk}_{\mathsf{S}}^*, \mathsf{vk}_{\mathsf{R}}, \sigma_{\mathsf{S}})$ to $\mathcal{A}$.

Trapdoor: Upon a query $(\mathsf{pk}_\mathsf{S}, kw)$ where $\mathsf{pk}_\mathsf{S} = (\mathsf{pk}'_\mathsf{S}, \mathsf{vk}_\mathsf{S})$, $\mathcal{B}_1$ sends $(\mathsf{pk}'_\mathsf{S}, \mathsf{vk}_\mathsf{S}||\mathsf{vk}^*_\mathsf{R}||kw)$ to $\mathcal{C}_2$ as a trapdoor query. $\mathcal{C}_2$ runs $\mathsf{TD}' \leftarrow \mathsf{Trapdoor}(\mathsf{pk}^{*\prime}_\mathsf{R}, \mathsf{sk}^{*\prime}_\mathsf{R}, \mathsf{pk}'_\mathsf{S}, \mathsf{vk}_\mathsf{S}||\mathsf{vk}^*_\mathsf{R}||kw)$ and returns TD' to $\mathcal{B}_2$. $\mathcal{B}_2$ computes $\sigma_\mathsf{R} \leftarrow \mathsf{Sign}(\mathsf{sigk}^*_\mathsf{R}, \mathsf{TD}')$, and returns $\mathsf{TD} = (\mathsf{TD}', \mathsf{vk}_\mathsf{S}, \mathsf{vk}^*_\mathsf{R}, \sigma_\mathsf{R})$ to $\mathcal{A}$.

Test: Upon a query (kw, CT) where $\mathsf{CT} = (\mathsf{CT}', \mathsf{vk}_\mathsf{S}, \mathsf{vk}_\mathsf{R}, \sigma_\mathsf{S})$, if $(\mathsf{vk}_\mathsf{S}, \mathsf{vk}_\mathsf{R}) \neq (\mathsf{vk}^*_\mathsf{S}, \mathsf{vk}^*_\mathsf{R})$, then $\mathcal{B}_2$ returns 0. Let $\mathsf{CT} = (\mathsf{CT}', \mathsf{vk}^*_\mathsf{S}, \mathsf{vk}^*_\mathsf{R}, \sigma_\mathsf{S})$. $\mathcal{B}_2$ returns 0 if $\mathsf{Verify}(\mathsf{vk}^*_\mathsf{S}, \mathsf{CT}', \sigma_\mathsf{S}) = 0$. Otherwise, in this game, CT is the response to an encryption query $(\mathsf{pk}_\mathsf{R}, kw')$ where $\mathsf{pk}_\mathsf{R} = (\mathsf{pk}'_\mathsf{R}, \mathsf{vk}^*_\mathsf{R})$. If $\mathsf{pk}'_\mathsf{R} \neq \mathsf{pk}^{*\prime}_\mathsf{R}$, then $\mathcal{B}_2$ returns as in Game_1. Otherwise, if $\mathsf{pk}'_\mathsf{R} = \mathsf{pk}^{*\prime}_\mathsf{R}$ (i.e., $\mathsf{pk}_\mathsf{R} = \mathsf{pk}^*_\mathsf{R}$), then $\mathcal{B}_2$ returns 1 if $kw = kw'$, and 0 otherwise.

In the challenge phase, $\mathcal{A}$ declares kw^*_0 and kw^*_1. $\mathcal{B}_2$ sends $\mathsf{vk}^*_\mathsf{S}||\mathsf{vk}^*_\mathsf{R}||kw^*_0$ and $\mathsf{vk}^*_\mathsf{S}||\mathsf{vk}^*_\mathsf{R}||kw^*_1$ to $\mathcal{C}_2$ as the challenge keywords. Here, since $\mathcal{A}$ did not send kw^*_0 and kw^*_1 to the trapdoor oracle, $\mathcal{B}_2$ did not send $\mathsf{vk}^*_\mathsf{S}||\mathsf{vk}^*_\mathsf{R}||kw^*_0$ and $\mathsf{vk}^*_\mathsf{S}||\mathsf{vk}^*_\mathsf{R}||kw^*_1$ to $\mathcal{C}_2$ as trapdoor queries. $\mathcal{C}_2$ randomly selects $\mathsf{coin} \xleftarrow{\$} \{0,1\}$, computes the challenge ciphertext $\mathsf{CT}^{*\prime} \leftarrow \mathsf{Enc}(\mathsf{pk}^{*\prime}_\mathsf{S}, \mathsf{sk}^{*\prime}_\mathsf{S}, \mathsf{pk}^{*\prime}_\mathsf{R}, \mathsf{vk}^*_\mathsf{S}||\mathsf{vk}^*_\mathsf{R}||kw^*_\mathsf{coin})$, and returns $\mathsf{CT}^{*\prime}$ to $\mathcal{B}_2$. $\mathcal{B}_2$ computes $\sigma^*_\mathsf{S} \leftarrow \mathsf{Sign}(\mathsf{sigk}^*_\mathsf{S}, \mathsf{CT}^{*\prime})$ and returns the challenge ciphertext $\mathsf{CT}^* = (\mathsf{CT}^{*\prime}, \mathsf{vk}^*_\mathsf{S}, \mathsf{vk}^*_\mathsf{R}, \sigma^*_\mathsf{S})$ to $\mathcal{A}$. $\mathcal{B}_2$ answers queries issued by $\mathcal{A}$ as follows.

Enc: The same as in the previous phase.

Trapdoor: The same as in the previous phase.

Test: The same as in the previous phase. $\mathcal{A}$ may send (kw^*_0, CT) (resp. (kw^*_1, CT)) where $\mathsf{CT} \neq \mathsf{CT}^*$. If CT is the response to an encryption query $(\mathsf{pk}^*_\mathsf{R}, kw^*_0)$ (resp. $(\mathsf{pk}^*_\mathsf{R}, kw^*_1)$), then $\mathcal{B}_2$ returns 1, and 0 otherwise.

Finally, $\mathcal{A}$ outputs coin'. $\mathcal{B}_2$ outputs coin' and breaks full CI security with the same advantage that $\mathcal{A}$ breaks CCA full CI security. Thus, we conclude that

$$|\Pr[E_2 \wedge \neg E] - \frac{1}{2}| \leq \mathsf{Adv}^{\mathsf{Full\text{-}CI}}_{\mathsf{PAEKS},\mathcal{B}_2}(\lambda)$$

Putting it all together, we have

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{CCA\text{-}FullCI}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda) &= |\Pr[E_0] - 1/2| \\
&= |\Pr[E_0] - \Pr[E_1] + \Pr[E_1] - 1/2| \\
&\leq |\Pr[E_0] - \Pr[E_1]| + |\Pr[E_1] - 1/2| \\
&\leq \Pr[E] + |\Pr[E_1] - 1/2| \\
&= \Pr[E] + |\Pr[E_1 \wedge E] + \Pr[E_1 \wedge \neg E] - 1/2| \\
&\leq \Pr[E] + \Pr[E_1 \wedge E] + |\Pr[E_1 \wedge \neg E] - 1/2| \\
&= \Pr[E] + \Pr[E_1 \wedge E] + |\Pr[E_2 \wedge \neg E] - 1/2| \\
&\leq \Pr[E] + \Pr[E] + |\Pr[E_2 \wedge \neg E] - 1/2| \\
&\leq 2 \cdot \mathsf{Adv}^{\mathsf{sEUF\text{-}CMA}}_{\mathsf{Sig},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{Full\text{-}CI}}_{\mathsf{PAEKS},\mathcal{B}_2}(\lambda)
\end{aligned}$$

□

Lemma 2. *The proposed transformation is fully CTA-secure if* PAEKS′ *is fully TI-secure and* Sig *is strong EUF-CMA-secure. Moreover, the proposed transformation is CTA-secure if* PAEKS′ *is TI-secure and* Sig *is strong EUF-CMA-secure.*

Proof. We use a sequence of games Game_0, Game_1, and Game_2. Let E_i be an event that $\mathcal{A}$ wins in Game_i ($i \in \{0,1,2\}$).

Game_0**.** This is the security game of CTA full TI. By definition,

$$\mathsf{Adv}^{\mathsf{CTA\text{-}Full\text{-}TI}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda) = |\Pr[E_0] - 1/2|$$

Game_1**.** This is the same as Game_0 except that if $\mathcal{A}$ sends a valid trapdoor to the test oracle that was not generated by the trapdoor oracle or was not the challenge trapdoor, then abort. Here, valid means that for a queried trapdoor $\mathsf{TD} = (\mathsf{TD}', \mathsf{vk_S}, \mathsf{vk_R}, \sigma_\mathsf{R})$, $(\mathsf{vk_S}, \mathsf{vk_R}) = (\mathsf{vk}^*_\mathsf{S}, \mathsf{vk}^*_\mathsf{R})$ and $\mathsf{Verify}(\mathsf{vk_R}, \mathsf{TD}', \sigma_\mathsf{R}) = 1$ hold. If this abort event does not happen, Game_0 and Game_1 are identical. Thus,

$$|\Pr[E_0] - \Pr[E_1]| \leq \Pr[E]$$

holds by the difference lemma.

We show that $\Pr[E]$ is negligible if Sig provides strong EUF-CMA security as follows. Let $\mathcal{A}$ be an adversary of CTA full TI and $\mathcal{C}_1$ be the challenger of the signature scheme Sig. We construct an algorithm $\mathcal{B}_1$ that breaks strong EUF-CMA security if the event happens as follows. $\mathcal{C}_1$ runs $(\mathsf{vk}, \mathsf{sigk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and sends vk to $\mathcal{B}_1$. $\mathcal{B}_1$ prepares $(\mathsf{pp}, \mathsf{pk}^*_\mathsf{S}, \mathsf{pk}^*_\mathsf{R})$ as follows.

- Run $\mathsf{pp}' \leftarrow \mathsf{Setup}'(1^\lambda)$ and set $\mathsf{pp} = \mathsf{pp}'$.
- Run $(\mathsf{pk}^{*\prime}_\mathsf{S}, \mathsf{sk}^{*\prime}_\mathsf{S}) \leftarrow \mathsf{KeyGen}'_\mathsf{S}(\mathsf{pp}')$ and $(\mathsf{vk_S}, \mathsf{sigk_S}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and set $\mathsf{pk}^*_\mathsf{S} = (\mathsf{pk}^{*\prime}_\mathsf{S}, \mathsf{vk}^*_\mathsf{S})$ and $\mathsf{sk}^*_\mathsf{S} = (\mathsf{sk}^{*\prime}_\mathsf{S}, \mathsf{sigk}^*_\mathsf{S})$.
- Set $\mathsf{vk} = \mathsf{vk}^*_\mathsf{R}$. Run $(\mathsf{pk}^{*\prime}_\mathsf{R}, \mathsf{sk}^{*\prime}_\mathsf{R}) \leftarrow \mathsf{KeyGen}'_\mathsf{R}(\mathsf{pp}')$ and set $\mathsf{pk}^*_\mathsf{R} = (\mathsf{pk}^{*\prime}_\mathsf{R}, \mathsf{vk}^*_\mathsf{R})$ and $\mathsf{sk}^*_\mathsf{R} = (\mathsf{sk}^{*\prime}_\mathsf{R}, -)$.

$\mathcal{B}_1$ sends $(\mathsf{pk}^*_\mathsf{S}, \mathsf{pk}^*_\mathsf{R})$ to $\mathcal{A}$. $\mathcal{B}_1$ answers queries issued by $\mathcal{A}$ as follows.

Enc: Upon a query $(\mathsf{pk_R}, kw)$ where $\mathsf{pk_R} = (\mathsf{pk}'_\mathsf{R}, \mathsf{vk_R})$, $\mathcal{B}_1$ runs $\mathsf{CT}' \leftarrow \mathsf{Enc}(\mathsf{pk}^{*\prime}_\mathsf{S}, \mathsf{sk}^{*\prime}_\mathsf{S}, \mathsf{pk}'_\mathsf{R}, \mathsf{vk}^*_\mathsf{S}||\mathsf{vk_R}||kw)$ and $\sigma_\mathsf{S} \leftarrow \mathsf{Sign}(\mathsf{sigk}^*_\mathsf{S}, \mathsf{CT}')$, and returns $(\mathsf{CT}', \mathsf{vk}^*_\mathsf{S}, \mathsf{vk_R}, \sigma_\mathsf{S})$ to $\mathcal{A}$.

Trapdoor: Upon a query $(\mathsf{pk_S}, kw)$ where $\mathsf{pk_S} = (\mathsf{pk}'_\mathsf{S}, \mathsf{vk_S})$, $\mathcal{B}_1$ runs $\mathsf{TD}' \leftarrow \mathsf{Trapdoor}(\mathsf{pk}^{*\prime}_\mathsf{R}, \mathsf{sk}^{*\prime}_\mathsf{R}, \mathsf{pk}'_\mathsf{S}, \mathsf{vk_S}||\mathsf{vk}^*_\mathsf{R}||kw)$. $\mathcal{B}_1$ sends TD' to $\mathcal{C}_1$ as a signing query. $\mathcal{C}_1$ runs $\sigma \leftarrow \mathsf{Sign}(\mathsf{sigk}, \mathsf{TD}')$ and returns σ to $\mathcal{B}_1$. $\mathcal{B}_1$ sets $\sigma_\mathsf{R} = \sigma$ and returns $\mathsf{TD} = (\mathsf{TD}', \mathsf{vk_S}, \mathsf{vk}^*_\mathsf{R}, \sigma_\mathsf{R})$ to $\mathcal{A}$.

Test: Upon a query (kw, TD) where $\mathsf{TD} = (\mathsf{TD}', \mathsf{vk_S}, \mathsf{vk_R}, \sigma_\mathsf{R})$, if $(\mathsf{vk_S}, \mathsf{vk_R}) \neq (\mathsf{vk}^*_\mathsf{S}, \mathsf{vk}^*_\mathsf{R})$, then $\mathcal{B}_1$ returns 0 (remark that a ciphertext, which is internally generated in $\mathcal{O}_{\mathsf{Test}}$, is generated by specifying pk^*_S and pk^*_R and thus the test result is 0 if $(\mathsf{vk_S}, \mathsf{vk_R}) \neq (\mathsf{vk}^*_\mathsf{S}, \mathsf{vk}^*_\mathsf{R})$ regardless of whether σ_R is valid or not). Let $\mathsf{TD} = (\mathsf{TD}', \mathsf{vk}^*_\mathsf{S}, \mathsf{vk}^*_\mathsf{R}, \sigma_\mathsf{R})$. $\mathcal{B}_1$ returns 0 if $\mathsf{Verify}(\mathsf{vk}^*_\mathsf{R}, \mathsf{TD}', \sigma_\mathsf{R}) = 0$. Otherwise, if TD is not a response to a trapdoor query (i.e., the abort event happens), then either (1) $\mathcal{B}_1$ did not send TD' to $\mathcal{C}_1$ as a signing query, or (2) $\mathcal{B}_1$ has sent TD' to $\mathcal{C}_1$ as a signing query and has obtained σ as its response but $\sigma \neq \sigma_\mathsf{R}$. $\mathcal{B}_1$ sends $(\mathsf{TD}', \sigma_\mathsf{R})$ to $\mathcal{C}_1$ and breaks EUF-CMA security in (1) and breaks strong EUF-CMA security in (2). Otherwise, if TD is a response to a trapdoor query, then $\mathcal{B}_1$ runs $\mathsf{CT}' \leftarrow \mathsf{Enc}(\mathsf{pk}^{*\prime}_\mathsf{S}, \mathsf{sk}^{*\prime}_\mathsf{S}, \mathsf{pk}^{*\prime}_\mathsf{R}, \mathsf{vk}^*_\mathsf{S}||\mathsf{vk}^*_\mathsf{R}||kw)$ and returns the result of $\mathsf{Test}'(\mathsf{CT}', \mathsf{TD}')$.

In the challenge phase, $\mathcal{A}$ declares kw^*_0 and kw^*_1. $\mathcal{B}_1$ randomly selects $\mathsf{coin} \xleftarrow{\$} \{0,1\}$ and computes $\mathsf{TD}^{*\prime} \leftarrow \mathsf{Trapdoor}(\mathsf{pk}^{*\prime}_\mathsf{R}, \mathsf{sk}^{*\prime}_\mathsf{R}, \mathsf{pk}^{*\prime}_\mathsf{S}, \mathsf{vk}^*_\mathsf{S}||\mathsf{vk}^*_\mathsf{R}||kw^*_\mathsf{coin})$. $\mathcal{B}_1$ sends $\mathsf{TD}^{*\prime}$ to $\mathcal{C}_1$ as a signing query. $\mathcal{C}_1$ runs $\sigma \leftarrow \mathsf{Sign}(\mathsf{sigk}, \mathsf{TD}^{*\prime})$ and returns σ to $\mathcal{B}_1$. $\mathcal{B}_1$ sets $\sigma^*_\mathsf{R} = \sigma$ and returns the challenge trapdoor $\mathsf{TD}^* = (\mathsf{TD}^{*\prime}, \mathsf{vk}^*_\mathsf{S}, \mathsf{vk}^*_\mathsf{R}, \sigma^*_\mathsf{R})$ to $\mathcal{A}$.

$\mathcal{B}_1$ responds to queries from $\mathcal{A}$ as in the previous phase. We emphasize that $\mathcal{B}_1$ breaks strong EUF-CMA security if $\mathcal{A}$ sends $(\cdot, (\mathsf{TD}^{*\prime}, \mathsf{vk}^*_{\mathsf{S}}, \mathsf{vk}^*_{\mathsf{R}}, \sigma_{\mathsf{R}}))$ as a test query where $\sigma_{\mathsf{R}} \neq \sigma^*_{\mathsf{R}}$ and $\mathsf{Verify}(\mathsf{vk}^*_{\mathsf{R}}, \mathsf{TD}^{*\prime}, \sigma_{\mathsf{R}}) = 1$. We conclude that

$$\Pr[E] \leq \mathsf{Adv}^{\mathsf{sEUF\text{-}CMA}}_{\mathsf{Sig},\mathcal{B}_1}(\lambda)$$

Game_2**.** This is the same as Game_1 except that upon each test query (kw, TD), return 1 if TD was generated as the response to a query $(\mathsf{pk}^*_{\mathsf{S}}, kw)$, and 0 otherwise. We remark that, if TD was generated as the response to a query $(\mathsf{pk}_{\mathsf{S}}, kw)$ where $\mathsf{pk}_{\mathsf{S}} = (\mathsf{pk}'_{\mathsf{S}}, \mathsf{vk}^*_{\mathsf{S}})$ and $\mathsf{pk}'_{\mathsf{S}} \neq \mathsf{pk}^{*\prime}_{\mathsf{S}}$, then $\mathcal{B}_2$ returns the same response in Game_1. We make it clear that, if $\mathsf{Test}'(\mathsf{CT}', \mathsf{TD}') = 1$ in Game_1 where $\mathsf{CT}' \leftarrow \mathsf{Enc}'(\mathsf{pk}^{*\prime}_{\mathsf{S}}, \mathsf{sk}^{*\prime}_{\mathsf{S}}, \mathsf{pk}^{*\prime}_{\mathsf{R}}, \mathsf{vk}^*_{\mathsf{S}}||\mathsf{vk}^*_{\mathsf{R}}||kw)$ and $\mathsf{TD}' \leftarrow \mathsf{Trapdoor}'(\mathsf{pk}^{*\prime}_{\mathsf{R}}, \mathsf{sk}^{*\prime}_{\mathsf{R}}, \mathsf{pk}'_{\mathsf{S}}, \mathsf{vk}^*_{\mathsf{S}}||\mathsf{vk}^*_{\mathsf{R}}||kw)$, then $\mathcal{B}_2$ returns 1. Obviously, this contradicts consistency (i.e., a different sender's public key $\mathsf{pk}'_{\mathsf{S}} \neq \mathsf{pk}^{*\prime}_{\mathsf{S}}$ is specified for generating trapdoor, even the same keyword $\mathsf{vk}^*_{\mathsf{S}}||\mathsf{vk}^*_{\mathsf{R}}||kw$ is specified). However, consistency is irrelevant to the privacy argument here and this does not gain information that $\mathcal{A}$ obtains. The crucial point here is that we respond to $\mathcal{A}$ with the same output as that of the Test' algorithm, without actually executing the Test' algorithm. Since all valid trapdoors contained in test queries are generated via the trapdoor oracle in Game_1 unless E happens, the simulation of the test oracle is identical. Thus,

$$\Pr[E_1 \wedge \neg E] = \Pr[E_2 \wedge \neg E]$$

We show that $|\Pr[E_2 \wedge \neg E] - \frac{1}{2}|$ is negligible if PAEKS' is fully TI secure. Let $\mathcal{C}_2$ be the challenger of the PAEKS scheme PAEKS'. We construct an algorithm $\mathcal{B}_2$ that breaks full TI security as follows. $\mathcal{C}_2$ runs $\mathsf{pp}' \leftarrow \mathsf{Setup}'(1^\lambda)$, $(\mathsf{pk}^{*\prime}_{\mathsf{S}}, \mathsf{sk}^{*\prime}_{\mathsf{S}}) \leftarrow \mathsf{KeyGen}'_{\mathsf{S}}(\mathsf{pp}')$, and $(\mathsf{pk}^{*\prime}_{\mathsf{R}}, \mathsf{sk}^{*\prime}_{\mathsf{R}}) \leftarrow \mathsf{KeyGen}'_{\mathsf{R}}(\mathsf{pp}')$, and sends $(\mathsf{pk}^{*\prime}_{\mathsf{S}}, \mathsf{pk}^{*\prime}_{\mathsf{R}})$ to $\mathcal{B}_2$. $\mathcal{B}_2$ runs $(\mathsf{vk}^*_{\mathsf{S}}, \mathsf{sigk}^*_{\mathsf{S}}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $(\mathsf{vk}^*_{\mathsf{R}}, \mathsf{sigk}^*_{\mathsf{R}}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and sends $\mathsf{pk}^*_{\mathsf{S}} = (\mathsf{pk}^{*\prime}_{\mathsf{S}}, \mathsf{vk}^*_{\mathsf{S}})$ and $\mathsf{pk}^*_{\mathsf{R}} = (\mathsf{pk}^{*\prime}_{\mathsf{R}}, \mathsf{vk}^*_{\mathsf{R}})$ to $\mathcal{A}$. $\mathcal{B}_2$ responds to queries from $\mathcal{A}$ as follows.

Enc: Upon a query $(\mathsf{pk}_{\mathsf{R}}, kw)$ where $\mathsf{pk}_{\mathsf{R}} = (\mathsf{pk}'_{\mathsf{R}}, \mathsf{vk}_{\mathsf{R}})$, $\mathcal{B}_1$ sends $(\mathsf{pk}'_{\mathsf{R}}, \mathsf{vk}^*_{\mathsf{S}}||\mathsf{vk}_{\mathsf{R}}||kw)$ to $\mathcal{C}_2$ as an encryption query. $\mathcal{C}_2$ runs $\mathsf{CT}' \leftarrow \mathsf{Enc}(\mathsf{pk}^{*\prime}_{\mathsf{S}}, \mathsf{sk}^{*\prime}_{\mathsf{S}}, \mathsf{pk}'_{\mathsf{R}}, \mathsf{vk}^*_{\mathsf{S}}||\mathsf{vk}_{\mathsf{R}}||kw)$ and returns CT' to $\mathcal{B}_2$. $\mathcal{B}_2$ computes $\sigma_{\mathsf{S}} \leftarrow \mathsf{Sign}(\mathsf{sigk}^*_{\mathsf{S}}, \mathsf{CT}')$ and returns $\mathsf{CT} = (\mathsf{CT}', \mathsf{vk}^*_{\mathsf{S}}, \mathsf{vk}_{\mathsf{R}}, \sigma_{\mathsf{S}})$ to $\mathcal{A}$.

Trapdoor: Upon a query $(\mathsf{pk}_{\mathsf{S}}, kw)$ where $\mathsf{pk}_{\mathsf{S}} = (\mathsf{pk}'_{\mathsf{S}}, \mathsf{vk}_{\mathsf{S}})$, $\mathcal{B}_1$ sends $(\mathsf{pk}'_{\mathsf{S}}, \mathsf{vk}_{\mathsf{S}}||\mathsf{vk}^*_{\mathsf{R}}||kw)$ to $\mathcal{C}_2$ as a trapdoor query. $\mathcal{C}_2$ runs $\mathsf{TD}' \leftarrow \mathsf{Trapdoor}(\mathsf{pk}^{*\prime}_{\mathsf{R}}, \mathsf{sk}^{*\prime}_{\mathsf{R}}, \mathsf{pk}'_{\mathsf{S}}, \mathsf{vk}_{\mathsf{S}}||\mathsf{vk}^*_{\mathsf{R}}||kw)$ and returns TD' to $\mathcal{B}_2$. $\mathcal{B}_2$ computes $\sigma_{\mathsf{R}} \leftarrow \mathsf{Sign}(\mathsf{sigk}^*_{\mathsf{R}}, \mathsf{TD}')$, and returns $\mathsf{TD} = (\mathsf{TD}', \mathsf{vk}_{\mathsf{S}}, \mathsf{vk}^*_{\mathsf{R}}, \sigma_{\mathsf{R}})$ to $\mathcal{A}$.

Test: Upon a query (kw, TD) where $\mathsf{TD} = (\mathsf{TD}', \mathsf{vk}_{\mathsf{S}}, \mathsf{vk}_{\mathsf{R}}, \sigma_{\mathsf{R}})$, if $(\mathsf{vk}_{\mathsf{S}}, \mathsf{vk}_{\mathsf{R}}) \neq (\mathsf{vk}^*_{\mathsf{S}}, \mathsf{vk}^*_{\mathsf{R}})$, then $\mathcal{B}_2$ returns 0. Let $\mathsf{TD} = (\mathsf{TD}', \mathsf{vk}^*_{\mathsf{S}}, \mathsf{vk}^*_{\mathsf{R}}, \sigma_{\mathsf{R}})$. $\mathcal{B}_2$ returns 0 if $\mathsf{Verify}(\mathsf{vk}^*_{\mathsf{R}}, \mathsf{TD}', \sigma_{\mathsf{R}}) = 0$. Otherwise, in this game, TD is the response to a trapdoor query $(\mathsf{pk}_{\mathsf{S}}, kw')$ where $\mathsf{pk}_{\mathsf{S}} = (\mathsf{pk}'_{\mathsf{S}}, \mathsf{vk}^*_{\mathsf{S}})$. If $\mathsf{pk}'_{\mathsf{S}} \neq \mathsf{pk}^{*\prime}_{\mathsf{S}}$, then $\mathcal{B}_2$ returns as in Game_1. Otherwise, if $\mathsf{pk}'_{\mathsf{S}} = \mathsf{pk}^{*\prime}_{\mathsf{S}}$ (i.e., $\mathsf{pk}_{\mathsf{S}} = \mathsf{pk}^*_{\mathsf{S}}$), then $\mathcal{B}_2$ returns 1 if $kw = kw'$, and 0 otherwise.

In the challenge phase, $\mathcal{A}$ declares kw^*_0 and kw^*_1. $\mathcal{B}_2$ sends $\mathsf{vk}^*_{\mathsf{S}}||\mathsf{vk}^*_{\mathsf{R}}||kw^*_0$ and $\mathsf{vk}^*_{\mathsf{S}}||\mathsf{vk}^*_{\mathsf{R}}||kw^*_1$ to $\mathcal{C}_2$ as the challenge keywords. Here, since $\mathcal{A}$ did not send kw^*_0 and kw^*_1 to the encryption oracle, $\mathcal{B}_2$ did not send $\mathsf{vk}^*_{\mathsf{S}}||\mathsf{vk}^*_{\mathsf{R}}||kw^*_0$ and $\mathsf{vk}^*_{\mathsf{S}}||\mathsf{vk}^*_{\mathsf{R}}||kw^*_1$ to $\mathcal{C}_2$ as encryption queries. $\mathcal{C}_2$ randomly selects $\mathsf{coin} \xleftarrow{\$} \{0,1\}$, computes the challenge trapdoor $\mathsf{TD}^{*\prime} \leftarrow \mathsf{Trapdoor}(\mathsf{pk}^{*\prime}_{\mathsf{R}}, \mathsf{sk}^{*\prime}_{\mathsf{R}}, \mathsf{pk}^{*\prime}_{\mathsf{S}}, \mathsf{vk}^*_{\mathsf{S}}||\mathsf{vk}^*_{\mathsf{R}}||kw^*_{\mathsf{coin}})$, and returns $\mathsf{TD}^{*\prime}$ to $\mathcal{B}_2$. $\mathcal{B}_2$ computes $\sigma^*_{\mathsf{R}} \leftarrow \mathsf{Sign}(\mathsf{sigk}^*_{\mathsf{R}}, \mathsf{TD}^{*\prime})$ and returns the challenge trapdoor $\mathsf{TD}^* = (\mathsf{TD}^{*\prime}, \mathsf{vk}^*_{\mathsf{S}}, \mathsf{vk}^*_{\mathsf{R}}, \sigma^*_{\mathsf{R}})$ to $\mathcal{A}$. $\mathcal{B}_2$ answers queries issued by $\mathcal{A}$ as follows.

Enc: The same as in the previous phase.

Trapdoor: The same as in the previous phase.

Test: The same as in the previous phase. $\mathcal{A}$ may send (kw_0^*, TD) (resp. (kw_1^*, TD)) where $\mathsf{TD} \neq \mathsf{TD}^*$. If TD is the response to a trapdoor query $(\mathsf{pk}_\mathsf{S}^*, kw_0^*)$ (resp. $(\mathsf{pk}_\mathsf{S}^*, kw_1^*)$), then $\mathcal{B}_2$ returns 1, and 0 otherwise.

Finally, $\mathcal{A}$ outputs coin'. $\mathcal{B}_2$ outputs coin' and breaks full TI security with the same advantage that $\mathcal{A}$ breaks CTA full TI security. Thus, we conclude that

$$|\Pr[E_2 \wedge \neg E] - \frac{1}{2}| \leq \mathsf{Adv}^{\mathsf{Full\text{-}TI}}_{\mathsf{PAEKS},\mathcal{B}_2}(\lambda)$$

Putting it all together, we have

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{CTA\text{-}Full\text{-}TI}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda) &= |\Pr[E_0] - 1/2| \\
&= |\Pr[E_0] - \Pr[E_1] + \Pr[E_1] - 1/2| \\
&\leq |\Pr[E_0] - \Pr[E_1]| + |\Pr[E_1] - 1/2| \\
&\leq \Pr[E] + |\Pr[E_1] - 1/2| \\
&= \Pr[E] + |\Pr[E_1 \wedge E] + \Pr[E_1 \wedge \neg E] - 1/2| \\
&\leq \Pr[E] + \Pr[E_1 \wedge E] + |\Pr[E_1 \wedge \neg E] - 1/2| \\
&= \Pr[E] + \Pr[E_1 \wedge E] + |\Pr[E_2 \wedge \neg E] - 1/2| \\
&\leq \Pr[E] + \Pr[E] + |\Pr[E_2 \wedge \neg E] - 1/2| \\
&\leq 2 \cdot \mathsf{Adv}^{\mathsf{sEUF\text{-}CMA}}_{\mathsf{Sig},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{Full\text{-}TI}}_{\mathsf{PAEKS},\mathcal{B}_2}(\lambda)
\end{aligned}$$

□

6 Efficient Search

Since each ciphertext and trapdoor depend not only on the keyword but also on the sender and the receiver, the search complexity in PAEKS is linear in the number of senders. To address this inefficiency, several schemes have been proposed, e.g., [10, 14, 29, 30, 32, 39–41, 45, 47]. However, although not all of these schemes are affected, several vulnerabilities have subsequently been reported [20, 21, 41, 44, 46]. We believe that these vulnerabilities stem from modifications to the original PAEKS syntax introduced to improve search efficiency, such as sender-group-oriented constructions and the introduction of additional entities (e.g., proxies or servers). In contrast, our transformation operates on plain PAEKS schemes and does not require any modification of the underlying syntax.

In the Test algorithm, which takes $\mathsf{CT} = (\mathsf{CT}', \mathsf{vk}_\mathsf{S}^c, \mathsf{vk}_\mathsf{R}^c, \sigma_\mathsf{S})$ and $\mathsf{TD} = (\mathsf{TD}', \mathsf{vk}_\mathsf{S}^t, \mathsf{vk}_\mathsf{R}^t, \sigma_\mathsf{R})$ as input, the algorithm immediately outputs 0 if

$$(\mathsf{vk}_\mathsf{S}^c, \mathsf{vk}_\mathsf{R}^c) \neq (\mathsf{vk}_\mathsf{S}^t, \mathsf{vk}_\mathsf{R}^t).$$

Only when

$$(\mathsf{vk}_\mathsf{S}^c, \mathsf{vk}_\mathsf{R}^c) = (\mathsf{vk}_\mathsf{S}^t, \mathsf{vk}_\mathsf{R}^t)$$

does the algorithm perform cryptographic operations: it first verifies σ_S and σ_R, and then executes the Test algorithm of the underlying PAEKS scheme on $(\mathsf{CT}', \mathsf{TD}')$. Since matching the verification keys $(\mathsf{vk}_\mathsf{S}^c, \mathsf{vk}_\mathsf{R}^c) \stackrel{?}{=} (\mathsf{vk}_\mathsf{S}^t, \mathsf{vk}_\mathsf{R}^t)$ can be performed using standard search techniques such as hash tables,

no cryptographic computation is required for sender-receiver matching. Consequently, the number of executions of the underlying PAEKS Test algorithm for a trapdoor generated for $(\mathsf{S}, \mathsf{R}, kw)$ depends only on the number of ciphertexts generated by sender S for receiver R. Therefore, for a trapdoor associated with a particular sender, the resulting search complexity, measured in terms of the number of cryptographic operations, is essentially the same as that of PEKS. Although a receiver still needs to generate a trapdoor for each sender, our transformation partially addresses the efficiency issue considered in [10, 14, 29, 30, 32, 39–41, 47].

7 Conclusion

In this paper, we define full CCTA security and propose a generic transformation that converts a PAEKS scheme into a CCTA-secure one. As a result, our transformation gives the first fully CCTA-secure PAEKS from pairings or lattices. For a trapdoor associated with a particular sender, the search complexity, measured in terms of the number of cryptographic operations, becomes independent of the number of senders, and the asymptotic search complexity is essentially the same as that of PEKS. From the PQC perspective, our generic construction provides a fully CCTA-secure lattice-based instantiation. Constructing fully CCTA-secure PQC instantiations based on assumptions other than lattices remains an interesting direction for future work.

Yoshida et al. [43] showed that matchmaking public key encryption (PK-ME), a public key variant of identity-based matchmaking encryption (IB-ME) [5], can be constructed from PAEKS. Their transformation assumes that the underlying PAEKS scheme provides anonymity (no sender and receiver information is leaked from ciphertexts), and they claimed that the Li-Boyen generic construction [31] provides anonymity since a NIKE shared key is indistinguishable from random. Obviously, our transformation does not preserve anonymity of the underlying PAEKS scheme since the sender's verification key and the receiver's verification key are explicitly contained in ciphertexts. As a result, their transformation cannot employ a CCTA-secure PAEKS scheme constructed via our transformation as its building block. Although reducing anonymity improves search efficiency, it is natural to consider applications that require anonymity. Proposing an anonymity preserving transformation that lifts up PAEKS to be CCTA-secure one may be an interesting topic and we leave it as future work.

Acknowledgment: This work was supported by JSPS KAKENHI Grant Number JP26K02908. This paper was edited for grammar using Copilot.

References

[1] Marius A. Aardal, Andrea Basso, Luca De Feo, Sikhar Patranabis, and Benjamin Wesolowski. A complete security proof of SQIsign. In *CRYPTO, Part VI*, pages 190–222, 2025.

[2] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. In *CRYPTO*, pages 205–222, 2005.

[3] Michel Abdalla, Mihir Bellare, and Gregory Neven. Robust encryption. In *Theory of Cryptography*, pages 480–497, 2010.

[4] Michel Abdalla, Mihir Bellare, and Gregory Neven. Robust encryption. *Journal of Cryptology*, 31(2):307–350, 2018.

[5] Giuseppe Ateniese, Danilo Francati, David Nuñez, and Daniele Venturi. Match me if you can: Matchmaking encryption and its applications. In *CRYPTO, Part II*, pages 701–731, 2019.

[6] Carsten Baum, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß, Christian Majenz, Shibam Mukherjee, Emmanuela Orsini, Sebastian Ramacher, Christian Rechberger, Lawrence Roy, and Peter Scholl. FAEST. tech. rep., national institute of standards and technology. `https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures`.

[7] Daniel J. Bernstein. Curve25519: New Diffie-Hellman speed records. In *Public Key Cryptography*, pages 207–228, 2006.

[8] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *EUROCRYPT*, pages 506–522, 2004.

[9] Dan Boneh, Emily Shen, and Brent Waters. Strongly unforgeable signatures based on computational Diffie-Hellman. In *Public Key Cryptography*, pages 229–240, 2006.

[10] Jiayin Cai, Xingwen Zhao, Dexin Li, Hui Li, and Kai Fan. Efficient and expressive public key authenticated encryption with keyword search in multi-user scenarios. *CoRR*, abs/2503.16828, 2025.

[11] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *EUROCRYPT*, pages 207–222, 2004.

[12] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, pages 523–552, 2010.

[13] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: an efficient post-quantum commutative group action. In *ASIACRYPT*, pages 395–427, 2018.

[14] Leixiao Cheng and Fei Meng. Server-aided public key authenticated searchable encryption with constant ciphertext and constant trapdoor. *IEEE Transactions on Information Forensics and Security*, 19:1388–1400, 2024.

[15] Gavin Cho, Georg Fuchsbauer, Adam O'Neill, and Marek Sefranek. Schnorr signatures are tightly secure in the ROM under a non-interactive assumption. In *CRYPTO, Part VI*, pages 223–255, 2025.

[16] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[17] Léo Ducas, Eamonn W. Postlethwaite, Ludo N. Pulles, and Wessel P. J. van Woerden. Hawk: Module LIP makes lattice signatures fast, compact and simple. In *ASIACRYPT, Part IV*, pages 65–94, 2022.

[18] Edward Eaton and Fang Song. Making existential-unforgeable signatures strongly unforgeable in the quantum random-oracle model. In *TQC*, LIPIcs, pages 147–162. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

[19] Keita Emura. Generic construction of public-key authenticated encryption with keyword search revisited: Stronger security and efficient construction. In *ACM ASIA Public-Key Cryptography Workshop*, pages 39–49, 2022.

[20] Keita Emura. Comments on "Lightweight multi-user public-key authenticated encryption with keyword search". *IEEE Transactions on Information Forensics and Security*, 21:5108–5109, 2026.

[21] Keita Emura. Security analysis on the multi-user keyword-field-free conjunctive searchable encryption scheme. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E109-A(8):1419–1420, 2026.

[22] Keita Emura and Kazumasa Omote. Security analysis on blockchain-based public key encryption with keyword search for medical data sharing in cloud environment. *Journal of Information Security and Applications*, 102:104610, 2026.

[23] Eduarda S. V. Freire, Dennis Hofheinz, Eike Kiltz, and Kenneth G. Paterson. Non-interactive key exchange. In *Public-Key Cryptography*, pages 254–271, 2013.

[24] Phillip Gajland, Bor de Kock, Miguel Quaresma, Giulio Malavolta, and Peter Schwabe. SWOOSH: efficient lattice-based non-interactive key exchange. In *USENIX Security*, 2024.

[25] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *ACM Symposium on Theory of Computing*, pages 197–206, 2008.

[26] Qiong Huang and Hongbo Li. An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. *Information Sciences*, 403:1–14, 2017.

[27] Eike Kiltz. Chosen-ciphertext security from tag-based encryption. In *Theory of Cryptography*, pages 581–600, 2006.

[28] Mukul Kulkarni and Keita Xagawa. Strong existential unforgeability and BUFF securities of MPC-in-the-head signatures. *Designs, Codes and Cryptography*, 94(7):159, 2026.

[29] Hongbo Li, Qiong Huang, Jianye Huang, and Willy Susilo. Public-key authenticated encryption with keyword search supporting constant trapdoor generation and fast search. *IEEE Transactions on Information Forensics and Security*, 18:396–410, 2023.

[30] Hongbo Li, Willy Susilo, Jian Shen, Chen Wang, Leixiao Cheng, and Qiong Huang. Proxy-free public-key authenticated updatable and searchable encryption for cloud storage. *IEEE Transactions on Dependable and Secure Computing*, 23(1):236–249, 2026.

[31] Qinyi Li and Xavier Boyen. Public-key authenticated encryption with keyword search made easy. *IACR Communications in Cryptology*, 1(2):16, 2024.

[32] Fucai Luo, Haiyan Wang, and Xingfu Yan. Re-PAEKS: Public-key authenticated re-encryption with keyword search. *IEEE Transactions on Mobile Computing*, 23(10):10077–10092, 2024.

[33] Sha Ma and Qiong Huang. A new framework of IND-CCA secure public key encryption with keyword search. *The Computer Journal*, 63(12):1849–1858, 2020.

[34] Riku Matsui and Keita Emura. Generic construction of public-key encryption with bidirectional keyword search. *IACR Cryptol. ePrint Arch.*, 2026:2945, 2026.

[35] Carlos Aguilar Melchor, Thibauld Feneuil, Nicolas Gama, Shay Gueron, James Howe, David Joseph, Antoine Joux, Edoardo Persichetti, Tovohery H. Randrianarisoa, Matthieu Rivain, and Dongze Yue. SDitH. tech. rep., national institute of standards and technology. `https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures`.

[36] Junling Pei and Fang-Wei Fu. MODRED: A code-based non-interactive key exchange protocol. *Theoretical Computer Science*, 1021:114943, 2024.

[37] Hirotomo Shinoki, Hisayoshi Sato, and Masayuki Yoshino. Fully secure searchable encryption from PRFs, pairings, and lattices. In *Applied Cryptography and Network Security, Part I*, pages 62–88, 2025.

[38] Isamu Teranishi, Takuro Oyama, and Wakaha Ogata. General conversion for obtaining strongly existentially unforgeable signatures. In *INDOCRYPT*, pages 191–205, 2006.

[39] Wei Wang, Dongli Liu, Zilin Zheng, Peng Xu, and Laurence Tianruo Yang. Identity-based group encryption with keyword search against keyword guessing attack. *IEEE Transactions on Information Forensics and Security*, 19:8023–8036, 2024.

[40] Yongliang Xu, Hang Cheng, Jiguo Li, Ximeng Liu, Xinpeng Zhang, and Meiqing Wang. Lightweight multi-user public-key authenticated encryption with keyword search. *IEEE Transactions on Information Forensics and Security*, 20:3234–3246, 2025.

[41] Wenjie Yang, Shujie Lin, Futai Zhang, and Anjia Yang. Dual-server public key authenticated searchable encryption with constant trapdoor against inside keyword guessing attacks. *IEEE Transactions on Information Forensics and Security*, 21:5794–5807, 2026.

[42] Lisha Yao, Jian Weng, Anjia Yang, Xiaojian Liang, Zhenghao Wu, Zike Jiang, and Lin Hou. Scalable CCA-secure public-key authenticated encryption with keyword search from ideal lattices in cloud computing. *Information Sciences*, 624:777–795, 2023.

[43] Takeshi Yoshida and Keita Emura. On the relations between matchmaking public key encryption and public key authenticated encryption with keyword search. In *Cryptology and Network Security*, pages 272–291, 2025.

[44] Takeshi Yoshida and Keita Emura. Comments on "Server-aided public key authenticated searchable encryption with constant ciphertext and constant trapdoor". *IEEE Transactions on Information Forensics and Security*, 21:9511–9512, 2026.

[45] Takeshi Yoshida and Keita Emura. Search-efficient public-key authenticated encryption with keyword search made easy. In *ProvSec*, pages 111–129, 2026.

[46] Takeshi Yoshida and Keita Emura. Security analysis on "Proxy-free public-key authenticated updatable and searchable encryption for cloud storage"'. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 2027, to appear. Available at `https://eprint.iacr.org/2026/1019`.

[47] Shengjie Zhang, Jianchang Lai, Liquan Chen, Jinguang Han, and Ge Wu. Keyword-field-free conjunctive searchable encryption for multiuser in EMR system. *IEEE Internet of Things Journal*, 12(15):30850–30861, 2025.

[48] Shiwen Zhang, Yibin Yang, Wei Liang, Junlong Zhou, Kuanching Li, and Rubén González Crespo. LFD-PEKS: A lightweight and fast dynamic public-key encryption with keyword search scheme for multi-user environments. *Computer Standards & Interfaces*, 95:104050, 2026.

A Appendix: Attacks against the LFD-PEKS scheme

In this appendix, we demonstrate that the LFD-PEKS scheme [48] is not secure. Concretely, keyword information is leaked from both ciphertexts and trapdoors. For breaking CI security, no oracle is employed. An adversary who observes the challenge ciphertext can break CI security. For breaking TI security, one trapdoor query corresponds to non-challenge keyword is employed. We clarify that we do not use the test oracle in both attacks. We also clarify that their encryption algorithm takes the data owner's secret key sk_O as input and considers KGA. Therefore, although the authors denote their scheme as PEKS, we regard it as a variant of PAEKS.

Let $\mathbb{G}$ and $\mathbb{G}_T$ be groups with prime order p, $g \in \mathbb{G}$ be a generator, and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a pairing. In the LFD-PEKS scheme, a data owner and a data user are defined, and let $\mathsf{pk}_O = g^{\mathsf{sk}_O}$ be a data owner's public key and pk_U be a data user's public key. A ciphertext consists of $C_1 = (\mathsf{pk}_U^{\mathsf{sk}_O})^s$, $C_2 = \mathsf{pk}_O^s = (g^{\mathsf{sk}_O})^s$, and $C_w = (H_2(kw)^{\mathsf{sk}_O})^s$ where $s \xleftarrow{\$} \mathbb{Z}_p$ and $H_2 : \{0,1\}^* \to \mathbb{G}$ is a collision resistant hash function. The CI adversary who observes public keys pk_O and pk_U declares two challenge keywords kw_0^* and kw_1^*, where $H_2(kw_0^*) \neq H_2(kw_1^*)$ holds, and sends them to the challenger. The challenger flips a coin $b \xleftarrow{\$} \{0,1\}$ and computes the challenge ciphertext corresponds to kw_b^*. Let $C_1^* = (\mathsf{pk}_U^{\mathsf{sk}_O})^{s^*}$, $C_2^* = \mathsf{pk}_O^{s^*} = (g^{\mathsf{sk}_O})^{s^*}$ and $C_w^* = (H_2(kw_b^*)^{\mathsf{sk}_O})^{s^*}$ be the challenge ciphertext. Here, $s^* \xleftarrow{\$} \mathbb{Z}_p$ is a random value chosen by the challenger. The adversary outputs $b = 0$ if $e(C_w^*, g) = e(H_2(kw_0^*), C_2^*)$ holds, and $b = 1$ if $e(C_w^*, g) = e(H_2(kw_1^*), C_2^*)$. This shows that the LFD-PEKS scheme is not CI secure. Since our attack employs only public values, our attack does not violate their security model.

A trapdoor consists of $T_w = H_2(kw)H_1(\mathsf{pk}_O^{\mathsf{sk}_U})$ (we omit other values). The TI adversary sends $kw \notin \{kw_0^*, kw_1^*\}$ to the challenger as a trapdoor query, and obtains T_w. Then, the adversary can extract $H_1(\mathsf{pk}_O^{\mathsf{sk}_U})$ by $T_w/H_2(kw)$. The adversary declares two challenge keywords kw_0^* and kw_1^*, where $H_2(kw_0^*) \neq H_2(kw_1^*)$ holds, and sends them to the challenger. The challenger flips a coin $b \xleftarrow{\$} \{0,1\}$ and computes the challenge trapdoor corresponds to kw_b^*, i..e, $T_w^* = H_2(kw_b^*)H_1(\mathsf{pk}_O^{\mathsf{sk}_U})$. The adversary outputs $b = 0$ if $T_{w_b^*}/(T_w/H_2(w)) = H_2(w_0^*)$ and $b = 1$ if $T_{w_b^*}/(T_w/H_2(w)) = H_2(w_1^*)$. This shows that the LFD-PEKS scheme is not TI secure. We remark that the trapdoor oracle is defined (Definition 6 in [48]) and our attack does not violate their security model.