Generic Construction of Public-key Encryption with Bidirectional Keyword Search

Riku Matsui[1] and Keita Emura*[1,2]

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

November 19, 2026

Abstract

Public-key Encryption with Bidirectional Keyword Search (PEBKS), proposed by Zhang, Qin, Dong, and Tian (Computer Standards & Interfaces, 2021), extends the functionality of Public-key Encryption with Keyword Search (PEKS) by allowing not only the receiver but also the sender to generate trapdoors for keyword search. In this paper, we identify a close relationship between PEBKS and Public-key Authenticated Encryption with Keyword Search (PAEKS). Leveraging the generic construction of PAEKS proposed by Li and Boyen (IACR Communications in Cryptology, 2024), which is based on Non-Interactive Key Exchange (NIKE) and symmetric-key Equality Predicate Encryption (EPE), we show that PEBKS can also be generically constructed from the same building blocks. While Zhang et al. proposed a concrete pairing-based PEBKS scheme, our generic construction yields PEBKS instantiations under a wide range of cryptographic assumptions, including pairing-free Diffie-Hellman assumptions, lattices, isogenies, and codes. Furthermore, we define Key Collision Resistance for EPE as a new security property required for proving Consistency of PEBKS. We prove that the Li-Boyen EPE construction based on a pseudorandom function (PRF) achieves this property if the underlying PRF is Key Collision Resistant. We also discuss the use of AES and HMAC as the underlying PRF in light of the key collision attack reported by Taiyama et al. (Asiacrypt 2024). Since Key Collision Resistance is likewise required for proving Consistency of PAEKS, our result complements the Li-Boyen generic construction of PAEKS and is of independent interest.

1 Introduction

1.1 Background

In Public-key Encryption with Keyword Search (PEKS) [1], a sender encrypts a keyword using the receiver's public key, while the receiver generates a trapdoor (search query) using its secret key. The trapdoor is then used to search for encrypted keywords. A typical scenario assumes that the sender uploads ciphertexts of keywords to a server, while the receiver uploads the corresponding trapdoors, enabling the server to perform keyword searches on behalf of the receiver. As a concrete application, encrypted e-mail search was introduced in [1] where a sender sends encrypted e-mails to a receiver and the receiver wishes to route messages differently depending on their contents. For example, by generating a trapdoor corresponding to the keyword "Urgent", an e-mail gateway

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

can distinguish urgent messages from non-urgent ones and forward them to different destinations, while preserving the confidentiality of keywords and the e-mail contents. Specifically, the email content is encrypted using a Public-Key Encryption (PKE) scheme under the receiver's public key, while the associated keywords are encrypted using PEKS. The resulting PKE ciphertext and PEKS ciphertext are sent to the email gateway. When a search query matches a keyword via a PEKS trapdoor, the email gateway forwards the corresponding PKE ciphertext to the receiver.

Public-key Encryption with Bidirectional Keyword Search (PEBKS). PEKS assumes that only the receiver generates trapdoors. This is a natural assumption, since the sender specifies the intended receiver by using the receiver's public key when encrypting a keyword. It should be noted, however, that trapdoor generation requires the receiver's secret key. Consequently, even the sender who generated the ciphertext cannot perform keyword searches on that ciphertext. To address this limitation, Zhang et al. [16] proposed Public-key Encryption with Bidirectional Keyword Search (PEBKS). As a motivating application, Zhang et al. considered encrypted e-mail search, which was originally introduced as an application of PEKS. In this setting, PEBKS additionally allows a sender to search encrypted e-mails that the sender previously sent. Concretely, Zhang et al. wrote that

> "*A user may search an email from the inbox and the outbox to confirm whether an important email has been replied. Therefore, an encrypted email system needs a secure way to search across the received emails and the sent emails.*"

Note that even if PEBKS enables search by the sender, the above email application still delivers to the sender only an email ciphertext encrypted under the receiver's public key. As a result, the sender cannot recover the email content. Although Zhang et al. did not discuss this issue, it can be naturally resolved by combining PEBKS with a non-interactive key exchange (NIKE) scheme. More precisely, the email content can be encrypted using a symmetric key derived from a NIKE shared secret established between the sender and the receiver.

1.2 Our Motivation

Zhang et al. [16] proposed a pairing-based PEBKS scheme (in the random oracle model), which is not post-quantum secure. Since encrypted e-mails are often used to exchange sensitive personal and business information and may remain confidential for many years, it is natural to consider them as potential targets of "Harvest Now, Decrypt Later" attacks. In such an attack, an adversary stores encrypted e-mails today and decrypts them in the future once the underlying hardness assumptions are broken. Therefore, post-quantum security is a particularly important consideration in the encrypted e-mail search application of PEBKS. One possible approach is to design a concrete scheme based on post-quantum assumptions such as lattices. Another approach is to develop a generic construction that can be instantiated using post-quantum secure cryptographic primitives. Moreover, given the possibility that a particular complexity assumption may be broken in the future, it is preferable to avoid constructing a cryptographic scheme whose security relies solely on complexity assumptions arising from a common mathematical foundation. Motivated by this observation, our goal in this paper is to propose a generic construction of PEBKS that admits instantiations under a wide range of cryptographic assumptions.

Table 1: PEBKS schemes. BDH: Bilinear Diffie-Hellman, DDH: Decisional Diffie-Hellman, DBDH: Decisional Bilinear Diffie-Hellman, MLWE: Module Learning with Errors, RSD: Restricted Syndrome Decoding, CSIDE: Commutative Supersingular Isogeny Diffie-Hellman, (Q)ROM: (Quantum) Random Oracle Model, STD: Standard Model

Scheme/Instantiations	NIKE	EPE	Assumption	Note
Zhang-Qin-Dong-Tian: Pairing-based PEBKS [16]	-	-	BDH	ROM
Ours: Pairing-free PEBKS	Diffie-Hellman [3]	PRF [11]	DDH	ROM
Ours: Pairing-based PEBKS	Pairing-based [7]	PRF [11]	DBDH	STD
Ours: Lattice-based PEBKS	Swoosh [8]	PRF [11]	MLWE	QROM, Post-Quantum
Ours: Code-based PEBKS	MODRED [13]	PRF [11]	RSD	ROM, Post-Quantum
Ours: Isogeny-based PEBKS	CSIDE [2]	PRF [11]	CSIDH	STD, Post-Quantum

1.3 Our Contribution

Our contributions are twofold.

- We identify a close relationship between PEBKS and Public-key Authenticated Encryption with Keyword Search (PAEKS) [10, 11]. Leveraging the generic construction of PAEKS proposed by Li and Boyen [11], which is based on NIKE and symmetric-key Equality Predicate Encryption (EPE), we show that PEBKS can also be generically constructed from the same building blocks.
  - While Zhang et al. [16] proposed a concrete pairing-based PEBKS scheme, our generic construction yields PEBKS instantiations under a wide range of cryptographic assumptions, including pairing-free Diffie-Hellman and pairing-based construction in the standard model[1], and post-quantum instantiations based on lattices, isogenies, and codes (Table 1).
- We define Key Collision Resistance for EPE as a new security property required for proving Consistency of PEBKS. We prove that the Li-Boyen EPE construction based on a pseudorandom function (PRF) achieves this property if the underlying PRF is Key Collision Resistant.
  - We also discuss the use of AES and HMAC as the underlying PRF in light of the key collision attack reported by Taiyama et al. [15].
  - Since Key Collision Resistance is likewise required for proving Consistency of PAEKS, our result complements the Li-Boyen generic construction of PAEKS. Furthermore, our analysis readily extends to pairing-based EPE [14] and lattice-based EPE [11]. Generic constructions that employ PAEKS as a building block (e.g., [4, 5, 12]) inherit their broad instantiability from the Li-Boyen framework. Since our analysis requires no modification to the underlying EPE, it does not affect these instantiation results.

[1]The Zhang et al. scheme is constructed over symmetric pairings whereas asymmetric pairings are suitable for providing an efficient implementation [9] due to the difference of embedded degree. Our pairing-based instantiation is constructed over asymmetric pairings. Besides avoiding random oracles, we argue that the use of asymmetric pairings is also an important practical advantage. Although this improvement is relatively minor from a theoretical standpoint, it can have a significant impact on implementation efficiency and deployment. For example, it has been reported that "pairing operations in symmetric settings are more than ten times slower than those in asymmetric settings, and scalar multiplication is 40-60 times slower than in asymmetric settings."in [6].

Technical Overview. We observe that

- In the syntax of PEBKS, a sender uses its own secret key (in addition to the receiver's public key) to encrypt a keyword, which is the same as that of PAEKS.
- The same shared key is obtained regardless of which of the two parties' secret keys is used due to the symmetry of NIKE.

In the Li-Boyen generic PAEKS construction based on NIKE, a NIKE key is (non-interactively) shared between the sender and the receiver. The sender uses the shared key to encrypt a keyword under EPE, thereby generating a PAEKS ciphertext. The receiver uses the same shared key to generate an EPE query key for the keyword, which serves as a PAEKS trapdoor. Therefore, it follows that a sender possessing the NIKE shared key can also generate an EPE query key. Due to the symmetry of NIKE, our PEBKS construction cannot distinguish a trapdoor generated by the sender from a trapdoor generated by the receiver. It should be noted, however, that while PEBKS requires both sender-generated and receiver-generated trapdoors to enable keyword search, it does not require their origins to be indistinguishable. We emphasize that PAEKS does not inherently provide trapdoor generation for the sender. Hence, our result should not be interpreted as a generic transformation from PAEKS to PEBKS. Instead, it is based solely on the fact that the Li-Boyen generic construction of PAEKS employs NIKE, from which both the sender and the receiver can derive the same shared secret.

For any keywords w, w', given a ciphertext $\mathsf{PEBKS.C}_w$ and a trapdoor $\mathsf{PEBKS.T}_{w'}$, Consistency of PEBKS guarantees that the test algorithm outputs 0 whenever $(\mathsf{S}, \mathsf{R}, w) \neq (\mathsf{S}', \mathsf{R}', w')$, where S and S' denote senders, and R and R' denote receivers. Without Consistency, one can construct a secure yet practically meaningless scheme by generating ciphertexts and trapdoors independently at random and defining the test algorithm to always output 1. Therefore, Consistency is an important security requirement in searchable encryption. Since a NIKE key is used as an EPE key and a keyword is encrypted under that key using EPE, the EPE scheme is required to provide Consistency. Nevertheless, no formal definition of Consistency for EPE was provided [11]. Although Shinoki et al. [14] defined Consistency for EPE (Definition 7 of [14]), their definition only considers the case where $w \neq w'$ under the same EPE key. It is necessary to consider evaluations under different EPE keys when $(\mathsf{S}, \mathsf{R}) \neq (\mathsf{S}', \mathsf{R}')$. We therefore introduce the notion of Key Collision Resistance for EPE. In essence, Key Collision Resistance reduces to the requirement that the probability of obtaining the same PRF output under different keys is negligible. For AES, we rely on the key collision attack against AES reported by Taiyama et al. [15]. In the case of HMAC, a keyed hashing component $H(K||m)$, where K is the key and m is the message, naturally supports Key Collision Resistance when H is a collision resistant hash function. Moreover, both the pairing-based EPE [14] and the lattice-based EPE [11] first apply a keyed PRF to transform the keyword. Consequently, their Key Collision Resistance follows from that of the underlying PRF.

2 Preliminary

2.1 Non-Interactive Key Exchange

A NIKE scheme NIKE consists of the following three algorithms ($\mathsf{NIKE.Setup}, \mathsf{NIKE.KeyGen}, \mathsf{SharedKey}$).

$\mathsf{NIKE.Setup}(\lambda)$**:** It takes a security parameter $\lambda \in \mathbb{N}$ as input, and outputs public parameters $\mathsf{NIKE.param}$. We assume that $\mathsf{NIKE.param}$ specifies an identity space $\mathcal{I}$ and a shared key space $\mathcal{K}$.

$\mathsf{NIKE.KeyGen}(\mathsf{ID})$**:** It takes an identifier $\mathsf{ID} \in \mathcal{I}$ as input, and outputs a public key and secret key pair $(\mathsf{NIKE.pk}, \mathsf{NIKE.sk})$.

$\mathsf{SharedKey}(\mathsf{ID}_1, \mathsf{pk}_1, \mathsf{ID}_2, \mathsf{sk}_2)$**:** It takes the identifier $\mathsf{ID}_1 \in \mathcal{I}$ and public key pk_1 of the other party, together with its own identifier $\mathsf{ID}_2 \in \mathcal{I}$ and secret key sk_2, as input, and outputs a shared key $\mathsf{K} \in \mathcal{K}$ or an error symbol $\perp$.

Next, we define the correctness. Correctness guarantees that if two users generate their key pairs using $\mathsf{NIKE.KeyGen}$ and each of them runs $\mathsf{SharedKey}$, they obtain the same shared key with overwhelming probability.

Definition 1 (**Correctness**). *A NIKE scheme* NIKE *is said to be Correct if for all* $\lambda \in \mathbb{N}$, $\mathsf{NIKE.param} \leftarrow \mathsf{NIKE.Setup}(\lambda)$, $\mathsf{ID}_1, \mathsf{ID}_2 \in \mathcal{I}$, $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{NIKE.KeyGen}(\mathsf{ID}_1)$, $(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow \mathsf{NIKE.KeyGen}(\mathsf{ID}_2)$, $\mathsf{K} \leftarrow \mathsf{SharedKey}(\mathsf{ID}_1, \mathsf{pk}_1, \mathsf{ID}_2, \mathsf{sk}_2)$, $\mathsf{K}' \leftarrow \mathsf{SharedKey}(\mathsf{ID}_2, \mathsf{pk}_2, \mathsf{ID}_1, \mathsf{sk}_1)$, *it holds that*

$$\Pr[\mathsf{K} = \mathsf{K}'] \geq 1 - \mathsf{negl}(\lambda)$$

Next, we define HKR-CKS security [7] for NIKE, where HKR stands for Honest-Key-Registration and CKS stands for Cash-Kiltz-Shoup. In the HKR-CKS model, the adversary submits an identifier ID to a registration oracle, which generates a key pair by running $\mathsf{NIKE.KeyGen}$ and returns the public key to the adversary. In other words, the adversary is not allowed to register pubic keys of its own choice. A stronger notion, called DKR-CKS security, has also been defined, where DKR stands for Dishonest Key Registration. In the DKR-CKS model, the adversary submits a pair $(\mathsf{ID}, \mathsf{pk})$ to the registration oracle and is thus allowed to use a public key pk of its own choice. The generic construction of PAEKS by Li-Boyen requires the underlying NIKE to be HKR-CKS secure.

HKR-CKS Security. Let $\mathcal{C}$ be a challenger and $\mathcal{A}$ be an adversary. We define the experiment $\mathsf{Exp}^{\mathsf{hkr\text{-}cks}}_{\mathsf{NIKE},\mathcal{A}}(\lambda)$ as follows.

Setup. $\mathcal{C}$ runs $\mathsf{NIKE.Setup}(\lambda)$ to generate public parameters param, initializes an empty list $\mathcal{L}$, and gives param to $\mathcal{A}$.

Phase1. $\mathcal{A}$ can issue a polynomial number of queries to the following oracle.

- Honest registration oracle $\mathsf{RegHon}(\mathsf{ID})$: $\mathcal{A}$ submits an identifier ID to the oracle. $\mathcal{C}$ runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{NIKE.KeyGen}(\mathsf{ID})$, adds $(\mathit{honest}, \mathsf{ID}, \mathsf{sk}, \mathsf{pk})$ to $\mathcal{L}$, and returns pk to $\mathcal{A}$.

Challenge. $\mathcal{A}$ submits two identifiers ID^*_1 and ID^*_2 to $\mathcal{C}$, where it is required that $\mathsf{ID}^*_1 \neq \mathsf{ID}^*_2$ and that both identifiers are registered as honest in $\mathcal{L}$. Let pk^*_1 and sk^*_2 denote the public key of ID^*_1 and the secret key of ID^*_2 recorded in $\mathcal{L}$, respectively. $\mathcal{C}$ chooses a random bit $\mu \in \{0, 1\}$.

- If $\mu = 0$, $\mathcal{C}$ computes the shared key $\mathsf{K}^*_0 \leftarrow \mathsf{SharedKey}(\mathsf{ID}^*_1, \mathsf{pk}^*_1, \mathsf{ID}^*_2, \mathsf{sk}^*_2)$.
- If $\mu = 1$, $\mathcal{C}$ chooses a random key $\mathsf{K}^*_1 \in \mathcal{K}$.

$\mathcal{C}$ returns the challenge key K^*_μ to $\mathcal{A}$.

Guess. $\mathcal{A}$ outputs a guess bit $\mu' \in \{0, 1\}$. The experiment outputs 1 if $\mu' = \mu$ (in which case we say that $\mathcal{A}$ wins), and 0 otherwise.

Definition 2. *A NIKE scheme* NIKE *is said to be HKR-CKS secure if for all PPT (probabilistic polynomial-time) adversaries* $\mathcal{A}$*, the advantage*

$$\mathsf{Adv}^{\mathsf{hkr\text{-}cks}}_{\mathsf{NIKE},\mathcal{A}}(\lambda) := \left| \Pr[\mathsf{Exp}^{\mathsf{hkr\text{-}cks}}_{\mathsf{NIKE},\mathcal{A}}(\lambda) = 1] - \frac{1}{2} \right|$$

is negligible in λ.

2.2 Symmetric-key Equality Predicate Encryption

An EPE scheme EPE consists of the following three algorithms $(\mathsf{EPE.KG}, \mathsf{EPE.Enc}, \mathsf{EPE.Qry})$. Let $\mathcal{K}$ be the key space. Since our construction uses a shared key of NIKE as a key of EPE, we identify the key space of EPE with the shared key space of NIKE throughout this paper.

$\mathsf{EPE.KG}(\mathsf{K}, w)$**:** It takes a secret key $\mathsf{K} \in \mathcal{K}$ and a keyword w as input, and outputs a query key dk_w.

$\mathsf{EPE.Enc}(\mathsf{K}, w)$**:** It takes a secret key $\mathsf{K} \in \mathcal{K}$ and a keyword w as input, and outputs a ciphertext σ_w.

$\mathsf{EPE.Qry}(\mathsf{dk}_w, \sigma_w)$**:** It takes a query key dk_w and a ciphertext σ_w, and outputs 1 or 0.

Next, we define correctness. Correctness guarantees that if a query key and a ciphertext are generated with the same key K for the same keyword w, then $\mathsf{EPE.Qry}$ outputs 1 with overwhelming probability.

Definition 3 (**Correctness**)**.** *An EPE scheme* EPE *is said to be Correct if for all* $\mathsf{K} \in \mathcal{K}$ *and all keywords* w, w' *such that* $w = w'$*, it holds that* $\Pr[\mathsf{EPE.Qry}(\mathsf{EPE.KG}(\mathsf{K}, w'), \mathsf{EPE.Enc}(\mathsf{K}, w)) = 1] \geq 1 - \mathsf{negl}(\lambda)$.

Consistency of EPE. Next, we define Consistency. Consistency guarantees that for a query key dk_w and a ciphertext σ_w generated with the same key K, $\mathsf{EPE.Qry}$ outputs 0 with overwhelming probability if $w \neq w'$. Although Li-Boyen did not provide a definition of Consistency, Shinoki et al. [14] defined it, and we follow their definition in this paper.

Let $\mathcal{C}$ be a challenger and $\mathcal{A}$ be an adversary. We define the experiment $\mathsf{Exp}^{\mathsf{cons}}_{\mathsf{EPE},\mathcal{A}}$ as follows.

Setup. $\mathcal{C}$ chooses a key $\mathsf{K} \leftarrow \mathcal{K}$ uniformly at random from the key space $\mathcal{K}$.

Challenge. $\mathcal{A}$ outputs two distinct keywords w and w' $(w \neq w')$. $\mathcal{C}$ runs $\sigma_w \leftarrow \mathsf{EPE.Enc}(\mathsf{K}, w)$ and $\mathsf{dk}_{w'} \leftarrow \mathsf{EPE.KG}(\mathsf{K}, w')$. The experiment outputs 1 if $\mathsf{EPE.Qry}(\mathsf{dk}_{w'}, \sigma_w) = 1$, and 0 otherwise.

Definition 4 (**Consistency**)**.** *An EPE scheme* EPE *is said to be Consistent if for all PPT adversaries* $\mathcal{A}$*,* $\Pr\left[\mathsf{Exp}^{\mathsf{cons}}_{\mathsf{EPE},\mathcal{A}}(\lambda) = 1\right]$ *is negligible in* λ*.*

Next, we define ciphertext indistinguishability (CT-IND) of EPE.

CT-IND Security. Let $\mathcal{C}$ be a challenger and $\mathcal{A}$ be an adversary. We define the experiment $\mathsf{Exp}^{\mathsf{ct\text{-}ind}}_{\mathsf{EPE},\mathcal{A}}$ as follows.

Setup. $\mathcal{C}$ chooses a key $\mathsf{K} \leftarrow \mathcal{K}$ uniformly at random from the key space $\mathcal{K}$.

Phase 1. $\mathcal{A}$ can issue a polynomial number of queries to the following two oracles.

- Encryption oracle $E_{\mathsf{K}}(w)$: $\mathcal{A}$ submits an arbitrary keyword w. $\mathcal{C}$ runs $\sigma_w \leftarrow \mathsf{EPE.Enc}(\mathsf{K}, w)$ and returns the ciphertext σ_w to $\mathcal{A}$.
- Query key generation oracle $K_{\mathsf{K}}(w)$: $\mathcal{A}$ submits an arbitrary keyword w. $\mathcal{C}$ runs $\mathsf{dk}_w \leftarrow \mathsf{EPE.KG}(\mathsf{K}, w)$ and returns the query key dk_w to $\mathcal{A}$.

Challenge. $\mathcal{A}$ submits two keywords w_0^* and w_1^* to $\mathcal{C}$, where neither w_0^* nor w_1^* has been queried to the query key generation oracle $K_{\mathsf{K}}(w)$. $\mathcal{C}$ chooses a random bit $\mu \leftarrow \{0, 1\}$, runs $\sigma^*_{w^*_\mu} \leftarrow \mathsf{EPE.Enc}(\mathsf{K}, w^*_\mu)$, and returns the challenge ciphertext $\sigma^*_{w^*_\mu}$ to $\mathcal{A}$.

Phase 2. $\mathcal{A}$ can issue a polynomial number of queries to the two oracles as in Phase 1, except that $\mathcal{A}$ is not allowed to query w_0^* or w_1^* to the query key generation oracle $K_{\mathsf{K}}(w)$. There is no restriction on queries to the encryption oracle $E_{\mathsf{K}}(w)$.

Guess. $\mathcal{A}$ outputs a guessing bit $\mu' \in \{0,1\}$. The experiment outputs 1 if $\mu' = \mu$ (in which case we say that $\mathcal{A}$ wins), and 0 otherwise.

Definition 5 (**Ciphertext Indistinguishability**). *An EPE scheme* EPE *is said to be CT-IND secure if for all PPT adversaries $\mathcal{A}$, the advantage*

$$\mathsf{Adv}^{\mathsf{ct\text{-}ind}}_{\mathsf{EPE},\mathcal{A}}(\lambda) := \left| \Pr[\mathsf{Exp}^{\mathsf{ct\text{-}ind}}_{\mathsf{EPE},\mathcal{A}}(\lambda) = 1] - \frac{1}{2} \right|$$

is negligible in λ.

3 Public-key Encryption with Bidirectional Keywords Search

In this section, we give the definition of PEBKS. As a modification in this paper, we add ID_{R} and ID_{S} to the inputs of the $\mathsf{PEBKS.KeyGen}_{\mathsf{R}}$ and $\mathsf{PEBKS.KeyGen}_{\mathsf{S}}$ algorithms, respectively. Zhang et al. distinguished a receiver's trapdoor from a sender's trapdoor by denoting them as $\mathsf{PEBKS.T}_w^{(\mathsf{R})}$ and $\mathsf{PEBKS.T}_w^{(\mathsf{S})}$, respectively.[2] In our proposed generic construction, however, it is impossible to distinguish whether a trapdoor was generated by the receiver or by the sender, since both of them derive the same NIKE shared key. Therefore, in this paper, we denote a trapdoor by $\mathsf{PEBKS.T}_w$ regardless of which party generated it. Note that we do not rule out the possibility that some applications of PEBKS need to distinguish between the two types of trapdoors.

A PEBKS scheme PEBKS consists of the following seven algorithms ($\mathsf{PEBKS.Setup}$, $\mathsf{PEBKS.KeyGen}_{\mathsf{R}}$, $\mathsf{PEBKS.KeyGen}_{\mathsf{S}}$, $\mathsf{PEBKS.KWEncrypt}$, $\mathsf{PEBKS.Trapdoor}_{\mathsf{R}}$, $\mathsf{PEBKS.Trapdoor}_{\mathsf{S}}$, $\mathsf{PEBKS.Test}$).

$\mathsf{PEBKS.Setup}(\lambda)$**:** It takes a security parameter $\lambda \in \mathbb{N}$ as input , and outputs public parameters param. We assume that param includes a keyword space $\mathcal{KW}$.

$\mathsf{PEBKS.KeyGen}_{\mathsf{R}}(\mathsf{param}, \mathsf{ID}_{\mathsf{R}})$**:** The receiver key generation algorithm takes param and ID_{R} as input, and outputs a public key $\mathsf{PEBKS.pk}_{\mathsf{R}}$ and a secret key $\mathsf{PEBKS.sk}_{\mathsf{R}}$.

$\mathsf{PEBKS.KeyGen}_{\mathsf{S}}(\mathsf{param}, \mathsf{ID}_{\mathsf{S}})$**:** The sender key generation algorithm takes param and ID_{S} as input, and outputs a public key $\mathsf{PEBKS.pk}_{\mathsf{S}}$ and a secret key $\mathsf{PEBKS.sk}_{\mathsf{S}}$.

$\mathsf{PEBKS.KWEncrypt}(\mathsf{PEBKS.sk}_{\mathsf{S}}, \mathsf{PEBKS.pk}_{\mathsf{R}}, w)$**:** The keyword encryption algorithm takes $\mathsf{PEBKS.sk}_{\mathsf{S}}$, $\mathsf{PEBKS.pk}_{\mathsf{R}}$, and a keyword $w \in \mathcal{KW}$ as input, and outputs a ciphertext $\mathsf{PEBKS.C}_w$.

$\mathsf{PEBKS.Trapdoor}_{\mathsf{R}}(\mathsf{PEBKS.sk}_{\mathsf{R}}, \mathsf{PEBKS.pk}_{\mathsf{S}}, w)$**:** The receiver trapdoor generation algorithm takes $\mathsf{PEBKS.sk}_{\mathsf{R}}$, $\mathsf{PEBKS.pk}_{\mathsf{S}}$, and a keyword $w \in \mathcal{KW}$ as input, and outputs a trapdoor $\mathsf{PEBKS.T}_w$.

$\mathsf{PEBKS.Trapdoor}_{\mathsf{S}}(\mathsf{PEBKS.sk}_{\mathsf{S}}, \mathsf{PEBKS.pk}_{\mathsf{R}}, w)$**:** The sender trapdoor generation algorithm takes $\mathsf{PEBKS.sk}_{\mathsf{S}}$, $\mathsf{PEBKS.pk}_{\mathsf{R}}$, and a keyword $w \in \mathcal{KW}$ as input, and outputs a trapdoor $\mathsf{PEBKS.T}_w$.

$\mathsf{PEBKS.Test}(\mathsf{PEBKS.T}_w, \mathsf{PEBKS.C}_w)$**:** The test algorithm takes $\mathsf{PEBKS.T}_w$ and $\mathsf{PEBKS.C}_w$ as input, and outputs 1 or 0.

[2] Although Zhang et al. distinguish $\mathsf{PEBKS.T}_w^{(\mathsf{S})}$ and $\mathsf{PEBKS.T}_w^{(\mathsf{R})}$ in their syntax, they are identical in their actual construction.

Next, we define correctness. Correctness guarantees that for any keyword $w \in \mathcal{KW}$, given a ciphertext $\mathsf{PEBKS.C}_w$ and a trapdoor $\mathsf{PEBKS.T}_w$ generated for w, $\mathsf{PEBKS.Test}$ returns 1.

Definition 6 (**Correctness**). *A PEBKS scheme is said to be Correct if for all* $\lambda \in \mathbb{N}$, *all* $\mathsf{param} \leftarrow \mathsf{PEBKS.Setup}(\lambda)$, *all* $(\mathsf{PEBKS.pk_R}, \mathsf{PEBKS.sk_R}) \leftarrow \mathsf{PEBKS.KeyGen_R}(\mathsf{param}, \mathsf{ID_R})$, *all* $(\mathsf{PEBKS.pk_S}, \mathsf{PEBKS.sk_S}) \leftarrow \mathsf{PEBKS.KeyGen_S}(\mathsf{param}, \mathsf{ID_S})$, *all* $w \in \mathcal{KW}$, *and all* $\mathsf{PEBKS.C}_w \leftarrow \mathsf{PEBKS.KWEncrypt}(\mathsf{PEBKS.sk_S}, \mathsf{PEBKS.pk_R}, w)$, *it holds with overwhelming probability that* $\mathsf{PEBKS.Test}(\mathsf{PEBKS.Trapdoor_R}(\mathsf{sk}, \mathsf{pk}, w), \mathsf{PEBKS.C}_w) = \mathsf{PEBKS.Test}(\mathsf{PEBKS.Trapdoor_S}(\mathsf{sk}, \mathsf{pk}, w), \mathsf{PEBKS.C}_w) = 1$.

Since Zhang et al. [16] did not define Consistency, we newly define it in this paper. Consistency guarantees that for any keywords $w, w' \in \mathcal{KW}$, given a ciphertext $\mathsf{PEBKS.C}_w$ and a trapdoor $\mathsf{PEBKS.T}_{w'}$, $\mathsf{PEBKS.Test}$ returns 0 if $(\mathsf{S}, \mathsf{R}, w) \neq (\mathsf{S}', \mathsf{R}', w')$. In the following, $\mathsf{PEBKS.T}_{w'}$ is generated by the $\mathsf{PEBKS.Trapdoor_S}$ algorithm, although it could also be generated by the $\mathsf{PEBKS.Trapdoor_R}$ algorithm. As mentioned above, in this paper we do not distinguish between trapdoors generated by the receiver and those generated by the sender. Therefore, without loss of generality, we generate $\mathsf{PEBKS.T}_{w'}$ by using the $\mathsf{PEBKS.Trapdoor_S}$ algorithm. Let $\mathcal{C}$ be a challenger and $\mathcal{A}$ be an adversary.

Definition 7 (**Consistency**). $\mathcal{C}$ *runs* $\mathsf{param} \leftarrow \mathsf{PEBKS.Setup}(\lambda)$, $(\mathsf{PEBKS.pk_{S_0}}, \mathsf{PEBKS.sk_{S_0}}) \leftarrow \mathsf{PEBKS.KeyGen_S}(\mathsf{param}, \mathsf{ID_{S_0}})$, $(\mathsf{PEBKS.pk_{S_1}}, \mathsf{PEBKS.sk_{S_1}}) \leftarrow \mathsf{PEBKS.KeyGen_S}(\mathsf{param}, \mathsf{ID_{S_1}})$, $(\mathsf{PEBKS.pk_{R_0}}, \mathsf{PEBKS.sk_{R_0}}) \leftarrow \mathsf{PEBKS.KeyGen_R}(\mathsf{param}, \mathsf{ID_{R_0}})$, *and* $(\mathsf{PEBKS.pk_{R_1}}, \mathsf{PEBKS.sk_{R_1}}) \leftarrow \mathsf{PEBKS.KeyGen_R}(\mathsf{param}, \mathsf{ID_{R_1}})$, *and gives* $(\mathsf{param}, \mathsf{PEBKS.pk_{S_0}}, \mathsf{PEBKS.pk_{S_1}}, \mathsf{PEBKS.pk_{R_0}}, \mathsf{PEBKS.pk_{R_1}})$ *to* $\mathcal{A}$. $\mathcal{A}$ *outputs* $(i, j, w, i', j', w') \in (\{0,1\}^2 \times \mathcal{KW})^2$. $\mathcal{C}$ *computes* $\mathsf{PEBKS.C}_w \leftarrow \mathsf{PEBKS.KWEncrypt}(\mathsf{PEBKS.sk_{S_i}}, \mathsf{PEBKS.pk_{R_j}}, w)$ *and* $\mathsf{PEBKS.T}_{w'} \leftarrow \mathsf{PEBKS.Trapdoor_S}(\mathsf{PEBKS.sk_{S_{i'}}}, \mathsf{PEBKS.pk_{R_{j'}}}, w')$. *We say that* $\mathcal{A}$ *wins if* $\mathsf{PEBKS.Test}(\mathsf{PEBKS.T}_{w'}, \mathsf{PEBKS.C}_w) = 1$ *and* $(\mathsf{S}_i, \mathsf{R}_j, w) \neq (\mathsf{S}_{i'}, \mathsf{R}_{j'}, w')$. *A PEBKS scheme* PEBKS *is said to be Consistent if for all PPT adversaries* $\mathcal{A}$, *the advantage* $\mathsf{Adv}^{\mathsf{cons}}_{\mathsf{PEBKS},\mathcal{A}}(\lambda) := \Pr[\mathcal{A}\ \textit{wins}]$ *is negligible in* λ.

IND-CKA Security. As a security notion for PEBKS schemes, we define ciphertext indistinguishability against chosen keyword attacks (IND-CKA). This notion guarantees that a ciphertext $\mathsf{PEBKS.C}_w$ generated by $\mathsf{PEBKS.KWEncrypt}(\mathsf{PEBKS.sk_S}, \mathsf{PEBKS.pk_R}, w)$ leaks no information about the keyword w unless the adversary $\mathcal{A}$ obtains a trapdoor $\mathsf{PEBKS.T}_w$. Let $\mathcal{C}$ be a challenger and $\mathcal{A}$ be an adversary.

Setup. First, $\mathcal{C}$ runs $\mathsf{PEBKS.Setup}(\lambda)$ to generate param and initializes empty lists L_S and L_R. Next, $\mathcal{C}$ runs $\mathsf{PEBKS.KeyGen_R}(\mathsf{param}, \mathsf{ID_{R^*}})$ to obtain a receiver's key pair $(\mathsf{PEBKS.pk_{R^*}}, \mathsf{PEBKS.sk_{R^*}})$, and adds $(\mathsf{ID_{R^*}}, \mathsf{PEBKS.pk_{R^*}}, \mathsf{PEBKS.sk_{R^*}})$ to L_R. $\mathcal{C}$ also runs $\mathsf{PEBKS.KeyGen_S}(\mathsf{param}, \mathsf{ID_{S^*}})$ to obtain a sender's key pair $(\mathsf{PEBKS.pk_{S^*}}, \mathsf{PEBKS.sk_{S^*}})$, and adds $(\mathsf{ID_{S^*}}, \mathsf{PEBKS.pk_{S^*}}, \mathsf{PEBKS.sk_{S^*}})$ to L_S. Then, $\mathcal{C}$ sends param, $\mathsf{PEBKS.pk_{R^*}}$, and $\mathsf{PEBKS.pk_{S^*}}$ to $\mathcal{A}$, and keeps $\mathsf{PEBKS.sk_{R^*}}$ and $\mathsf{PEBKS.sk_{S^*}}$ secret. We assume that $\mathcal{A}$ can observe identifiers and public keys if they are stored on either L_S or L_R.

Phase 1. $\mathcal{A}$ can issue a polynomial number of queries to the following three oracles.

- **Receiver Trapdoor Generation Oracle**: On input $(\mathsf{ID_S}, w)$ from $\mathcal{A}$, $\mathcal{C}$ searches the list L_S. If $\mathsf{ID_S}$ has not been registered, $\mathcal{C}$ runs $\mathsf{PEBKS.KeyGen_S}(\mathsf{param}, \mathsf{ID_S})$ to generate a key pair $(\mathsf{PEBKS.pk_S}, \mathsf{PEBKS.sk_S})$ and adds $(\mathsf{ID_S}, \mathsf{PEBKS.pk_S}, \mathsf{PEBKS.sk_S})$ to L_S. If $\mathsf{ID_S}$ has already been registered, $\mathcal{C}$ retrieves $\mathsf{PEBKS.pk_S}$ from L_S. Then, $\mathcal{C}$ runs $\mathsf{PEBKS.T}_w \leftarrow \mathsf{PEBKS.Trapdoor_R}(\mathsf{PEBKS.sk_{R^*}}, \mathsf{PEBKS.pk_S}, w)$ and returns $\mathsf{PEBKS.T}_w$ to $\mathcal{A}$.

- **Sender Trapdoor Generation Oracle**: On input $(\mathsf{ID}_\mathsf{R}, w)$ from $\mathcal{A}$, $\mathcal{C}$ searches the list L_R. If ID_R has not been registered, $\mathcal{C}$ runs $\mathsf{PEBKS.KeyGen}_\mathsf{R}(\mathsf{param}, \mathsf{ID}_\mathsf{R})$ to generate a key pair $(\mathsf{PEBKS.pk}_\mathsf{R}, \mathsf{PEBKS.sk}_\mathsf{R})$ and adds $(\mathsf{ID}_\mathsf{R}, \mathsf{PEBKS.pk}_\mathsf{R}, \mathsf{PEBKS.sk}_\mathsf{R})$ to L_R. If ID_R has already been registered, $\mathcal{C}$ retrieves $\mathsf{PEBKS.pk}_\mathsf{R}$ from L_R. Then, $\mathcal{C}$ runs $\mathsf{PEBKS.T}_w \leftarrow \mathsf{PEBKS.Trapdoor}_\mathsf{S}(\mathsf{PEBKS.sk}_{\mathsf{S}^*}, \mathsf{PEBKS.pk}_\mathsf{R}, w)$ and returns $\mathsf{PEBKS.T}_w$ to $\mathcal{A}$.
- **Encryption Oracle**: On input $(\mathsf{ID}_\mathsf{R}, w)$ from $\mathcal{A}$, $\mathcal{C}$ searches the list L_R. If ID_R has not been registered, $\mathcal{C}$ runs $\mathsf{PEBKS.KeyGen}_\mathsf{R}(\mathsf{param}, \mathsf{ID}_\mathsf{R})$ to generate a key pair ($\mathsf{PEBKS.pk}_\mathsf{R}$, $\mathsf{PEBKS.sk}_\mathsf{R}$) and adds $(\mathsf{ID}_\mathsf{R}, \mathsf{PEBKS.pk}_\mathsf{R}, \mathsf{PEBKS.sk}_\mathsf{R})$ to L_R. If ID_R has already been registered, $\mathcal{C}$ retrieves $\mathsf{PEBKS.pk}_\mathsf{R}$ from L_R. Then, $\mathcal{C}$ runs $\mathsf{PEBKS.C}_w \leftarrow \mathsf{PEBKS.KWEncrypt}$ $(\mathsf{PEBKS.sk}_{\mathsf{S}^*}, \mathsf{PEBKS.pk}_\mathsf{R}, w)$ and returns $\mathsf{PEBKS.C}_w$ to $\mathcal{A}$.

Challenge. $\mathcal{A}$ submits two keywords w_0^* and w_1^* to $\mathcal{C}$, under the condition that $(\mathsf{ID}_\mathsf{S}, w) \notin \{(\mathsf{ID}_{\mathsf{S}^*}, w_0^*), (\mathsf{ID}_{\mathsf{S}^*}, w_1^*)\}$ for all receiver trapdoor generation queries and $(\mathsf{ID}_\mathsf{R}, w) \notin \{(\mathsf{ID}_{\mathsf{R}^*}, w_0^*), (\mathsf{ID}_{\mathsf{R}^*}, w_1^*)\}$ for all sender trapdoor generation queries in Phase 1. $\mathcal{C}$ chooses a random bit $b \leftarrow \{0, 1\}$, generates the challenge ciphertext $\mathsf{C}^*_{w_b^*} \leftarrow \mathsf{PEBKS.KWEncrypt}(\mathsf{PEBKS.sk}_{\mathsf{S}^*}, \mathsf{PEBKS.pk}_{\mathsf{R}^*}, w_b^*)$, and returns $\mathsf{C}^*_{w_b^*}$ to $\mathcal{A}$.

Phase 2. $\mathcal{A}$ can issue a polynomial number of queries to the three oracles as in Phase 1, under the condition that $(\mathsf{ID}_\mathsf{S}, w) \notin \{(\mathsf{ID}_{\mathsf{S}^*}, w_0^*), (\mathsf{ID}_{\mathsf{S}^*}, w_1^*)\}$ for receiver trapdoor generation queries and $(\mathsf{ID}_\mathsf{R}, w) \notin \{(\mathsf{ID}_{\mathsf{R}^*}, w_0^*), (\mathsf{ID}_{\mathsf{R}^*}, w_1^*)\}$ for sender trapdoor generation queries.

Guess. $\mathcal{A}$ outputs a guess bit $b' \in \{0, 1\}$. We say that $\mathcal{A}$ wins if $b' = b$.

Definition 8 (IND-CKA Security)**.** *A PEBKS scheme* PEBKS *is said to be IND-CKA secure if for all PPT adversaries $\mathcal{A}$, the advantage in the above game*

$$\mathsf{Adv}^{\mathsf{IND\text{-}CKA}}_{\mathsf{PEBKS},\mathcal{A}}(\lambda) := \left| \Pr[b' = b] - \frac{1}{2} \right|$$

is negligible in the security parameter λ.

4 Key Collision Resistance

In this section, we newly define Key Collision Resistance of EPE. We also prove that the PRF-based EPE scheme by Li and Boyen [11] is Key Collision Resistant if the underlying PRF is Key Collision Resistant.

4.1 Key Collision Resistance of PRFs

We follow the notion of key collisions introduced by Taiyama et al. [15].

Key Collision Resistance of PRFs. Let $\lambda \in \mathbb{N}$ be a security parameter, let $\mathcal{K}$, $\mathcal{M}$, and $\mathcal{Y}$ be countable sets, and let $\mathcal{F} = \{f : \mathcal{M} \to \mathcal{Y}\}$ be the set of functions with domain $\mathcal{M}$ and range $\mathcal{Y}$. Let $\mathsf{PRF} : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ be a pseudorandom function. We say that PRF is *key collision resistant* if for all PPT adversaries $\mathcal{A}$ that output (K, K', M), the advantage

$$\mathsf{Adv}^{\mathsf{kc}}_{\mathsf{PRF},\mathcal{A}}(\lambda) := \Pr\left[\mathsf{PRF}(K, M) = \mathsf{PRF}(K', M) \wedge K \neq K'\right]$$

is negligible in λ.

Since AES is a pseudorandom permutation (PRP), different inputs always yield different outputs under the same key. In the case of different keys and different inputs, Taiyama et al. [15] found

key collisions for round-reduced AES, while they left finding key collisions for full-round AES as an open problem. Since no practical key collision attack is known against full-round AES, it is natural to assume that AES-based PRFs are Key Collision Resistant. We also consider HMAC, which Li and Boyen explicitly mention as an instantiation of the PRF. In the case of HMAC, $H(K||m)$ is computed, where K is the key and m is the message. Therefore, if two evaluations with different keys or different messages produced the same output, one would immediately obtain a collision in the underlying hash function. Hence, it is natural to assume that HMAC-based PRFs satisfy Key Collision Resistance when the underlying hash function is collision resistant. Therefore, we also assume that HMAC-based PRFs are Key Collision Resistant.

4.2 Key Collision Resistance of EPE

Next, we define Key Collision Resistance of EPE. This notion guarantees that for dk_w and σ_w generated with distinct keys K and K', $\mathsf{EPE.Qry}$ outputs 0 with overwhelming probability even if $w = w'$.

Key Collision Resistance of EPE. Let $\mathcal{C}$ be a challenger and $\mathcal{A}$ be an adversary. We define the experiment $\mathsf{Exp}^{\mathsf{kc}}_{\mathsf{EPE},\mathcal{A}}(\lambda)$ as follows.

Setup. $\mathcal{C}$ chooses keys $\mathsf{K}, \mathsf{K}' \leftarrow \mathcal{K}$ uniformly at random from the key space $\mathcal{K}$ such that $\mathsf{K} \neq \mathsf{K}'$.

Challenge. $\mathcal{A}$ outputs arbitrary keywords w and w'. We note that $w = w'$ may be held. $\mathcal{C}$ runs $\mathsf{dk}_w \leftarrow \mathsf{EPE.KG}(\mathsf{K}, w)$ and $\sigma_{w'} \leftarrow \mathsf{EPE.Enc}(\mathsf{K}', w')$. The experiment outputs 1 if $\mathsf{EPE.Qry}(\mathsf{dk}_w, \sigma_{w'}) = 1$, and 0 otherwise.

Definition 9 (**Key Collision Resistance**). *An EPE scheme* EPE *is said to be Key Collision Resistant if for all PPT adversaries* $\mathcal{A}$*, the advantage*

$$\mathsf{Adv}^{\mathsf{kc}}_{\mathsf{EPE},\mathcal{A}}(\lambda) := \Pr\left[\mathsf{Exp}^{\mathsf{kc}}_{\mathsf{EPE},\mathcal{A}}(\lambda) = 1\right]$$

is negligible in λ.

4.3 Analysis of the PRF-Based EPE Scheme

We first describe the PRF-based EPE scheme by Li and Boyen [11]. Let $\mathsf{PRF}_1 : \mathcal{K} \times \mathcal{KW} \to \mathcal{K}'$ and $\mathsf{PRF}_2 : \mathcal{K}' \times \mathcal{R} \to \mathcal{S}$ be pseudorandom functions. The construction is as follows.

$\mathsf{EPE.KG}(\mathsf{K}, w)$**:** Output $\mathsf{dk}_w = \mathsf{PRF}_1(\mathsf{K}, w)$.

$\mathsf{EPE.Enc}(\mathsf{K}, w)$**:** Choose $r \leftarrow \mathcal{R}$, compute $t = \mathsf{PRF}_2(\mathsf{PRF}_1(\mathsf{K}, w), r)$, and output $\sigma_w = (t, r)$.

$\mathsf{EPE.Qry}(\mathsf{dk}_w, \sigma_w)$**:** Parse $\sigma_w = (t, r)$. Output 1 if $t = \mathsf{PRF}_2(\mathsf{dk}_w, r)$, and 0 otherwise.

Li and Boyen [11] showed that this scheme is CT-IND secure if PRF_1 and PRF_2 are pseudorandom functions. In the following, we show that the PRF-based EPE scheme by Li and Boyen is Key Collision Resistant if the underlying PRF is Key Collision Resistant.

Theorem 1. *If* PRF_1 *is pseudorandom,* PRF_1 *and* PRF_2 *are Key Collision Resistant, and* $|\mathcal{K}'| > 2^\lambda$*, then the PRF-based EPE scheme is Key Collision Resistant. Specifically, for any PPT adversary* $\mathcal{A}$*, there exist PPT algorithms* $\mathcal{B}_1$*,* $\mathcal{B}_2$*, and* $\mathcal{B}_3$ *such that*

$$\mathsf{Adv}^{\mathsf{kc}}_{\mathsf{EPE},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{kc}}_{\mathsf{PRF}_1,\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{kc}}_{\mathsf{PRF}_2,\mathcal{B}_2}(\lambda) + 2\mathsf{Adv}^{\mathsf{pseudorandom}}_{\mathsf{PRF}_1,\mathcal{B}_3}(\lambda) + 2^{-\lambda}$$

Proof. For the keyword w output by $\mathcal{A}$, let $k := \mathsf{PRF}_1(\mathsf{K}, w)$ and $k' := \mathsf{PRF}_1(\mathsf{K}', w')$. Since $\mathsf{dk}_w = k$ and $\sigma_w = (\mathsf{PRF}_2(k', r), r)$, the equality $\mathsf{PRF}_2(k, r) = \mathsf{PRF}_2(k', r)$ holds when $\mathcal{A}$ wins. This equality holds in only two cases: $k = k'$ or $k \neq k'$. The former means that $\mathsf{K} \neq \mathsf{K}'$ and $\mathsf{PRF}_1(\mathsf{K}, w) = \mathsf{PRF}_1(\mathsf{K}', w')$. If $w = w'$, then it finds a key collision of PRF_1. If $w \neq w'$, then $\mathsf{PRF}_1(\mathsf{K}, w) \neq \mathsf{PRF}_1(\mathsf{K}', w')$ holds with overwhelming probability due to pseudorandomness of PRF_1 . The latter is a key collision of PRF_2. Based on this observation, we construct $\mathcal{B}_1$, $\mathcal{B}_2$, and $\mathcal{B}_3$ as follows. $\mathcal{B}_1$ and $\mathcal{B}_2$ choose random keys $\mathsf{K}, \mathsf{K}' \leftarrow \mathcal{K}$ satisfying $\mathsf{K} \neq \mathsf{K}'$ by themselves. After receiving w and w' from $\mathcal{A}$ where $w = w'$, they choose $r \leftarrow \mathcal{R}$ and compute k and k'. $\mathcal{B}_1$ outputs $(\mathsf{K}, \mathsf{K}', w)$, and $\mathcal{B}_2$ outputs (k, k', r). If $k = k'$, the output of $\mathcal{B}_1$ is a key collision of PRF_1. If $k \neq k'$, the output of $\mathcal{B}_2$ is a key collision of PRF_2. Finally, we need to consider the case that $\mathsf{K} \neq \mathsf{K}'$ and $w \neq w'$. In this case, $\mathsf{PRF}_1(\mathsf{K}, \cdot)$ and $\mathsf{PRF}_1(\mathsf{K}', \cdot)$ are regarded as different PRFs. Due to the pseudorandomness, they are indistinguishable from random functions. After the replacement, $\mathsf{PRF}_1(\mathsf{K}, w)$ and $\mathsf{PRF}_1(\mathsf{K}', w')$ are uniformly distributed over $\mathcal{K}'$. Thus, $\mathsf{PRF}_1(\mathsf{K}, w) = \mathsf{PRF}_1(\mathsf{K}', w')$ holds with the probability $1/|\mathcal{K}'| < 2^{-\lambda}$.

□

Remark. At first glance, one might attempt to derive Key Collision Resistance of the PRF-based EPE solely from the pseudorandomness of PRF_1, by viewing evaluations under distinct keys as evaluations of computationally independent random functions. However, such an argument obscures the different sources of collisions. In our proof, we intentionally distinguish the cases $w = w'$ and $w \neq w'$. The former corresponds to collisions across distinct PRF keys and is captured naturally by Key Collision Resistance of PRF_1, whereas the latter corresponds to collisions between outputs produced under distinct PRF keys and distinct inputs, and is analyzed through the pseudorandomness of PRF_1. This separation clarifies which type of collision is addressed by each assumption.

PEBKS does not consider TI (trapdoor indistinguishability) security, the PRF-based EPE scheme is sufficient for our construction. On the other hand, since the main goal of PAEKS is to achieve TI security, the pairing-based EPE scheme [14] or the lattice-based EPE scheme [11] needs to be used, especially to achieve full TI security (where the adversary is allowed to query the challenge keywords to the trapdoor oracle). These constructions also use PRFs (they compute a PRF on input a keyword w under an EPE key K). If the PRF is Key Collision Resistant, it is obvious that these EPE schemes are also Key Collision Resistant, and thus they can be used in the PAEKS construction without modification.

5 Proposed Generic Construction of PEBKS

In this section, we apply the generic construction of PAEKS from NIKE and EPE by Li-Boyen [11] to PEBKS. While the construction by Li-Boyen assumes unidirectional search (i.e., only the receiver can search), in our construction, the sender can also generate trapdoors by using, as the key of EPE, the shared key that the sender and the receiver non-interactively establish via NIKE. In the following, we construct a PEBKS scheme from a NIKE scheme $\mathsf{NIKE} = (\mathsf{NIKE.Setup}, \mathsf{NIKE.KeyGen}, \mathsf{SharedKey})$ and an EPE scheme $\mathsf{EPE} = (\mathsf{EPE.KG}, \mathsf{EPE.Enc}, \mathsf{EPE.Qry})$.

$\mathsf{Setup}(\lambda)$**:** Run $\mathsf{param} \leftarrow \mathsf{NIKE.Setup}(\lambda)$ and output param.

$\mathsf{KeyGen_S}(\mathsf{param}, \mathsf{ID_S})$**:** Run $(\mathsf{pk_S}, \mathsf{sk_S}) \leftarrow \mathsf{NIKE.KeyGen}(\mathsf{ID_S})$ and output $(\mathsf{PEBKS.pk_S}, \mathsf{PEBKS.sk_S}) = (\mathsf{pk_S}, \mathsf{sk_S})$.

$\mathsf{KeyGen_R}(\mathsf{param}, \mathsf{ID_R})$**:** Run $(\mathsf{pk_R}, \mathsf{sk_R}) \leftarrow \mathsf{NIKE.KeyGen}(\mathsf{ID_R})$ and output $(\mathsf{PEBKS.pk_R}, \mathsf{PEBKS.sk_R}) = (\mathsf{pk_R}, \mathsf{sk_R})$.

$\mathsf{Encrypt}(\mathsf{ID_S}, \mathsf{PEBKS.sk_S}, \mathsf{ID_R}, \mathsf{PEBKS.pk_R}, w)$**:** Parse $\mathsf{PEBKS.sk_S} = \mathsf{sk_S}$ and $\mathsf{PEBKS.pk_R} = \mathsf{pk_R}$. Run $\mathsf{K} \leftarrow \mathsf{SharedKey}(\mathsf{ID_R}, \mathsf{pk_R}, \mathsf{ID_S}, \mathsf{sk_S})$ and $\sigma_w \leftarrow \mathsf{EPE.Enc}(\mathsf{K}, w)$, and output $\mathsf{PEBKS.C}_w = \sigma_w$.

$\mathsf{Trapdoor_S}(\mathsf{ID_S}, \mathsf{PEBKS.sk_S}, \mathsf{ID_R}, \mathsf{PEBKS.pk_R}, w)$**:** Parse $\mathsf{PEBKS.sk_S} = \mathsf{sk_S}$ and $\mathsf{PEBKS.pk_R} = \mathsf{pk_R}$. Run $\mathsf{K} \leftarrow \mathsf{SharedKey}(\mathsf{ID_R}, \mathsf{pk_R}, \mathsf{ID_S}, \mathsf{sk_S})$ and $\mathsf{dk}_w \leftarrow \mathsf{EPE.KG}(\mathsf{K}, w)$, and output $\mathsf{PEBKS.Trapdoor} = \mathsf{dk}_w$.

$\mathsf{Trapdoor_R}(\mathsf{ID_R}, \mathsf{PEBKS.sk_R}, \mathsf{ID_S}, \mathsf{PEBKS.pk_S}, w)$**:** Parse $\mathsf{PEBKS.sk_R} = \mathsf{sk_R}$ and $\mathsf{PEBKS.pk_S} = \mathsf{pk_S}$. Run $\mathsf{K} \leftarrow \mathsf{SharedKey}(\mathsf{ID_S}, \mathsf{pk_S}, \mathsf{ID_R}, \mathsf{sk_R})$ and $\mathsf{dk}_w \leftarrow \mathsf{EPE.KG}(\mathsf{K}, w)$, and output $\mathsf{PEBKS.Trapdoor} = \mathsf{dk}_w$.

$\mathsf{Test}(\mathsf{PEBKS.Trapdoor}, \mathsf{PEBKS.C}_w)$**:** Parse $\mathsf{PEBKS.Trapdoor} = \mathsf{dk}_w$ and $\mathsf{PEBKS.C}_w = \sigma_w$. Output the result of $\mathsf{EPE.Qry}(\mathsf{dk}_w, \sigma_w)$.

Obviously, the proposed construction is correct if NIKE and EPE are correct.

6 Security Proofs

Theorem 2. *If the NIKE scheme* NIKE *is correct and HKR-CKS secure, and the EPE scheme* EPE *is Consistent and Key Collision Resistant, then the proposed PEBKS scheme is Consistent. Specifically, for all PPT adversaries $\mathcal{A}$, there exist PPT algorithms $\mathcal{B}_1$, $\mathcal{B}_2$, and $\mathcal{B}_3$ such that*

$$\mathsf{Adv}^{\mathsf{cons}}_{\mathsf{PEBKS},\mathcal{A}}(\lambda) \leq 4\mathsf{Adv}^{\mathsf{hkr\text{-}cks}}_{\mathsf{NIKE},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{cons}}_{\mathsf{EPE},\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathsf{kc}}_{\mathsf{EPE},\mathcal{B}_3}(\lambda).$$

Proof. Let (i, j, w, i', j', w') be the output of $\mathcal{A}$, and denote the shared key between S_i and R_j by $\mathsf{K}_{ij} := \mathsf{SharedKey}(\mathsf{ID}_{\mathsf{R}_j}, \mathsf{pk}_{\mathsf{R}_j}, \mathsf{ID}_{\mathsf{S}_i}, \mathsf{sk}_{\mathsf{S}_i})$. In the proposed construction, both PEBKS.KWEncrypt and $\mathsf{PEBKS.Trapdoor_S}$ compute the shared key in this way. Hence, we have $\mathsf{PEBKS.C}_w = \mathsf{EPE.Enc}(\mathsf{K}_{ij}, w)$, $\mathsf{PEBKS.T}_{w'} = \mathsf{EPE.KG}(\mathsf{K}_{i'j'}, w')$, and $\mathsf{PEBKS.Test}(\mathsf{PEBKS.T}_{w'}, \mathsf{PEBKS.C}_w) = \mathsf{EPE.Qry}(\mathsf{PEBKS.T}_{w'}, \mathsf{PEBKS.C}_w)$. Note that by the correctness of NIKE, the shared key $\mathsf{SharedKey}(\mathsf{ID}_{\mathsf{S}_{i'}}, \mathsf{pk}_{\mathsf{S}_{i'}}, \mathsf{ID}_{\mathsf{R}_{j'}}, \mathsf{sk}_{\mathsf{R}_{j'}})$ computed by $\mathsf{PEBKS.Trapdoor_R}$ also coincides with $\mathsf{K}_{i'j'}$ with overwhelming probability. Therefore, the same argument applies when $\mathsf{PEBKS.T}_{w'}$ is generated by $\mathsf{PEBKS.Trapdoor_R}$. $\mathcal{A}$ wins if $\mathsf{EPE.Qry}(\mathsf{PEBKS.T}_{w'}, \mathsf{PEBKS.C}_w) = 1$ and $(\mathsf{S}_i, \mathsf{R}_j, w) \neq (\mathsf{S}_{i'}, \mathsf{R}_{j'}, w')$. To reduce Consistency or Key Collision Resistance of EPE, we need to handle the situation that EPE keys K and K′ are randomly chosen by the challenger. Thus, we need to replace NIKE keys to be random. After the HKR-CKS hybrid transitions, each shared key is replaced by an independently random key. Therefore, whenever $(\mathsf{S}_i, \mathsf{R}_j) \neq (\mathsf{S}_{i'}, \mathsf{R}_{j'})$, the corresponding EPE keys are independently sampled and equal only with negligible probability since the shared-key space $\mathcal{K}$ is exponential in the security parameter λ.

The challenger of HKR-CKS security sends param to $\mathcal{B}_1$. $\mathcal{B}_1$ sends $\mathsf{ID}_{\mathsf{S}_0}$, $\mathsf{ID}_{\mathsf{S}_1}$, $\mathsf{ID}_{\mathsf{R}_0}$, and $\mathsf{ID}_{\mathsf{R}_1}$ to the honest registration oracle, and obtains $(\mathsf{PEBKS.pk}_{\mathsf{S}_0}, \mathsf{PEBKS.pk}_{\mathsf{S}_1}, \mathsf{PEBKS.pk}_{\mathsf{R}_0}, \mathsf{PEBKS.pk}_{\mathsf{R}_1})$. $\mathcal{B}_1$ sends $(\mathsf{param}, \mathsf{PEBKS.pk}_{\mathsf{S}_0}, \mathsf{PEBKS.pk}_{\mathsf{S}_1}, \mathsf{PEBKS.pk}_{\mathsf{R}_0}, \mathsf{PEBKS.pk}_{\mathsf{R}_1})$ to $\mathcal{A}$. Since there are four NIKE keys, we can replace these keys to be random via the HKR-CKS security. By a standard sequence of four hybrids, each replacing one shared key with a random key, the distinguishing gap is bounded by $4\mathsf{Adv}^{\mathsf{hkr\text{-}cks}}_{\mathsf{NIKE},\mathcal{B}_1}(\lambda)$. After these replacements, we consider the following two cases.

(i) $(\mathsf{S}_i, \mathsf{R}_j) = (\mathsf{S}_{i'}, \mathsf{R}_{j'})$ and $w \neq w'$. In this case, $\mathsf{PEBKS.C}_w$ and $\mathsf{PEBKS.T}_{w'}$ are generated for distinct keywords under the same shared key K_{ij}. We construct an algorithm $\mathcal{B}_2$ that breaks Consistency of EPE as follows. $\mathcal{B}_2$ generates $(\mathsf{param}, \mathsf{PEBKS.pk}_{\mathsf{S}_0}, \mathsf{PEBKS.pk}_{\mathsf{S}_1}, \mathsf{PEBKS.pk}_{\mathsf{R}_0}, \mathsf{PEBKS.pk}_{\mathsf{R}_1})$ and sends them to $\mathcal{A}$. $\mathcal{A}$ outputs (i, j, w, i', j', w'). Since $\mathcal{A}$ wins, we have $\mathsf{EPE.Qry}(\mathsf{dk}_{w'}, \sigma_w) = 1$, and $\mathcal{B}_2$ breaks Consistency of EPE.

(ii) $(\mathsf{S}_i, \mathsf{R}_j) \neq (\mathsf{S}_{i'}, \mathsf{R}_{j'})$. In this case, $\mathsf{PEBKS.C}_w$ and $\mathsf{PEBKS.T}_{w'}$ are generated under the distinct shared keys K_{ij} and $\mathsf{K}_{i'j'}$. $\mathcal{B}_3$ generates $(\mathsf{param}, \mathsf{PEBKS.pk}_{\mathsf{S}_0}, \mathsf{PEBKS.pk}_{\mathsf{S}_1}, \mathsf{PEBKS.pk}_{\mathsf{R}_0}, \mathsf{PEBKS.pk}_{\mathsf{R}_1})$ and sends them to $\mathcal{A}$. $\mathcal{A}$ outputs (i, j, w, i', j', w'). $\mathcal{B}_3$ outputs w and w' in the key collision resistance experiment of EPE. Since $\mathsf{EPE.Qry}(\mathsf{dk}_w, \sigma_{w'}) = 1$, $\mathcal{B}_3$ breaks Key Collision Resistance of EPE. □

Theorem 3. *If the NIKE scheme* NIKE *is HKR-CKS secure and the EPE scheme* EPE *is CT-IND secure, then the proposed construction is IND-CKA secure. Specifically, for all PPT adversaries $\mathcal{A}$, there exist PPT algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$ such that*

$$\mathsf{Adv}^{\mathsf{IND\text{-}CKA}}_{\mathsf{PEBKS},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{hkr\text{-}cks}}_{\mathsf{NIKE},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{ct\text{-}ind}}_{\mathsf{EPE},\mathcal{B}_2}(\lambda)$$

Proof. Let S_i denote the event that $\mathcal{A}$ outputs a guess bit b' such that $b' = b$ in Game i ($i \in \{0, 1\}$).

Game 0. This game is exactly the IND-CKA security experiment for the PEBKS scheme. $\mathcal{C}$ generates the parameters and key pairs according to the proposed generic construction and correctly answers the queries from $\mathcal{A}$. In the challenge phase, $\mathcal{A}$ submits two keywords w_0^* and w_1^*. $\mathcal{C}$ chooses a random bit $b \leftarrow \{0, 1\}$ and derives the shared key $\mathsf{K} \leftarrow \mathsf{SharedKey}(\mathsf{ID}_{\mathsf{R}^*}, \mathsf{pk}_{\mathsf{R}^*}, \mathsf{ID}_{\mathsf{S}^*}, \mathsf{sk}_{\mathsf{S}^*})$ according to the algorithm. Then, $\mathcal{C}$ generates the challenge ciphertext $\mathsf{C}^*_{w_b^*} \leftarrow \mathsf{EPE.Enc}(\mathsf{K}, w_b^*)$ and returns it to $\mathcal{A}$. By definition, we have

$$\mathsf{Adv}^{\mathsf{IND\text{-}CKA}}_{\mathsf{PEBKS},\mathcal{A}}(\lambda) = \left| \Pr[S_0] - \frac{1}{2} \right|.$$

Game 1. We modify the generation of the challenge ciphertext in Game 0 as follows. $\mathcal{C}$ replaces the shared key K between $\mathsf{ID}_{\mathsf{S}^*}$ and $\mathsf{ID}_{\mathsf{R}^*}$ with a key $\mathsf{K}^* \leftarrow \mathcal{K}$ chosen uniformly at random, generates the challenge ciphertext $\mathsf{C}^*_{w_b^*} \leftarrow \mathsf{EPE.Enc}(\mathsf{K}^*, w_b^*)$ by using K^*, and returns it to $\mathcal{A}$. □

Lemma 1. *If the NIKE scheme* NIKE *is HKR-CKS secure, then for any PPT adversary $\mathcal{A}$, there exists a PPT algorithm $\mathcal{B}_1$ such that*

$$|\Pr[S_0] - \Pr[S_1]| \leq \mathsf{Adv}^{\mathsf{hkr\text{-}cks}}_{\mathsf{NIKE},\mathcal{B}_1}(\lambda).$$

Proof. $\mathcal{B}_1$ interacts with the challenger $\mathcal{C}_{\mathsf{NIKE}}$ of the HKR-CKS security game while acting as the challenger $\mathcal{C}$ of PEBKS for $\mathcal{A}$.

Setup. $\mathcal{B}_1$ receives the parameters param from $\mathcal{C}_{\mathsf{NIKE}}$. Next, $\mathcal{B}_1$ initializes empty lists L_{R} and L_{S}. Then, $\mathcal{B}_1$ submits $\mathsf{ID}_{\mathsf{R}^*}$ and $\mathsf{ID}_{\mathsf{S}^*}$ to the oracle RegHon of $\mathcal{C}_{\mathsf{NIKE}}$ and obtains $\mathsf{pk}_{\mathsf{R}^*}$ and $\mathsf{pk}_{\mathsf{S}^*}$. $\mathcal{B}_1$ adds $(\mathsf{ID}_{\mathsf{R}^*}, \mathsf{pk}_{\mathsf{R}^*}, \perp)$ to L_{R} and $(\mathsf{ID}_{\mathsf{S}^*}, \mathsf{pk}_{\mathsf{S}^*}, \perp)$ to L_{S}, where $\perp$ indicates that $\mathcal{B}_1$ does not hold the secret key corresponding to the identifier. After that, $\mathcal{B}_1$ submits the two identifiers $(\mathsf{ID}_{\mathsf{R}^*}, \mathsf{ID}_{\mathsf{S}^*})$ to $\mathcal{C}_{\mathsf{NIKE}}$ and receives the challenge key K^*_μ. Finally, $\mathcal{B}_1$ sends param, $\mathsf{pk}_{\mathsf{R}^*}$, and $\mathsf{pk}_{\mathsf{S}^*}$ to $\mathcal{A}$ and keeps K^*_μ.

Phase 1. $\mathcal{A}$ can issue a polynomial number of queries to the following three oracles.

- **Receiver Trapdoor Generation Oracle**: On input $(\mathsf{ID}_{\mathsf{S}}, w)$ from $\mathcal{A}$, if $\mathsf{ID}_{\mathsf{S}} = \mathsf{ID}_{\mathsf{S}^*}$, $\mathcal{B}_1$ runs $\mathsf{T}_w \leftarrow \mathsf{EPE.KG}(\mathsf{K}^*_\mu, w)$ by using the challenge key K^*_μ. If $\mathsf{ID}_{\mathsf{S}} \neq \mathsf{ID}_{\mathsf{S}^*}$, $\mathcal{B}_1$ searches the list L_{S}. If ID_{S} has not been registered, $\mathcal{B}_1$ runs $\mathsf{NIKE.KeyGen}(\mathsf{ID}_{\mathsf{S}})$ to generate a key pair $(\mathsf{pk}_{\mathsf{S}}, \mathsf{sk}_{\mathsf{S}})$ and adds $(\mathsf{ID}_{\mathsf{S}}, \mathsf{pk}_{\mathsf{S}}, \mathsf{sk}_{\mathsf{S}})$ to L_{S}. $\mathcal{B}_1$ then runs $\mathsf{K}' \leftarrow \mathsf{SharedKey}(\mathsf{ID}_{\mathsf{R}^*}, \mathsf{pk}_{\mathsf{R}^*}, \mathsf{ID}_{\mathsf{S}}, \mathsf{sk}_{\mathsf{S}})$ by using sk_{S} stored in L_{S}, and runs $\mathsf{T}_w \leftarrow \mathsf{EPE.KG}(\mathsf{K}', w)$. $\mathcal{B}_1$ returns T_w to $\mathcal{A}$.

- **Sender Trapdoor Generation Oracle**: On input $(\mathsf{ID}_\mathsf{R}, w)$ from $\mathcal{A}$, if $\mathsf{ID}_\mathsf{R} = \mathsf{ID}_{\mathsf{R}^*}$, $\mathcal{B}_1$ runs $\mathsf{T}_w \leftarrow \mathsf{EPE.KG}(\mathsf{K}^*_\mu, w)$ by using the challenge key K^*_μ. If $\mathsf{ID}_\mathsf{R} \neq \mathsf{ID}_{\mathsf{R}^*}$, $\mathcal{B}_1$ searches the list L_R. If ID_R has not been registered, $\mathcal{B}_1$ runs $\mathsf{NIKE.KeyGen}(\mathsf{ID}_\mathsf{R})$ to generate a key pair $(\mathsf{pk}_\mathsf{R}, \mathsf{sk}_\mathsf{R})$ and adds $(\mathsf{ID}_\mathsf{R}, \mathsf{pk}_\mathsf{R}, \mathsf{sk}_\mathsf{R})$ to L_R. $\mathcal{B}_1$ then runs $\mathsf{K}' \leftarrow \mathsf{SharedKey}(\mathsf{ID}_{\mathsf{S}^*}, \mathsf{pk}_{\mathsf{S}^*}, \mathsf{ID}_\mathsf{R}, \mathsf{sk}_\mathsf{R})$ by using sk_R stored in L_R, and runs $\mathsf{T}_w \leftarrow \mathsf{EPE.KG}(\mathsf{K}', w)$. $\mathcal{B}_1$ returns T_w to $\mathcal{A}$.
- **Encryption Oracle**: On input $(\mathsf{ID}_\mathsf{R}, w)$ from $\mathcal{A}$, if $\mathsf{ID}_\mathsf{R} = \mathsf{ID}_{\mathsf{R}^*}$, $\mathcal{B}_1$ runs $\mathsf{C}_w \leftarrow \mathsf{EPE.Enc}(\mathsf{K}^*_\mu, w)$. If $\mathsf{ID}_\mathsf{R} \neq \mathsf{ID}_{\mathsf{R}^*}$, $\mathcal{B}_1$ searches the list L_R. If ID_R has not been registered, $\mathcal{B}_1$ runs $\mathsf{NIKE.KeyGen}(\mathsf{ID}_\mathsf{R})$ to generate a key pair $(\mathsf{pk}_\mathsf{R}, \mathsf{sk}_\mathsf{R})$ and adds $(\mathsf{ID}_\mathsf{R}, \mathsf{pk}_\mathsf{R}, \mathsf{sk}_\mathsf{R})$ to L_R. $\mathcal{B}_1$ then runs $\mathsf{K}' \leftarrow \mathsf{SharedKey}(\mathsf{ID}_{\mathsf{S}^*}, \mathsf{pk}_{\mathsf{S}^*}, \mathsf{ID}_\mathsf{R}, \mathsf{sk}_\mathsf{R})$ by using sk_R stored in L_R, and runs $\mathsf{C}_w \leftarrow \mathsf{EPE.Enc}(\mathsf{K}', w)$. $\mathcal{B}_1$ returns C_w to $\mathcal{A}$.

Challenge. $\mathcal{A}$ submits two distinct keywords w_0^* and w_1^* to $\mathcal{B}_1$. $\mathcal{B}_1$ chooses a random bit $b \leftarrow \{0,1\}$, runs $\mathsf{C}^*_{w_b^*} \leftarrow \mathsf{EPE.Enc}(\mathsf{K}^*_\mu, w_b^*)$, and returns the challenge ciphertext $\mathsf{C}^*_{w_b^*}$ to $\mathcal{A}$.

Phase 2. $\mathcal{B}_1$ responds to the queries in the same way as in Phase 1.

Guess. $\mathcal{A}$ outputs a guess bit $b' \in \{0,1\}$. $\mathcal{B}_1$ outputs $\mu' = 0$ to $\mathcal{C}_\mathsf{NIKE}$ if $b' = b$, and $\mu' = 1$ otherwise.

Correctness of the Simulation. First, note that $\mathcal{B}_1$ holds neither $\mathsf{sk}_{\mathsf{R}^*}$ nor $\mathsf{sk}_{\mathsf{S}^*}$. If a query of $\mathcal{A}$ concerns the target pair $(\mathsf{ID}_{\mathsf{S}^*}, \mathsf{ID}_{\mathsf{R}^*})$, $\mathcal{B}_1$ can use K^*_μ obtained from $\mathcal{C}_\mathsf{NIKE}$. If a query of $\mathcal{A}$ concerns a pair other than the target pair $(\mathsf{ID}_{\mathsf{S}^*}, \mathsf{ID}_{\mathsf{R}^*})$, $\mathcal{B}_1$ derives the shared key by using the secret key of the sender or receiver specified by $\mathcal{A}$. By the correctness of NIKE, $\mathsf{SharedKey}(\mathsf{ID}_{\mathsf{R}^*}, \mathsf{pk}_{\mathsf{R}^*}, \mathsf{ID}_\mathsf{S}, \mathsf{sk}_\mathsf{S}) = \mathsf{SharedKey}(\mathsf{ID}_\mathsf{S}, \mathsf{pk}_\mathsf{S}, \mathsf{ID}_{\mathsf{R}^*}, \mathsf{sk}_{\mathsf{R}^*})$ holds with overwhelming probability. That is, when running $\mathsf{Trapdoor}_\mathsf{R}(\mathsf{ID}_{\mathsf{R}^*}, \mathsf{sk}_{\mathsf{R}^*}, \mathsf{ID}_\mathsf{S}, \mathsf{pk}_\mathsf{S}, w)$, $\mathcal{B}_1$ can use the NIKE key K computed by $\mathsf{SharedKey}(\mathsf{ID}_{\mathsf{R}^*}, \mathsf{pk}_{\mathsf{R}^*}, \mathsf{ID}_\mathsf{S}, \mathsf{sk}_\mathsf{S})$. Similarly, when running $\mathsf{Trapdoor}_\mathsf{S}(\mathsf{ID}_{\mathsf{S}^*}, \mathsf{sk}_{\mathsf{S}^*}, \mathsf{ID}_\mathsf{R}, \mathsf{pk}_\mathsf{R}, w)$ and $\mathsf{Encrypt}(\mathsf{ID}_{\mathsf{S}^*}, \mathsf{sk}_{\mathsf{S}^*}, \mathsf{ID}_\mathsf{R}, \mathsf{pk}_\mathsf{R}, w)$, $\mathcal{B}_1$ can use the NIKE key K computed by $\mathsf{SharedKey}(\mathsf{ID}_{\mathsf{S}^*}, \mathsf{pk}_{\mathsf{S}^*}, \mathsf{ID}_\mathsf{R}, \mathsf{sk}_\mathsf{R})$. By the symmetry of NIKE, $\mathcal{B}_1$ can correctly simulate all the oracles without using any secret key of the targets.

If the bit chosen by $\mathcal{C}_\mathsf{NIKE}$ is $\mu = 0$, K^*_μ is the shared key derived by $\mathsf{SharedKey}$, and thus the view of $\mathcal{A}$ is identical to that in Game 0. If $\mu = 1$, K^*_μ is chosen uniformly at random from the key space $\mathcal{K}$, and thus the view of $\mathcal{A}$ is identical to that in Game 1. In other words, the advantage of $\mathcal{A}$ in distinguishing whether the shared key is generated according to the syntax or chosen at random is directly translated into the advantage of $\mathcal{B}_1$. Therefore, we have $|\Pr[S_0] - \Pr[S_1]| \leq \mathsf{Adv}^{\mathsf{hkr\text{-}cks}}_{\mathsf{NIKE},\mathcal{B}_1}(\lambda)$.

Validity of the Simulation. By the correctness of NIKE, with overwhelming probability, the view of $\mathcal{A}$ simulated by $\mathcal{B}_1$ is identically distributed to that in Game 0 when $\mu = 0$, and to that in Game 1 when $\mu = 1$.

The view of $\mathcal{A}$ consists of the public information $(\mathsf{param}, \mathsf{pk}_{\mathsf{S}^*}, \mathsf{pk}_{\mathsf{R}^*})$, all the oracle responses in Phase 1 and Phase 2, and the challenge ciphertext. All key pairs are generated by $\mathsf{NIKE.KeyGen}$. Moreover, owing to the lists L_R and L_S, the same identifier always corresponds to the same key pair. Hence, the distribution of the public information is identical to that in the real game. In both the real game and the simulation by $\mathcal{B}_1$, each response received by $\mathcal{A}$ is an output of $\mathsf{EPE.KG}$ or $\mathsf{EPE.Enc}$ on the shared key assigned to the sender–receiver pair involved in the query and the keyword w. Therefore, it suffices to show that the shared key assigned to each pair coincides with that in the real game. There are the following three types of identifier pairs.

- **Target pair** $(\mathsf{ID}_{\mathsf{S}^*}, \mathsf{ID}_{\mathsf{R}^*})$**.** $\mathcal{B}_1$ uses K^*_μ in all of the receiver trapdoor generation with $\mathsf{ID}_\mathsf{S} = \mathsf{ID}_{\mathsf{S}^*}$, the sender trapdoor generation and the encryption with $\mathsf{ID}_\mathsf{R} = \mathsf{ID}_{\mathsf{R}^*}$, and the generation of the challenge ciphertext. When $\mu = 0$, $\mathsf{K}^*_0 = \mathsf{SharedKey}(\mathsf{ID}_{\mathsf{R}^*}, \mathsf{pk}_{\mathsf{R}^*}, \mathsf{ID}_{\mathsf{S}^*}, \mathsf{sk}_{\mathsf{S}^*}) = \mathsf{K}$, which

coincides with the key used for the target pair in Game 0. When $\mu = 1$, K_1^* is uniformly distributed over $\mathcal{K}$, which is identically distributed to K^* in Game 1.

- **Pair** $(\mathsf{ID}_{\mathsf{S}^*}, \mathsf{ID}_{\mathsf{R}})$**.** In the real game, $\mathsf{SharedKey}(\mathsf{ID}_{\mathsf{R}}, \mathsf{pk}_{\mathsf{R}}, \mathsf{ID}_{\mathsf{S}^*}, \mathsf{sk}_{\mathsf{S}^*})$ is used. By the correctness of NIKE, $\mathsf{K}' = \mathsf{SharedKey}(\mathsf{ID}_{\mathsf{S}^*}, \mathsf{pk}_{\mathsf{S}^*}, \mathsf{ID}_{\mathsf{R}}, \mathsf{sk}_{\mathsf{R}})$ used by $\mathcal{B}_1$ coincides with it with overwhelming probability.
- **Pair** $(\mathsf{ID}_{\mathsf{S}}, \mathsf{ID}_{\mathsf{R}^*})$**.** In the real game, $\mathsf{SharedKey}(\mathsf{ID}_{\mathsf{S}}, \mathsf{pk}_{\mathsf{S}}, \mathsf{ID}_{\mathsf{R}^*}, \mathsf{sk}_{\mathsf{R}^*})$ is used. By the correctness of NIKE, $\mathsf{K}' = \mathsf{SharedKey}(\mathsf{ID}_{\mathsf{R}^*}, \mathsf{pk}_{\mathsf{R}^*}, \mathsf{ID}_{\mathsf{S}}, \mathsf{sk}_{\mathsf{S}})$ used by $\mathcal{B}_1$ coincides with it with overwhelming probability.

In any case, since $\mathsf{SharedKey}$ is deterministic, the same key is used for queries involving the same pair, regardless of which oracle is queried. Hence, the simulation is consistent. □

Lemma 2. *If the EPE scheme* EPE *is CT-IND secure, then for any PPT adversary* $\mathcal{A}$*, there exists a PPT algorithm* $\mathcal{B}_2$ *such that*

$$\left|\Pr[S_1] - \frac{1}{2}\right| \leq \mathsf{Adv}^{\mathsf{ct\text{-}ind}}_{\mathsf{EPE},\mathcal{B}_2}(\lambda).$$

Proof. $\mathcal{B}_2$ interacts with the challenger $\mathcal{C}_{\mathsf{EPE}}$ of the CT-IND security game while simulating the environment of Game 1 for $\mathcal{A}$. $\mathcal{C}_{\mathsf{EPE}}$ secretly holds a shared key $\mathsf{K} \leftarrow \mathcal{K}$, and $\mathcal{B}_2$ identifies the shared key of the target pair with this K.

Setup. First, $\mathcal{B}_2$ runs $\mathsf{NIKE.Setup}(\lambda)$ to generate the parameters param and initializes empty lists L_{R} and L_{S}. Next, $\mathcal{B}_2$ runs $\mathsf{NIKE.KeyGen}(\mathsf{ID}_{\mathsf{R}^*})$ and $\mathsf{NIKE.KeyGen}(\mathsf{ID}_{\mathsf{S}^*})$ to obtain key pairs, and adds $(\mathsf{ID}_{\mathsf{R}^*}, \mathsf{pk}_{\mathsf{R}^*}, \mathsf{sk}_{\mathsf{R}^*})$ to L_{R} and $(\mathsf{ID}_{\mathsf{S}^*}, \mathsf{pk}_{\mathsf{S}^*}, \mathsf{sk}_{\mathsf{S}^*})$ to L_{S}. Finally, $\mathcal{B}_2$ sends param, $\mathsf{pk}_{\mathsf{R}^*}$, and $\mathsf{pk}_{\mathsf{S}^*}$ to $\mathcal{A}$.

Phase 1. $\mathcal{A}$ can issue a polynomial number of queries to the following three oracles.

- **Receiver Trapdoor Generation Oracle**: On input $(\mathsf{ID}_{\mathsf{S}}, w)$ from $\mathcal{A}$, if $\mathsf{ID}_{\mathsf{S}} = \mathsf{ID}_{\mathsf{S}^*}$, $\mathcal{B}_2$ submits w to the query key generation oracle $K_{\mathsf{K}}(\cdot)$ of $\mathcal{C}_{\mathsf{EPE}}$ and returns the obtained dk_w to $\mathcal{A}$ as T_w. If $\mathsf{ID}_{\mathsf{S}} \neq \mathsf{ID}_{\mathsf{S}^*}$, $\mathcal{B}_2$ searches the list L_{S}. If ID_{S} has not been registered, $\mathcal{B}_2$ runs $\mathsf{NIKE.KeyGen}(\mathsf{ID}_{\mathsf{S}})$ to generate a key pair $(\mathsf{pk}_{\mathsf{S}}, \mathsf{sk}_{\mathsf{S}})$ and adds $(\mathsf{ID}_{\mathsf{S}}, \mathsf{pk}_{\mathsf{S}}, \mathsf{sk}_{\mathsf{S}})$ to L_{S}. $\mathcal{B}_2$ then runs $\mathsf{K}' \leftarrow \mathsf{SharedKey}(\mathsf{ID}_{\mathsf{R}^*}, \mathsf{pk}_{\mathsf{R}^*}, \mathsf{ID}_{\mathsf{S}}, \mathsf{sk}_{\mathsf{S}})$ by using sk_{S} stored in L_{S}, runs $\mathsf{T}_w \leftarrow \mathsf{EPE.KG}(\mathsf{K}', w)$, and returns T_w to $\mathcal{A}$.
- **Sender Trapdoor Generation Oracle**: On input $(\mathsf{ID}_{\mathsf{R}}, w)$ from $\mathcal{A}$, if $\mathsf{ID}_{\mathsf{R}} = \mathsf{ID}_{\mathsf{R}^*}$, $\mathcal{B}_2$ submits w to the query key generation oracle $K_{\mathsf{K}}(\cdot)$ of $\mathcal{C}_{\mathsf{EPE}}$ and returns the obtained dk_w to $\mathcal{A}$ as T_w. If $\mathsf{ID}_{\mathsf{R}} \neq \mathsf{ID}_{\mathsf{R}^*}$, $\mathcal{B}_2$ searches the list L_{R}. If ID_{R} has not been registered, $\mathcal{B}_2$ runs $\mathsf{NIKE.KeyGen}(\mathsf{ID}_{\mathsf{R}})$ to generate a key pair $(\mathsf{pk}_{\mathsf{R}}, \mathsf{sk}_{\mathsf{R}})$ and adds $(\mathsf{ID}_{\mathsf{R}}, \mathsf{pk}_{\mathsf{R}}, \mathsf{sk}_{\mathsf{R}})$ to L_{R}. $\mathcal{B}_2$ then runs $\mathsf{K}' \leftarrow \mathsf{SharedKey}(\mathsf{ID}_{\mathsf{S}^*}, \mathsf{pk}_{\mathsf{S}^*}, \mathsf{ID}_{\mathsf{R}}, \mathsf{sk}_{\mathsf{R}})$ by using sk_{R} stored in L_{R}, runs $\mathsf{T}_w \leftarrow \mathsf{EPE.KG}(\mathsf{K}', w)$, and returns T_w to $\mathcal{A}$.
- **Encryption Oracle**: On input $(\mathsf{ID}_{\mathsf{R}}, w)$ from $\mathcal{A}$, if $\mathsf{ID}_{\mathsf{R}} = \mathsf{ID}_{\mathsf{R}^*}$, $\mathcal{B}_2$ submits w to the encryption oracle $E_{\mathsf{K}}(\cdot)$ of $\mathcal{C}_{\mathsf{EPE}}$ and returns the obtained σ_w to $\mathcal{A}$ as C_w. If $\mathsf{ID}_{\mathsf{R}} \neq \mathsf{ID}_{\mathsf{R}^*}$, $\mathcal{B}_2$ searches the list L_{R}. If ID_{R} has not been registered, $\mathcal{B}_2$ runs $\mathsf{NIKE.KeyGen}(\mathsf{ID}_{\mathsf{R}})$ to generate a key pair $(\mathsf{pk}_{\mathsf{R}}, \mathsf{sk}_{\mathsf{R}})$ and adds $(\mathsf{ID}_{\mathsf{R}}, \mathsf{pk}_{\mathsf{R}}, \mathsf{sk}_{\mathsf{R}})$ to L_{R}. $\mathcal{B}_2$ then runs $\mathsf{K}' \leftarrow \mathsf{SharedKey}(\mathsf{ID}_{\mathsf{S}^*}, \mathsf{pk}_{\mathsf{S}^*}, \mathsf{ID}_{\mathsf{R}}, \mathsf{sk}_{\mathsf{R}})$ by using sk_{R} stored in L_{R}, runs $\mathsf{C}_w \leftarrow \mathsf{EPE.Enc}(\mathsf{K}', w)$, and returns C_w to $\mathcal{A}$.

Challenge. $\mathcal{A}$ submits two distinct keywords w_0^* and w_1^* to $\mathcal{B}_2$. $\mathcal{B}_2$ submits w_0^* and w_1^* to $\mathcal{C}_{\mathsf{EPE}}$ as the challenge keywords of EPE. $\mathcal{C}_{\mathsf{EPE}}$ chooses a random bit $b \leftarrow \{0,1\}$, runs $\sigma^*_{w_b^*} \leftarrow \mathsf{EPE.Enc}(\mathsf{K}, w_b^*)$, and returns the challenge ciphertext $\sigma^*_{w_b^*}$ to $\mathcal{B}_2$. $\mathcal{B}_2$ returns it to $\mathcal{A}$ as $\mathsf{C}^*_{w_b^*}$.

Phase 2. $\mathcal{B}_2$ responds to the queries in the same way as in Phase 1.

Guess. $\mathcal{A}$ outputs a guess bit $b' \in \{0,1\}$. $\mathcal{B}_2$ outputs b' to $\mathcal{C}_{\mathsf{EPE}}$.

Correctness of the Simulation. By the definition of Game 1, the shared key of the target pair is $\mathsf{K}^* \leftarrow \mathcal{K}$, which is identically distributed to $\mathsf{K} \leftarrow \mathcal{K}$ held by $\mathcal{C}_{\mathsf{EPE}}$. Since we identify the key space of NIKE with that of EPE in this paper, this identification is consistent.

Furthermore, we confirm that the oracle restrictions in the two games exactly correspond to each other. The restriction in Phase 2 prohibits $\mathcal{A}$ from querying $(\mathsf{ID}_{\mathsf{S}^*}, w_0^*)$ and $(\mathsf{ID}_{\mathsf{S}^*}, w_1^*)$ to the receiver trapdoor generation oracle and $(\mathsf{ID}_{\mathsf{R}^*}, w_0^*)$ and $(\mathsf{ID}_{\mathsf{R}^*}, w_1^*)$ to the sender trapdoor generation oracle. These coincide with the queries that $\mathcal{B}_2$ would make to $K_{\mathsf{K}}(\cdot)$ with $w \in \{w_0^*, w_1^*\}$, and thus $\mathcal{B}_2$ never violates the restriction on the query key generation oracle in the CT-IND security game. On the other hand, since there is no restriction on the encryption oracle in the CT-IND game, $\mathcal{B}_2$ can simulate the encryption oracle in the IND-CKA game. Therefore, the view of $\mathcal{A}$ is identical to that in Game 1, and we have $\Pr[\mathcal{B}_2 \text{ wins}] = \Pr[S_1]$.

Validity of the Simulation. By the correctness of NIKE, with overwhelming probability, the view of $\mathcal{A}$ simulated by $\mathcal{B}_2$ is identically distributed to that in Game 1. Since the distribution of the public information is identical to that in the real game and each response is an output of $\mathsf{EPE.KG}$ or $\mathsf{EPE.Enc}$ under the shared key assigned to the pair, we show that the shared key assigned to each pair coincides with that in Game 1.

- **Target pair** $(\mathsf{ID}_{\mathsf{S}^*}, \mathsf{ID}_{\mathsf{R}^*})$**.** $\mathcal{B}_2$ forwards all queries involving the target pair to the encryption oracle $E_{\mathsf{K}}(\cdot)$ or the query key generation oracle $K_{\mathsf{K}}(\cdot)$ of $\mathcal{C}_{\mathsf{EPE}}$, and the challenge ciphertext is also generated by $\mathcal{C}_{\mathsf{EPE}}$. $\mathcal{C}_{\mathsf{EPE}}$ generates all of them by running $\mathsf{EPE.KG}$ or $\mathsf{EPE.Enc}$ under the single key $\mathsf{K} \leftarrow \mathcal{K}$. Since K is identically distributed to K^* in Game 1 and the challenge bit b is also chosen uniformly at random, the responses for the target pair are identically distributed to those in Game 1.
- **Pairs other than the target pair.** As in Lemma 1, by the correctness of NIKE, the key K' used by $\mathcal{B}_2$ coincides with the key in the real game with overwhelming probability.

□

By Lemmas 1 and 2 and applying the triangle inequality, we obtain

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{IND\text{-}CKA}}_{\mathsf{PEBKS},\mathcal{A}}(\lambda) &= \left|\Pr[S_0] - \frac{1}{2}\right| \\
&\leq |\Pr[S_0] - \Pr[S_1]| + \left|\Pr[S_1] - \frac{1}{2}\right| \\
&\leq \mathsf{Adv}^{\mathsf{hkr\text{-}cks}}_{\mathsf{NIKE},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{ct\text{-}ind}}_{\mathsf{EPE},\mathcal{B}_2}(\lambda).
\end{aligned}$$

Therefore, the PEBKS scheme based on the proposed generic construction is IND-CKA secure.

7 Conclusion

In this paper, focusing on the similarity between PEBKS and PAEKS, we applied the generic construction methodology of PAEKS by Li and Boyen [11] and showed that PEBKS can be generically constructed from NIKE and EPE. While only concrete pairing-based PEBKS schemes have been proposed so far, our generic construction yields PEBKS schemes based on various computational assumptions, such as pairing-free discrete-logarithm-type assumptions, lattices, isogenies, and codes.

PEBKS was proposed as an extension of PEKS and requires that ciphertexts leak no information about keywords; however, information leakage about keywords from trapdoors, i.e., TI security, has not been considered. One of the original goals of PAEKS is to provide resistance to keyword guessing attacks through TI security. In light of these observations, we expect that the proposed generic construction achieves some form of TI security. Formalizing TI security for PEBKS and proving the TI security that the proposed generic construction is expected to satisfy are left as future work.

Acknowledgment: The authors would like to thank Prof. Toshihiro Ohigashi and Prof. Nobuyuki Sugio for their valuable comments on key collisions in HMAC. This work was supported by JSPS KAKENHI Grant Number JP26K02908. This paper was edited for grammar using Copilot.

References

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

[2] 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.

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

[4] Keita Emura. Generic construction of forward secure public key authenticated encryption with keyword search. In *Applied Cryptography and Network Security, Part I*, pages 237–256, 2024.

[5] Keita Emura. Generic construction of dual-server public key authenticated encryption with keyword search. *IET Information Security*, 2025(1), 2025.

[6] Keita Emura, Toshihiro Ohigashi, and Nobuyuki Sugio. Quick draw queries: Lightweight searchable public-key ciphertexts with hidden structures via non-interactive key exchange. *IACR Cryptol. ePrint Arch.*, 2026:703, 2026.

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

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

[9] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

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

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

[12] Tomoki Mori and Keita Emura. Chosen ciphertext and trapdoor secure public key authenticated encryption with keyword search. Cryptology ePrint Archive, Paper 2026/2446, 2026.

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

[14] 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.

[15] Kodai Taiyama, Kosei Sakamoto, Ryoma Ito, Kazuma Taka, and Takanori Isobe. Key collisions on AES and its applications. In *ASIACRYPT, Part VII*, pages 267–300, 2024.

[16] Wenzheng Zhang, Baodong Qin, Xinfeng Dong, and Aikui Tian. Public-key encryption with bidirectional keyword search and its application to encrypted emails. *Computer Standards & Interfaces*, 78:103542, 2021.