Generic Constructions of Integrated PKE and PEKS

Yu Chen* Jiang Zhang† Dongdai Lin* Zhenfeng Zhang†

December 1, 2014

Abstract

In this paper we investigate the topic of integrated public-key encryption (PKE) and public-key encryption with keyword search (PEKS) schemes (PKE-PEKS as shorthand). We first formalize the strongest security notion to date for PKE-PEKS schemes, named joint CCA-security. We then propose two simple constructions of jointly CCA-secure PKE-PEKS schemes from anonymous (hierarchical) identity-based encryption schemes. Besides, we also define the notion of consistency for PKE-PEKS schemes, as well as revisit its related notions (including consistency of PEKS schemes, robustness and collision-freeness of IBE schemes), which may be of independent interest.

*State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China. Yu Chen and Dongdai Lin are supported by the National Natural Science Foundation of China under Grant No. 61303257, 61379141, the Strategic Priority Research Program of CAS under Grant No. XDA06010701, and the National 973 Program of China under Grant No.2011CB302400.

†Trusted Computing and Information Assurance Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China. Jiang Zhang and Zhenfeng Zhang are sponsored by the National Basic Research Program of China under Grant No. 2013CB338003, and the National Natural Science Foundation of China under Grant No. 61170278, 91118006.

1 Introduction

Public-key encryption with keyword search (PEKS) [BDOP04] is an useful primitive which allows one to delegate to a third party the capability of "searching on public-key encrypted data" without impacting privacy. It has found numerous applications in various fields such as the design of spam filter and searchable cloud storage. Boneh et al. [BDOP04] illustrated this mechanism more precisely with the following example. Let (pk, sk) be Alice's public/secret key pair. Alice wishes her email gateway can identify if the incoming email contains a particular keyword w. To do so, Alice sends the gateway a token t_w for w. When Bob sends Alice an email m labeled by a keyword w privately, he first encrypts m under pk using a standard PKE scheme, then "encrypts" w using a PEKS scheme. The overall encrypted email is of the form PKE.$\mathsf{Encrypt}(pk, m)||$PEKS.$\mathsf{Encrypt}(pk, w)$. Upon receiving an encrypted email, the email gateway uses its token t_w to test if the email is labeled by w. Except the testing result, the gateway learns essentially no more information about this email and the keywords.

Chosen-Ciphertext Security for PEKS. The original security notion for PEKS schemes is indistinguishability against chosen-plaintext attack (IND-PEKS-CPA)[1] defined in [BDOP04], where the adversary is only given access to a token oracle. Obviously, this security notion fails to capture the attacks from more powerful adversaries, say, who can inject packets into the network and observe actions taken based on them. For instance, an adversary can send a PEKS ciphertext s to a gateway holding token t_w, then learns some useful information from the routing results. It is thus necessary to consider a stronger security notion for PEKS, namely IND-PEKS-CCA security [ABN10]. In the corresponding security experiment, besides token oracle, the adversary is also given access to a test oracle which can decide if a PEKS ciphertext s encrypts a keyword w.

Integrated PKE and PEKS. PEKS is introduced to provide searchable functionality for PKE. As pointed out by [BDOP04, BSNS06, ZI07, ABN10], due to lack of data retrieval function, a PEKS scheme is only meaningful when coupled with a PKE scheme. For this reason, a full-fledged primitive named integrated PKE and PEKS scheme[2] is suggested, which combines PEKS with PKE to encrypt message m and its keyword w together. It is thus of practical interest to consider PKE and PEKS in a joint sense rather than separately. From here on, we refer to integrated PKE and PEKS scheme as PKE-PEKS scheme. Loosely speaking, we say a PKE-PEKS scheme is jointly secure if it provides data privacy (the confidentiality of m) and keyword privacy (the confidentiality of w) simultaneously. However, constructing a jointly secure PKE-PEKS scheme in a strong sense is not an easy task. As envisioned by [BSNS06, ZI07, ABN10], the straightforward approach of concatenating ciphertexts of PKE and PEKS works fine for joint CPA-security, but is insufficient for joint CCA-security. Note that for PKE, CCA-security is closely related to the notion of non-malleability [DDN00], which captures an adversary's inability that given a ciphertext c to output a different ciphertext c' such that the plaintexts m, m' underlying these two ciphertexts are "meaningfully related". In what follows, we will use the notion of non-malleability to analyze previous constructions and illustrate our approaches on an intuitive level.

[1]In the PEKS setting, "plaintext" in fact means "keyword". We will slightly abuse this term where it is clear from the context.

[2]Integrated PKE and PEKS scheme is also known as combined PKE/PEKS scheme.

1.1 Known Solutions and Their Limitations

Baek et al. [BSNS06] first considered the problem of combining PKE and PEKS in a secure manner. They also gave a joint security notion for PKE-PEKS scheme. However, as pointed out in [ZI07], their notion is not complete since it only considers data privacy, but neglects keyword privacy. Besides, their notion is restricted for that in the challenge stage the target keyword w^* must differ from the target messages m_0^* and m_1^*. In the same paper, two solutions meeting their security notion are proposed in the random oracle model. One is a concrete scheme based on the REACT version of ElGamal [OP01] (serve as the PKE component) and the BDOP-PEKS [BDOP04] (serve as the PEKS component). The main idea is exploring the randomness reuse technique to bind a PKE ciphertext and a PEKS ciphertext together and then using a hash function (act as a MAC) to protect integrity of the overall ciphertext. The other is a generic construction based on a OW-PCA (one-wayness against plaintext checking attack) secure PKE scheme and a PEKS scheme equipped with the same public/secret key pair as the PKE scheme. These requirements may be too stringent for a generic construction.

Be aware of the incompleteness of Baek et al.'s security notion, Zhang and Imai [ZI07] gave another security notion which captures both data privacy (IND-PKE-CCA) and keyword privacy (IND-PEKS-CPA). They also gave a generic construction based on two independent primitives: a tag-based CCA-secure PKE scheme and a CPA-secure PEKS scheme. To bind them together, the PEKS ciphertext s is used as a part of the tag for the PKE scheme. The public/secret key pair of the resulting PKE-PEKS scheme is a direct composition of the two corresponding public/secret key pairs belong to the underlying PKE scheme and PEKS scheme, respectively. However, their construction merely performs one-directional binding, not bidirectional binding. Though the PKE-PEKS ciphertext is non-malleable with respect to the PKE component, it is still malleable with respect to the PEKS component. This explains why their PKE-PEKS construction [ZI07] has IND-PKE-CCA security for data privacy but only IND-PEKS-CPA security for keyword privacy. Besides, in order to achieve joint security (the main principle is to avoid mutual dependency), their PKE-PEKS construction resorts to *key separation strategy*, i.e., using different keys for different cryptographic operations. As a result, the resulting PKE-PEKS construction suffers from double key size, which could be critical in resource-constrained applications.

Abdalla et al. [ABN10, Appendix F] noted that the security notions for PKE-PEKS schemes considered in [BSNS06, ZI07] are not strong enough since the adversary is not given access to a test oracle. Thereby, they introduced a new combined CCA-security notion that the adversary can access to both a decryption oracle and a test oracle.[3] They also sketched how to construct a PKE-PEKS scheme satisfying their security notion in the standard model with the techniques of [DK05], that is, choose a tag-based CCA-secure PKE scheme and a tag-based CCA-secure PEKS scheme[4], then bind the PKE ciphertext and the PEKS ciphertext together by using the verification key of a one-time signature scheme as a common tag, and append a signature of the two ciphertexts under the corresponding signing key. It is worth to note that the underlying tag-based PKE component and tag-based PEKS component are already CCA-secure themselves, which might make the resulting PKE-PEKS scheme a bit expensive. Besides, their construction also suffers from double key size due to the adoption of key separation strategy.

[3]This combined CCA-security notion could be made stronger by giving the adversary access to an additional token oracle. We believe the absence of the token oracle is probably a careless mistake.

[4]In [ABN10], tag-based is referred to as label-based [Sho01].

1.2 Our Contributions

Opposed to key separation strategy, key reuse strategy dictates using the same key for multiple cryptographic operation [HP01]. Similar to the case of joint encryption and signature analyzed in [PSST11], using the same key pair for both PKE and PEKS components in PKE-PEKS scheme can offer us at least two practical advantages: 1) reduce storage requirement for certificates as well as key pairs; 2) reduce the cost of public key certification, and the time taken for public key verification. These savings may be critical in resource-constrained applications. With this motivation in mind, we focus on building jointly CCA-secure PKE-PEKS schemes with single key pair, a problem of which, as we discussed before, there currently exists no satisfactory solution.

We first formalize the notion of joint CCA-security for PKE-PEKS schemes in Section 3.1 by incorporating IND-PKE-CCA security and IND-PEKS-CCA security together in a joint sense. The resulting joint CCA-security is strictly stronger than previous ones [BSNS06, ZI07].

We then present two generic constructions of jointly CCA-secure PKE-PEKS schemes from (hierarchical) IBE schemes[5] in Section 4 and Section 5, respectively. The first construction is based on any two-level HIBE scheme that is anonymous against chosen-plaintext attack (ANO-HIBE-CPA), while the second construction is based on any IBE scheme that is anonymous against chosen-ciphertext attack (ANO-IBE-CCA) and weakly robust. Our two constructions embody the same main idea that applying the BCHK transform [BCHK07] and the BDOP transform [BDOP04, ABC$^+$08] to the underlying (H)IBE scheme in an interweaving manner. Our constructions have several advantages over existing ones. Firstly, our constructions satisfy the strongest security notions for PKE-PEKS so far.[6] Secondly, our constructions use a single key pair for both PKE and PEKS operation, thus enjoy the advantages brought by key reuse as described above. Thirdly, our constructions are mainly derived from one (H)IBE scheme, thus the major operations of PKE and PEKS are actually implemented in one (H)IBE scheme. This enables us to achieve better practical efficiency and compact cryptographic code. So far, a number of anonymous (H)IBE schemes based on various assumptions are known, such as [BF03, BGH07, GPV08] in the random oracle model and [Gen06, DIP10, ABB10, CHKP10, SC11] in the standard model. Instantiating our constructions from these (H)IBE schemes yields jointly CCA-secure PKE-PEKS schemes based on the same assumptions.

As another contribution, in Section 6 we formally define the notion of consistency for PKE-PEKS schemes, as well as revisit its related notions, including consistency of PEKS schemes, robustness and collision-freeness of IBE schemes. For a PEKS scheme derived from the BDOP transform [BDOP04, ABC$^+$08], we prove that it is consistent as long as the underlying IBE scheme is weakly collision-free. Previous result [ABN10] has to assume the underlying IBE scheme to be strong robust. We also present enhanced notions of consistency for PEKS schemes and collision-freeness for IBE schemes. Interestingly, most schemes satisfying the original notions also satisfy the enhanced ones without any modification.

2 Overview of Our Approach

Before sketching our approach, we first describe our definition of joint CCA-security for PKE-PEKS schemes more clearly, then identify the main challenges in constructing jointly CCA-secure PKE-PEKS schemes.

[5]Our constructions also make use of a one-time signature scheme, but it can be derived from one-way functions which in turn implied by CPA-secure encryption.

[6]We believe the two key PKE-PEKS construction sketched in [ABN10] is also jointly CCA-secure.

2.1 Joint CCA-security for PKE-PEKS

The standard security notion for PKE is IND-PKE-CCA security while the strongest security notion so far for PEKS is IND-PEKS-CCA security (cf. definition in Appendix A.1). Towards an as-strong-as-possible joint security, we require that a PKE-PEKS scheme retains IND-PKE-CCA security (with respect to data privacy) in the presence of additional unrestricted token oracle and test oracle, and meanwhile retains IND-PEKS-CCA security (with respect to keyword privacy) in the presence of an additional unrestricted decryption oracle.

2.2 The Main Challenges

Minimizing the Size of Keys. In PKE-PEKS system, the key pair for PKE operation and the key pair for PEKS operation are usually different. Therefore, the overall key pair for PKE-PEKS comprises two key pairs. To settle the problem of key expansion, a natural solution is adopting the *key reuse strategy*, namely using a single key pair for both PKE and PEKS operation. Thus the primary difficulty is to come up with a PKE component and a PEKS component sharing the same key pair. Moreover, in our joint CCA-security notion for PKE-PEKS, the adversary against data privacy has unrestricted access to an additional token oracle and an additional test oracle, while the adversary against keyword privacy has unrestricted access to an additional decryption oracle. It is very likely that the PKE component and the PEKS component badly interacts with one another when they use the same key pair, and thus compromise data privacy or keyword privacy, leading to the corruption of joint security. Therefore, delicate technique need to be used to dismiss such bad interaction due to mutual dependency. Perhaps exactly for this concern, constructions in [ZI07] and [ABN10] followed key separation strategy, which easily avoid possible mutual undermining since the PKE operation and the PEKS operation are essentially independent, but at the expense of doubling key size.

Achieving the Joint CCA-security. To attain joint CCA-security is not an easy job. Intuitively, the PKE-PEKS ciphertext should be non-malleable with respect to both PKE component and PEKS component. Several subtle issues may arise when binding a PKE scheme and a PEKS scheme together to build a joint CCA-secure PKE-PEKS system. One issue is that the binding should be bidirectional, otherwise the resulting PKE-PEKS could not be jointly CCA-secure since the overall ciphertext is malleable with respect to either PKE component or PEKS component. For example, if one binds a CCA-secure PKE scheme and a CCA-secure PEKS scheme in the way like [ZI07], the binding is only unidirectional. The other issue is that both the user and the gateway should be able to check the well-formedness of a PKE-PEKS ciphertext. Note that in PKE-PEKS systems, the decryption capability of the user is more powerful than that of the gateway. It is possible that the gateway is incapable to check the well-formedness of a ciphertext without the knowledge of the secret key (in this case the adversary may violate keyword privacy by feeding the gateway with malicious-generated ciphertext). We provide such an example in Remark 5.2 (the first attempt of using MAC to replace one-time signature).

2.3 Our Approach

We first focus on how to obtain a PKE component and a PEKS component equipped with the same key pair. Our starting point is a CPA-secure and anonymous IBE scheme (see definition in Appendix A.2) consisting of algorithms Setup, $\mathsf{Extract}$, $\mathsf{Encrypt}$, $\mathsf{Decrypt}$. On the one hand, the BCHK transform [BCHK07] shows how to build a CCA-secure PKE from a CPA-secure IBE. Interestingly, [CHK04] indicated that a lite version of the BCHK transform also allows one to create a CCA1-secure (thus of course CPA-secure) PKE with essentially no overhead as compared to the underlying IBE, which works as follows: for key generation one runs Setup

and uses the resulting master public/secret key pair as the public/secret key pair (pk, sk). To encrypt a message m under pk, one chooses a random "identity" c_1, then sets the ciphertext c to be (c_1, c_2), where $c_2 \leftarrow \mathsf{Encrypt}(pk, c_1, m)$. To recover the message from a ciphertext $c = (c_1, c_2)$, one computes $m \leftarrow \mathsf{Decrypt}(dk, c_2)$, where $dk \leftarrow \mathsf{Extract}(sk, c_1)$. The resulting PKE scheme is essentially tag-based where c_1 serves as the tag. On the other hand, the BDOP transform [BDOP04, ABC+08] shows how to build PEKS from anonymous IBE, which works as follows: key generation is performed the same way as the BCHK transform. To generate a token for a keyword w, the user computes $t_w \leftarrow \mathsf{Extract}(sk, w)$, where t_w is actually a decryption key for "identity" w. To encrypt a keyword w under pk, one picks a value s_1 from the message space of the underlying IBE, then computes $s_2 \leftarrow \mathsf{Encrypt}(pk, w, s_1)$ and sets the final ciphertext s to be (s_1, s_2). The gateway determines if s encrypts the keyword w by testing if $s_1 = \mathsf{Decrypt}(t_w, s_2)$. Thus, applying the lite BCHK transform and the BDOP transform to a CPA-secure and anonymous IBE scheme simultaneously yields a PKE scheme and a PEKS scheme with the same key pair. By directly combining them together, we obtain a basic PKE-PEKS construction that uses a single key pair for both PKE operation and PEKS operation. Next, we show how to change it to a jointly CCA-secure one.

Securely Reusing the Keys. The above basic PKE-PEKS construction exactly implement the key reuse strategy. However, as we envisioned before, key reuse is not without its side effect. Observe that in the basic construction, a token for keyword w is exactly a decryption key for "identity" w. This fact enables the adversary to launch the following attack against data privacy: given a PKE-PEKS ciphertext $u = (c, s)$ encrypting message-keyword pair (m, w), where $c = (c_1, \mathsf{Encrypt}(pk, c_1, m))$ is a PKE ciphertext of message m and $s = (s_1, \mathsf{Encrypt}(pk, w, s_1))$ is a PEKS ciphertext of keyword w, it can simply decrypt c by querying the token for keyword "c_1". The reason underlying such attack is the tag space of the PKE component and the keyword space of the PEKS component overlap each other, and consequently the decryption keys and tokens might be meaningfully related. We resolve this problem by using a bit prefix to provide domain separation between decryption keys and tokens, that is, mapping tag c_1 to "identity" $0||c_1$ while mapping keyword w to "identity" $1||w$. Our use of bit prefix trick to partitioning the identity space is reminiscent of prior work in the context of joint security of encryption and signature [PSST11]. The difference is that prior work uses bit prefix to separate private keys and signatures.

Securely Binding PKE and PEKS. After securely reusing the key pair, we then focus on securely binding. In the above enhanced basic construction, both the PKE component and the PEKS component are only CPA-secure, and there is no binding between them. Therefore, it can not resist chosen-ciphertext attack. Our initial attempt is as follows: generate a one-time signature key pair (vk, sk_σ), create a PKE ciphertext c of m under tag vk, create a PEKS ciphertext s of w as before, then sign the concatenate ciphertext $c||s$ using sk and append a signature σ. However, the overall ciphertext is still malleable with respect to the PEKS component. To see this, note that for a ciphertext (vk, c, s, σ), an adversary can simply generate a new signature pair (vk', sk'_σ), then creates a new valid ciphertext $(vk', c'||s, \sigma')$. This is because the PEKS operation is still independent to the PKE operation. We solve this problem by setting vk as a part of input of the PEKS encryption algorithm.

One idea is encoding vk to the "identity" in the PEKS component. More precisely, we set keyword w as the first level identity and use vk as the second level identity. In this way a PKE ciphertext and a PEKS ciphertext are binded together under a common value vk. Coupled with a one-time signature, the final PKE-PEKS ciphertext is non-malleable with respect to both the PKE component and the PEKS component (see Section 4 for details). The advantage of this construction is that the joint CCA-security immediately follows from CPA security and anonymity of the underlying HIBE scheme. The crux is that in the security proof, the use of one-time signature forces the adversary's test queries and decryption queries to differ from the challenge ciphertext in a special way. The disadvantage is that the underlying IBE scheme should be hierarchical to accommodate vk as a second level identity.

The other idea is encoding vk to the "message" in the PEKS component. We stress that in such construction, the overall ciphertext is still malleable with respect to the PEKS ciphertext. This is because the PEKS ciphertext might be malleable with respect to "message" vk if the underlying IBE scheme is merely anonymous against chosen-plaintext attack. Therefore, to guarantee joint CCA-security, the underlying IBE scheme should be at least anonymous against chosen-ciphertext attack (ANO-IBE-CCA). Somewhat surprisingly, ANO-IBE-CCA anonymity is still inadequate. Unlike the case of our first construction, the use of one-time signature here cannot force the adversary's test queries to differ from the challenge ciphertext, since now vk acts as message in the PEKS encryption. Consequently, when reducing keyword privacy to ANO-IBE-CCA anonymity of the underlying IBE, the simulator has to handle the test query directly related to the challenge ciphertext itself. We overcome this obstacle by requiring the underlying IBE to satisfy a natural property, named *weak robustness* [ABN10, FLPQ13] (see definition in Section 6), which stipulates the decryption result of an honestly-generated ciphertext is $\perp$ if the decryption id' does not match the encryption id. We will elaborate this on details in the proof of Lemma 5.2.

2.4 Related Work

Boneh et al. [BDOP04] first proposed the concept of PEKS and discussed the connection between PEKS and IBE, that is, PEKS implies IBE. Also a transform from anonymous IBE to PEKS (known as the BDOP transform) is implicitly presented in [BDOP04] without a formal proof. Shortly afterwards, Abdalla et al. [ABC+08] revised the BDOP transform and provided a formal proof. Their revised transform is known as the new BDOP transform. In the same paper, they also systematically studied the notion of consistency of PEKS.

Subsequent works are divided into two main directions. One direction focuses on proposing new constructions and security models. Di Crescenzo and Saraswat [DS07] constructed a PEKS scheme based on a variant of the Cocks' IBE [Coc01] in the random oracle model. As stated above, Baek et al. [BSNS06], Zhang et al. [ZI07], and Abdalla et al. [ABN10] gave their own security notions and constructions for PKE-PEKS schemes. The other direction focused on extending the basic concept of PEKS. Boneh and Waters [BW07] constructed a PKE scheme that supports conjunctive, subset, and range queries over the keywords. Boneh et al. [BKOI07] showed how to create a PKE scheme that allows PIR (Private Information Retrieval) searching. Fuhr and Paillier [FP07] introduced the concept of decryptable PEKS which allows the user to retrieve the keywords. They also gave a concrete construction in the random oracle model. Hofheinz and Weinreb [HW08] then gave a decryptable PEKS scheme in the standard model.

3 Definitions

Notation and conventions. For a finite set X, we use $x \xleftarrow{R} X$ to denote that x is sampled from X uniformly at random. Throughout the paper, $\kappa \in \mathbb{N}$ denotes the security parameter. We use $\perp$ to represent a distinguished symbol which falls outside the message space, and we use $x||y$ to represent the string concatenation of x and y. A probabilistic polynomial-time (PPT) algorithm $\mathcal{A}$ is a randomized algorithm that runs in time polynomial in κ. If $\mathcal{A}$ is a randomized algorithm, we write $y \leftarrow \mathcal{A}(x_1, \ldots, x_n; r)$ to indicate that $\mathcal{A}$ outputs y on inputs $(x_1, \ldots, x_n)$ and random coins r. Sometimes for brevity, we omit r when it is not necessary to make explicit the random coins $\mathcal{A}$ uses. Particularly, for probabilistic encryption schemes, we say c an honestly-generated ciphertext if it is output by the encryption algorithm with fresh random coins. In this paper, all the security experiments are played between an adversary $\mathcal{A}$ and a challenger $\mathcal{CH}$. The advantage of the adversary is defined over the random coins used by $\mathcal{A}$ and $\mathcal{CH}$.

In our constructions, we will mainly use an (H)IBE scheme and a one-time signature scheme as the underlying primitives (see definitions and security notions of them in Appendix A.2 and A.3, respectively). Let t_g, t_s, and t_v be the maximum times for key generation, signing, and verification respectively in a signature scheme, and let t_b, t_k, t_e, and t_d be the maximum times for key generation, decryption key extraction, encryption, and decryption respectively in an (H)IBE scheme.

3.1 Integrated PKE and PEKS

We begin by reviewing the syntax of a PKE-PEKS scheme [BSNS06, ZI07, ABN10].

Definition 3.1. A PKE-PEKS scheme consists of the following five PPT algorithms:

- $\mathsf{KeyGen}(\kappa)$: take as input a security parameter κ, output a public/secret key pair (pk, sk). Let M be the message space, W be the keyword space, and U be the ciphertext space. We assume pk to be an implicit input for algorithms $\mathsf{Decrypt}$, $\mathsf{TokenGen}$, as well as Test.
- $\mathsf{Encrypt}(pk, m, w)$: take as input a public key pk, a message $m \in M$ and a keyword $w \in W$, output a PKE-PEKS ciphertext u.
- $\mathsf{Decrypt}(sk, u)$: take as input a secret key sk and a PKE-PEKS ciphertext $u \in U$, output the plaintext $m \in M$ or a reject symbol $\perp$ indicating u is invalid.
- $\mathsf{TokenGen}(sk, w)$: take as input a secret key sk and a keyword $w \in W$, output a token t_w.
- $\mathsf{Test}(t_w, u)$: take as input a token t_w for keyword w and a PKE-PEKS ciphertext $u \in U$ which encrypts keyword w' under pk, output 1 if $w' = w$ and 0 otherwise.

Correctness. For any $(pk, sk) \leftarrow \mathsf{KeyGen}(\kappa)$, any $m \in M$, any $w \in W$ and any $t_w \leftarrow \mathsf{TokenGen}(sk, w)$, we have $\mathsf{Decrypt}(sk, \mathsf{Encrypt}(pk, m, w)) = m$ and $\mathsf{Test}(t_w, \mathsf{Encrypt}(pk, m, w)) = 1$.

Consistency. Except for correctness, we also need to consider the notion of consistency for PKE-PEKS schemes. Roughly speaking, we say a PKE-PEKS scheme is consistent if for any $m \in M$ and any $w \neq w'$, we have $\mathsf{Test}(t_{w'}, \mathsf{Encrypt}(pk, m, w)) = 0$. We defer the formal definition of consistency in Section 6.

3.2 Joint CCA-security for PKE-PEKS

We consider data privacy and keyword privacy for PKE-PEKS schemes in joint sense as follows.

Data Privacy for PKE-PEKS schemes is defined via the following experiment.
Setup: $\mathcal{CH}$ runs $\mathsf{KeyGen}(\kappa)$ to generate (pk, sk) and gives $\mathcal{A}$ the public key pk.
Phase 1: $\mathcal{A}$ can adaptively make three types of queries:

- Decryption query $\langle u \rangle$: $\mathcal{CH}$ responds with $m \leftarrow \mathsf{Decrypt}(sk, u)$.
- Token query $\langle w \rangle$: $\mathcal{CH}$ responds with $t_w \leftarrow \mathsf{TokenGen}(sk, w)$.
- Test query $\langle u, w \rangle$: $\mathcal{CH}$ responds with $\mathsf{Test}(u, t_w)$ where $t_w \leftarrow \mathsf{TokenGen}(sk, w)$.

Challenge: $\mathcal{A}$ outputs two messages m_0^* and m_1^* and a keyword w^*. $\mathcal{CH}$ picks a random bit b and sends $u^* \leftarrow \mathsf{Encrypt}(pk, m_b^*, w^*)$ to $\mathcal{A}$ as the challenge ciphertext.

Phase 2: $\mathcal{A}$ can adaptively make more decryption queries $\langle u \rangle$ subject to the restriction that $u \neq u^*$, more token queries $\langle w \rangle$ and more test queries $\langle u, w \rangle$ with no restriction (token query $\langle w^* \rangle$ and test query $\langle u^*, w^* \rangle$ are allowed!). $\mathcal{CH}$ responds the same way as in Phase 1.
Guess: $\mathcal{A}$ outputs its guess b' for b and succeeds if $b' = b$. We denote this event by SuccA and define $\mathcal{A}$'s advantage as $\mathrm{Adv}_{\mathcal{A}}(\kappa) \stackrel{\text{def}}{=} |\Pr[\mathsf{SuccA}] - 1/2|$.

Definition 3.2. A PKE-PEKS scheme is said to has $(t, q_w, q_t, q_d, \epsilon)$ data privacy if for all t-time adversaries making at most q_w token queries, at most q_t test queries, and at most q_d decryption queries have advantage at most ϵ against its data privacy. Informally, we say a PKE-PEKS scheme has data privacy if there is no PPT adversary having non-negligible advantage in κ in the above experiment.

Keyword Privacy for PKE-PEKS schemes is defined via the following experiment.
Setup: $\mathcal{CH}$ runs $\mathsf{KeyGen}(\kappa)$ to generate (pk, sk) and gives $\mathcal{A}$ the public key pk.
Phase 1: Same as that in the experiment for data privacy.
Challenge: $\mathcal{A}$ outputs a message m^* and two keywords w_0^* and w_1^* subject to the restriction that they had not been asked for tokens in Phase 1. $\mathcal{CH}$ picks a random bit b and sends $u^* \leftarrow \mathsf{Encrypt}(pk, m^*, w_b^*)$ to $\mathcal{A}$ as the challenge ciphertext.
Phase 2: $\mathcal{A}$ can adaptively make more token queries $\langle w \rangle$ subject to the restriction that $w \neq w_0^*, w_1^*$, more test queries $\langle u, w \rangle$ subject to the restriction that $(u, w) \neq (u^*, w_0^*), (u^*, w_1^*)$, and more decryption queries with no restriction ($\langle u^* \rangle$ is allowed!).
Guess: $\mathcal{A}$ outputs its guess b' for b and succeeds if $b' = b$. We denote this event by SuccA and define $\mathcal{A}$'s advantage as $\mathrm{Adv}_{\mathcal{A}}(\kappa) \stackrel{\text{def}}{=} |\Pr[\mathsf{SuccA}] - 1/2|$.

Definition 3.3. A PKE-PEKS scheme is said to has $(t, q_w, q_t, q_d, \epsilon)$ keyword privacy if for all t-time adversaries making at most q_w token queries, at most q_t test queries, and at most q_d decryption queries have advantage at most ϵ against its keyword privacy. Informally, we say a PKE-PEKS scheme has keyword privacy if there is no PPT adversary having non-negligible advantage in κ in the above experiment.

Definition 3.4. We say a PKE-PEKS scheme is jointly CCA-secure if it has keyword privacy and data privacy simultaneously.

Our joint CCA-security notion for PKE-PEKS schemes is stronger than previous ones considered in [BSNS06, ZI07], since it embodies both IND-PKE-CCA security and IND-PEKS-CCA security in the joint sense. As analyzed earlier, the PKE-PEKS constructions proposed in [BSNS06, ZI07] are both insecure in our joint CCA-security notion.

4 A Generic Construction from HIBE

We first show how to construct a PKE-PEKS scheme from a two-level HIBE scheme with algorithms (Setup, $\mathsf{Extract}$, Derive, $\mathsf{Encrypt}$, $\mathsf{Decrypt}$). In our construction we also make use of a strong one-time signature scheme OT with algorithms (KeyGen, Sign, Verify) (see definition in Appendix A.3). Without loss of generality, we assume that the message space of the HIBE scheme is $\{0,1\}^n$, the identity space of the HIBE scheme is $\{0,1\}^*$ for each level, and the verification key space of the signature scheme is $\{0,1\}^n$, where $n = n(\kappa)$ is a polynomially bounded function.

$\mathsf{KeyGen}(\kappa)$: Run $(mpk, msk) \leftarrow \text{HIBE}.\mathsf{KeyGen}(\kappa)$, output $(pk, sk) \leftarrow (mpk, msk)$.

$\mathsf{Encrypt}(pk, m, w)$:

1. Run $(vk, sk_\sigma) \leftarrow \text{OT}.\mathsf{KeyGen}(\kappa)$.
2. Encrypt message m using level-1 identity $0||vk$ as $c \leftarrow \text{HIBE}.\mathsf{Encrypt}(pk, 0||vk, m)$.
3. Encrypt vk using level-2 identity $(1||w, vk)$ as $s \leftarrow \text{HIBE}.\mathsf{Encrypt}(pk, (1||w, vk), vk)$.
4. Compute $\sigma \leftarrow \text{OT}.\mathsf{Sign}(sk_\sigma, c||s)$, output the final ciphertext $u = (vk, c, s, \sigma)$.

```
Decrypt$(sk, u)$:
  1. Parse $u$ as $(vk, c, s, \sigma)$.
  2. If OT.Verify$(vk, c||s, \sigma) = 1$, then compute $dk \leftarrow$ HIBE.Extract$(sk, 0||vk)$,
         output $m \leftarrow$ HIBE.Decrypt$(dk, c)$.
     Else output $\perp$.
```

$\mathsf{TokenGen}(sk, w)$: compute $t_1 \leftarrow \text{HIBE}.\mathsf{Extract}(sk, 1||w)$, $t_2 \leftarrow w$, output $t_w = (t_1, t_2)$.

```
Test$(t_w, u)$:
  1. Parse $t_w$ as $(t_1, t_2)$, $u$ as $(vk, c, s, \sigma)$.
  2. If OT.Verify$(vk, c||s, \sigma) = 1$, then compute $dk \leftarrow$ HIBE.Derive$(t_1, (1||t_2, vk))$,
         output 1 if $vk =$ HIBE.Decrypt$(dk, s)$ and 0 otherwise.
     Else output 0.
```

This completes the description of our first construction. The keyword space W of the resulting PKE-PEKS scheme is $\{0,1\}^*$. The correctness of the PKE-PEKS scheme follows readily from that of the underlying HIBE scheme. We will show how to instantiate our first construction from pairings and lattices in Section 7.1 and Section 7.2.

Theorem 4.1. *The above PKE-PEKS construction is jointly CCA-secure, provided that the HIBE scheme is IND-HIBE-CPA secure in selective-identity sense and ANO-HIBE-CPA anonymous at level one and the signature scheme is one-time sEUF-CMA secure.*

Proof. We first fix some notations and definitions. A PKE-PEKS ciphertext $u = (vk, c, s, \sigma)$ is said to be valid if $\text{OT}.\mathsf{Verify}(vk, c||s, \sigma) = 1$. Let $u^* = (vk^*, c^*, s^*, \sigma^*)$ denote the challenge PKE-PEKS ciphertext received by $\mathcal{A}$. We prove this theorem by the following two lemmas.

Lemma 4.1. *Assume the HIBE scheme is $(t_1, q_{k_1}, \epsilon_1)$ IND-HIBE-CPA secure in selective-identity sense and the signature scheme is $(t_3, 1, \epsilon_3)$ sEUF-CMA secure. Then the PKE-PEKS scheme has $(t, q_w, q_t, q_d, \epsilon)$ data privacy such that*

$$\epsilon \geq \epsilon_1 + \tfrac{1}{2}\epsilon_3, q_w + q_t + q_d \leq q_{k_1},$$
$$t \leq \min\{t_3 - t_b - q_w t_k - (q_t + q_d)(t_k + t_v + t_d) - 2t_e, t_1 - t_g - (q_t + q_d)(t_v + t_d) - t_e - t_s\}.$$

Proof. Suppose there is an adversary $\mathcal{A}$ that has advantage $\text{Adv}_\mathcal{A}$ against the data privacy of the PKE-PEKS construction in running time t. Let Forge denote the event that $\mathcal{A}$ submits a valid ciphertext (vk^*, c, s, σ) to the decryption oracle (we may assume that vk^* is chosen at the outset of the experiment so this event is well-defined even before $\mathcal{A}$ is given the challenge ciphertext.) We prove the following claims:

Claim 4.1. $\Pr[\mathsf{Forge}] \leq \epsilon_3$.

Claim 4.2. $|\Pr[\mathsf{SuccA} \wedge \overline{\mathsf{Forge}}] + \frac{1}{2}\Pr[\mathsf{Forge}] - \frac{1}{2}| \leq \epsilon_1$.

Proof of Claim 4.1 We use $\mathcal{A}$ to construct a forger $\mathcal{F}$ against sEUF-CMA security of the signature scheme OT. $\mathcal{F}$ simulates $\mathcal{A}$'s challenger in the data-privacy experiment for PKE-PEKS as follows: given input κ and the verification key vk^* (output by $\text{OT}.\mathsf{KeyGen}(\kappa)$), $\mathcal{F}$ first runs $\text{HIBE}.\mathsf{KeyGen}(\kappa)$ to obtain (pk, sk), and then runs $\mathcal{A}(pk)$. Note that $\mathcal{F}$ can answer any token queries, test queries, decryption queries with sk. If $\mathcal{A}$ happens to submit a valid ciphertext (vk^*, c, s, σ) to decryption oracle before requesting the challenge ciphertext, then $\mathcal{F}$ simply outputs $(c||s, \sigma)$ to its own challenger and aborts. Otherwise, when $\mathcal{A}$ outputs two messages m_0^* and m_1^* and a keyword w^* where it wants to be challenged on, $\mathcal{F}$ proceeds as follows: it chooses a random bit b, computes $c^* \leftarrow \text{HIBE}.\mathsf{Encrypt}(pk, 0||vk^*, m_b^*)$, $s^* \leftarrow \text{HIBE}.\mathsf{Encrypt}(pk, (1||w^*, vk^*), vk^*)$, then obtains a signature σ^* of message $c^*||s^*$ via calling its signing oracle. Finally, $\mathcal{F}$ sends the challenge ciphertext $(vk^*, c^*, s^*, \sigma^*)$ to $\mathcal{A}$. If $\mathcal{A}$ issues a valid decryption query (vk^*, c, s, σ) in Phase 2 (note that in this case we must have $(c, s, \sigma) \neq (c^*, s^*, \sigma^*)$), then $\mathcal{F}$ simply outputs $(c||s, \sigma)$ as its forgery. It is easy to see that $\mathcal{F}$'s success probability is exactly $\Pr[\mathsf{Forge}]$. Security of OT implies this claim. $\square$

Proof of Claim 4.2 We use $\mathcal{A}$ to construct an adversary $\mathcal{D}$ against selective-identity IND-HIBE-CPA security of the HIBE scheme. $\mathcal{D}$ simulates $\mathcal{A}$'s challenger in the data-privacy experiment for PKE-PEKS as follows:

Setup: $\mathcal{D}$ runs $\text{OT}.\mathsf{KeyGen}(\kappa)$ to generate (vk^*, sk_σ^*), commits $id^* = 0||vk^*$ to its own challenger as the target identity, and is then given mpk of the HIBE scheme. $\mathcal{D}$ sets $pk = mpk$ and runs $\mathcal{A}(pk)$.

Phase 1: Upon receiving token queries, test queries, and decryption queries issued by $\mathcal{A}$, $\mathcal{D}$ responds as follows:

- Token query $\langle w \rangle$: $\mathcal{D}$ issues decryption key extraction query $\langle 1||w \rangle$ to its own challenger and forwards the reply with w to $\mathcal{A}$.
- Test query $\langle u, w \rangle$: $\mathcal{D}$ first obtains a token t_w for w in the same way as it answers the token query $\langle w \rangle$, then responds with $\mathsf{Test}(t_w, u)$.
- Decryption query $\langle u \rangle$: $\mathcal{D}$ parses u as (vk, c, s, σ). If $\text{OT}.\mathsf{Verify}(vk, c||s, \sigma) = 0$ then $\mathcal{D}$ rejects this decryption query with $\perp$. Otherwise, $\mathcal{D}$ proceeds as follows:
  - Case $vk = vk^*$: event Forge occurs, $\mathcal{D}$ aborts and outputs a random bit.
  - Case $vk \neq vk^*$: $\mathcal{D}$ obtains a decryption key dk for "identity" $0||vk$ (by issuing the decryption key extraction query $\langle 0||vk \rangle$ to its own challenger), then responds with $\text{HIBE}.\mathsf{Decrypt}(dk, c)$.

Challenge: $\mathcal{A}$ outputs two messages m_0^* and m_1^* and a keyword w^* where it wants to be challenged on. $\mathcal{D}$ proceeds as follows: it submits m_0^* and m_1^* to its own challenger, and is then given $c^* \leftarrow \text{HIBE}.\mathsf{Encrypt}(pk, 0||vk^*, m_b^*)$, where b is a random bit chosen by $\mathcal{D}$'s challenger. $\mathcal{D}$ then computes $s^* \leftarrow \text{HIBE}.\mathsf{Encrypt}(pk, (1||w, vk^*), vk^*)$, $\sigma^* \leftarrow \text{OT}.\mathsf{Sign}(sk_\sigma^*, c^*||s^*)$. Finally, $\mathcal{D}$ sends $u^* = (vk^*, c^*, s^*, \sigma^*)$ to $\mathcal{A}$ as the challenge ciphertext.

Phase 2: $\mathcal{A}$ can adaptively make more decryption queries, token queries, and test queries. Note that $\mathcal{D}$ can answer all the token queries and test queries correctly since decryption key extraction queries of the form $\langle 1||w \rangle$ are always permitted by $\mathcal{D}$'s challenger. $\mathcal{D}$ proceeds the decryption queries in the same way as in Phase 1 except that it will directly reject $\langle u^* \rangle$ with $\perp$.

Guess: As soon as $\mathcal{A}$ outputs its guess b' for b, $\mathcal{D}$ outputs b' to its own challenger.

Note that $\mathcal{D}$ represents a legal strategy for attacking selective-identity IND-HIBE-CPA security of the HIBE scheme. Furthermore, $\mathcal{D}$ provides a perfect simulation for $\mathcal{A}$ conditioned on the event Forge never occurs. Let SuccD denote the event of $\mathcal{D}$ outputting the correct bit in the selective-identity IND-HIBE-CPA security experiment. It is easy to see that:

$$\left|\Pr[\mathsf{SuccD}] - \tfrac{1}{2}\right| = \left|\Pr[\mathsf{SuccA} \wedge \overline{\mathsf{Forge}}] + \tfrac{1}{2}\Pr[\mathsf{Forge}] - \tfrac{1}{2}\right|.$$

The assumed security of the HIBE implies this claim. $\square$

Based on Claim 4.1 and Claim 4.2, we can derive a concrete bound of ϵ. Note that:

$$\begin{aligned} |\Pr[\mathsf{SuccA}] - \tfrac{1}{2}| &\leq |\Pr[\mathsf{SuccA} \wedge \mathsf{Forge}] - \tfrac{1}{2}\Pr[\mathsf{Forge}]| + |\Pr[\mathsf{SuccA} \wedge \overline{\mathsf{Forge}}] + \tfrac{1}{2}\Pr[\mathsf{Forge}] - \tfrac{1}{2}| \\ &\leq \tfrac{1}{2}\Pr[\mathsf{Forge}] + |\Pr[\mathsf{SuccA} \wedge \overline{\mathsf{Forge}}] + \tfrac{1}{2}\Pr[\mathsf{Forge}] - \tfrac{1}{2}|, \end{aligned}$$

we have $\epsilon \geq \frac{1}{2}\epsilon_3 + \epsilon_1$. For the time complexity, it is easy to check that the running time of $\mathcal{F}$ is at most $t + t_b + q_w t_k + (q_t + q_d)(t_k + t_v + t_d) + 2t_e$, and the running time of $\mathcal{D}$ is at most $t + t_g + (q_t + q_d)(t_v + t_d) + t_e + t_s$. During the simulation, $\mathcal{D}$ asks at most $q_w + q_t + q_d$ decryption key extraction queries. This proves Lemma 4.1. ■

Lemma 4.2. *Assume the HIBE scheme is $(t_2, q_{k_2}, \epsilon_2)$ ANO-HIBE-CPA anonymous at level 1 and the signature scheme is $(t_3, 1, \epsilon_3)$ sEUF-CMA secure. Then the PKE-PEKS scheme has $(t, q_w, q_t, q_d, \epsilon)$ keyword privacy such that:*

$$\begin{aligned}
&\epsilon \geq \epsilon_2 + \tfrac{1}{2}\epsilon_3, q_w + q_t + q_d \leq q_{k_2},\\
&t \leq \min\{t_3 - t_b - q_w t_k - (q_t + q_d)(t_k + t_v + t_d) - 2t_e, t_2 - t_g - (q_t + q_d)(t_v + t_d) - t_e - t_s\}.
\end{aligned}$$

Proof. Suppose there is an adversary $\mathcal{A}$ has advantage $\mathrm{Adv}_{\mathcal{A}}$ against the keyword privacy of the PKE-PEKS construction in running time t. Let Forge denote the event that $\mathcal{A}$ submits a test query $\langle u, w\rangle$ in Phase 2 where $u = (vk^*, c, s, \sigma)$ is a valid PKE-PEKS ciphertext and w is either w_0^* or w_1^*. We prove the following claims:

Claim 4.3. $\Pr[\mathsf{Forge}] \leq \epsilon_3$.

Claim 4.4. $|\Pr[\mathsf{SuccA} \wedge \overline{\mathsf{Forge}}] + \frac{1}{2}\Pr[\mathsf{Forge}] - \frac{1}{2}| \leq \epsilon_2$.

Proof of Claim 4.3 We use $\mathcal{A}$ to construct a forger $\mathcal{F}$ against sEUF-CMA security of the signature scheme OT. $\mathcal{F}$ simulates $\mathcal{A}$'s challenger in the keyword-privacy experiment for PKE-PEKS as follows: given input κ and the verification key vk^* (output by OT.$\mathsf{KeyGen}(\kappa)$), $\mathcal{F}$ first runs HIBE.$\mathsf{KeyGen}(\kappa)$ to obtain (pk, sk), and then runs $\mathcal{A}(pk)$. Note that $\mathcal{F}$ can answer any token queries, test queries, decryption queries with sk. When $\mathcal{A}$ outputs two keywords w_0^* and w_1^* and a message m^* where it wants to be challenged on, $\mathcal{F}$ proceeds as follows: it chooses a random bit b, computes $c^* \leftarrow$ HIBE.$\mathsf{Encrypt}(pk, 0||vk^*, m^*)$, $s^* \leftarrow$ HIBE.$\mathsf{Encrypt}(pk, (1||w_b^*, vk^*), vk^*)$, then obtains a signature σ^* of message $c^*||s^*$ via calling its signing oracle. Finally, $\mathcal{F}$ sends the challenge ciphertext $(vk^*, c^*, s^*, \sigma^*)$ to $\mathcal{A}$. If $\mathcal{A}$ submits a valid test query $\langle u, w\rangle$ in Phase 2 where $u = (vk^*, c, s, \sigma)$ and w is either w_0^* or w_1^* (now for a legal test query we must have $(c, s, \sigma) \neq (c^*, s^*, \sigma^*)$), $\mathcal{F}$ simply outputs $(c||s, \sigma)$ as its forgery. It is easy to see that $\mathcal{F}$'s success probability is exactly $\Pr[\mathsf{Forge}]$. Security of OT implies this claim. □

Proof of Claim 4.4 We use $\mathcal{A}$ to construct an adversary $\mathcal{D}$ against ANO-HIBE-CPA anonymity at level 1 of the HIBE scheme. $\mathcal{D}$ simulates $\mathcal{A}$'s challenger in the keyword-privacy experiment for PKE-PEKS as follows:

Setup: $\mathcal{D}$ is given mpk of the HIBE scheme. $\mathcal{D}$ sets $pk = mpk$ and runs $\mathcal{A}(pk)$.

Phase 1: Upon receiving token queries, test queries, and decryption queries issued by $\mathcal{A}$, $\mathcal{D}$ responds as follows:

- Token query $\langle w\rangle$: $\mathcal{D}$ issues decryption key query $\langle 1||w\rangle$ to its own challenger and forwards the reply with w to $\mathcal{A}$.
- Test query $\langle u, w\rangle$: $\mathcal{D}$ obtains a token t_w for w in the same way as it answers the token query $\langle w\rangle$, then responds with $\mathsf{Test}(t_w, u)$.
- Decryption query $\langle u\rangle$: $\mathcal{D}$ parses u as (vk, c, s, σ). If OT.$\mathsf{Verify}(vk, c||s, \sigma) = 0$, then $\mathcal{D}$ rejects this decryption query with $\perp$. If not, $\mathcal{D}$ obtains a decryption key dk for "identity" $0||vk$ (by issuing decryption key extraction query $\langle 0||vk\rangle$ to its own challenger), then responds with HIBE.$\mathsf{Decrypt}(dk, c)$.

Challenge: When $\mathcal{A}$ outputs a message m^*, two keywords w_0^* and w_1^* where it wants to be challenged on, $\mathcal{D}$ proceeds as follows:

1. Run $(vk^*, sk_\sigma^*) \leftarrow$ OT.$\mathsf{KeyGen}(\kappa)$, set $id^* = 0||vk^*$.

2. Compute $c^* \leftarrow \text{HIBE}.\mathsf{Encrypt}(pk, id^*, m^*)$.
3. Submit vk^* (as the message) and two target identities $(1||w_0^*, vk^*)$ and $(1||w_1^*, vk^*)$ to its own challenger, and get the encryption s^* of vk^* under identity $(1||w_b^*, vk^*)$, where b is a random bit chosen by $\mathcal{D}$'s challenger.
4. Compute $\sigma^* \leftarrow \text{OT}.\mathsf{Sign}(sk_\sigma^*, c^*||s^*)$.
5. Send $u^* = (vk^*, c^*, s^*, \sigma^*)$ to $\mathcal{A}$ as the challenge ciphertext.

Phase 2: $\mathcal{A}$ can adaptively make more token queries, test queries, and decryption queries in Phase 2, $\mathcal{D}$ responds as follows:

- Token query $\langle w \rangle$: as long as $w \neq w_0^*, w_1^*$, $\mathcal{D}$ can always issue decryption key extraction query for identity $1||w$, then sends the reply with w to $\mathcal{A}$.
- Test query $\langle u, w \rangle$: test queries $\langle u^*, w_0^* \rangle$ and $\langle u^*, w_1^* \rangle$ will be rejected. $\mathcal{D}$ parses u as (vk, c, s, σ), first checks if $\text{OT}.\mathsf{Verify}(vk, c||s, \sigma) = 0$. If so, $\mathcal{D}$ returns 0. If not, when w is not w_0^* or w_1^*, $\mathcal{D}$ obtains a token t_w for w in the same way as it answers the token query $\langle w \rangle$ and then responds with $\mathsf{Test}(t_w, u)$. Otherwise, $\mathcal{D}$ proceeds as follows:
  - Case $vk = vk^*$: event Forge occurs (now w is either w_0^* or w_1^*, thus for a legal query we must have $u \neq u^*$), $\mathcal{D}$ aborts and outputs a random bit.
  - Case $vk \neq vk^*$: $\mathcal{D}$ obtains a decryption key dk for identity $(1||w, vk)$, returns 1 if $vk = \text{HIBE}.\mathsf{Decrypt}(dk, s)$ and 0 otherwise.
- Decryption query $\langle u \rangle$: $\mathcal{D}$ responds to the decryption queries as it did in Phase 1. Since decryption key extraction queries of the form $\langle 0||vk \rangle$ are always permitted, $\mathcal{D}$ can answer all the decryption queries correctly.

Guess. As soon as $\mathcal{A}$ outputs its guess b' for b, $\mathcal{D}$ outputs b' to its own challenger.
Note that $\mathcal{D}$ represents a legal strategy for attacking ANO-HIBE-CPA anonymity of the underlying HIBE scheme. Furthermore, $\mathcal{D}$ provides a perfect simulation for $\mathcal{A}$ in the keyword privacy experiment of PKE-PEKS conditioned on Forge never occurs. Let SuccD denote the event of $\mathcal{D}$ outputting the correct bit in the ANO-HIBE-CPA anonymity experiment for the HIBE scheme. It is easy to see that:

$$|\Pr[\mathsf{SuccD}] - \tfrac{1}{2}| = |\Pr[\mathsf{SuccA} \wedge \overline{\mathsf{Forge}}] + \tfrac{1}{2}\Pr[\mathsf{Forge}] - \tfrac{1}{2}|.$$

The assumed anonymity of the HIBE implies this claim. □

Based on Claim 4.3 and Claim 4.4, we can derive a concrete bound of ϵ. Note that:

$$\begin{aligned} |\Pr[\mathsf{SuccA}] - \tfrac{1}{2}| &\leq |\Pr[\mathsf{SuccA} \wedge \mathsf{Forge}] - \tfrac{1}{2}\Pr[\mathsf{Forge}]| + |\Pr[\mathsf{SuccA} \wedge \overline{\mathsf{Forge}}] + \tfrac{1}{2}\Pr[\mathsf{Forge}] - \tfrac{1}{2}| \\ &\leq \tfrac{1}{2}\Pr[\mathsf{Forge}] + |\Pr[\mathsf{SuccA} \wedge \overline{\mathsf{Forge}}] + \tfrac{1}{2}\Pr[\mathsf{Forge}] - \tfrac{1}{2}|, \end{aligned}$$

we have $\epsilon \geq \frac{1}{2}\epsilon_3 + \epsilon_2$. For the time complexity, it is easy to check that the running time of $\mathcal{F}$ is at most $t + t_b + q_w t_k + (q_t + q_d)(t_k + t_v + t_d) + 2t_e$, and the running time of $\mathcal{D}$ is at most $t + t_g + (q_t + q_d)(t_v + t_d) + t_e + t_s$. During the simulation, $\mathcal{D}$ asks at most $q_w + q_t + q_d$ decryption key extraction queries. This proves Lemma 4.2. ■

Theorem 4.1 immediately follows from Lemma 4.1 and Lemma 4.2. ■

5 A Generic Construction from IBE

We now show how to construct a PKE-PEKS scheme from an IBE scheme with algorithms (Setup, $\mathsf{Extract}$, $\mathsf{Encrypt}$, $\mathsf{Decrypt}$). As in the first construction, we also make use of a strong one-time signature scheme OT, and assume that the message space of the IBE scheme is $\{0,1\}^n$, the identity space of the IBE scheme is $\{0,1\}^*$, and the verification key space of the signature scheme is $\{0,1\}^n$.

$\mathsf{KeyGen}(\kappa)$: Run $(mpk, msk) \leftarrow \text{IBE}.\mathsf{KeyGen}(\kappa)$, output $(pk, sk) \leftarrow (mpk, msk)$.

$\mathsf{Encrypt}(pk, m, w)$:

```
1. Run $(vk, sk_\sigma) \leftarrow \text{OT}.\mathsf{KeyGen}(\kappa)$.
2. Encrypt message $m$ using identity $0||vk$ as $c \leftarrow \text{IBE}.\mathsf{Encrypt}(pk, 0||vk, m)$.
3. Encrypt keyword $w$ using identity $1||w$ as $s \leftarrow \text{IBE}.\mathsf{Encrypt}(pk, 1||w, vk)$.
4. Compute $\sigma \leftarrow \text{OT}.\mathsf{Sign}(sk_\sigma, c||s)$, output the final ciphertext $u = (vk, c, s, \sigma)$.
```

$\mathsf{Decrypt}(sk, u)$:

```
1. Parse $u$ as $(vk, c, s, \sigma)$.
2. If $\text{OT}.\mathsf{Verify}(vk, c||s, \sigma) = 1$,
       then compute $dk \leftarrow \text{IBE}.\mathsf{Extract}(sk, 0||vk)$, output $m \leftarrow \text{IBE}.\mathsf{Decrypt}(dk, c)$.
   Else output $\perp$.
```

$\mathsf{TokenGen}(sk, w)$: output $t_w \leftarrow \text{IBE}.\mathsf{Extract}(sk, 1||w)$.

$\mathsf{Test}(t_w, u)$:

```
1. Parse $u$ as $(vk, c, s, \sigma)$.
2. If $\text{OT}.\mathsf{Verify}(vk, c||s, \sigma) = 1$,
       output 1 if $vk = \text{IBE}.\mathsf{Decrypt}(t_w, s)$ and 0 otherwise.
   Else output 0.
```

This completes the description of our second construction. Similar to our first construction, the keyword space W of the resulting PKE-PEKS scheme is $\{0,1\}^*$. The correctness of this construction follows readily from that of the IBE scheme. We will show how to instantiate our second construction from pairing in Section 7.3.

Theorem 5.1. *The above PKE-PEKS construction is jointly CCA-secure, provided that the IBE scheme is IND-IBE-CPA secure in selective-identity sense and ANO-IBE-CCA anonymous and weakly robust and the signature scheme is one-time sEUF-CMA secure.*

Proof. We first fix some notations and definitions. A PKE-PEKS ciphertext $u = (vk, c, s, \sigma)$ is said to be valid if $\text{OT}.\mathsf{Verify}(vk, c||s,) = 1$. Let $u^* = (vk^*, c^*, s^*, \sigma^*)$ denote the challenge PKE-PEKS ciphertext received by $\mathcal{A}$. We prove this theorem by the following two lemmas.

Lemma 5.1. *Assume the IBE scheme is (t_1, q_{k_1}, ϵ) IND-IBE-CPA secure in selective-identity sense and the signature scheme is $(t_3, 1, \epsilon_3)$ sEUF-CMA secure. Then the PKE-PEKS scheme has $(t, q_w, q_t, q_d, \epsilon)$ data privacy such that*

$$\epsilon \geq \epsilon_1 + \frac{1}{2}\epsilon_3, q_w + q_t + q_d \leq q_{k_1},$$
$$t \leq \min\{t_3 - t_b - q_w t_k - (q_t + q_d)(t_k + t_v + t_d) - 2t_e, t_1 - t_g - (q_t + q_d)(t_v + t_d) - t_e - t_s\}.$$

Proof. Suppose there is an adversary $\mathcal{A}$ has advantage $\text{Adv}_\mathcal{A}$ against the data privacy of the PKE-PEKS construction in running time t. Let Forge denote the event that $\mathcal{A}$ submits a valid PKE-PEKS ciphertext (vk^*, c, s, σ) to the decryption oracle. We prove the following claims:

Claim 5.1. $\Pr[\mathsf{Forge}] \leq \epsilon_3$.

Claim 5.2. $|\Pr[\mathsf{SuccA} \wedge \overline{\mathsf{Forge}}] + \frac{1}{2}\Pr[\mathsf{Forge}] - \frac{1}{2}| \leq \epsilon_1$.

Proof of Claim 5.1 We use $\mathcal{A}$ to construct a forger $\mathcal{F}$ against sEUF-CMA security of the signature scheme OT. $\mathcal{F}$ simulates $\mathcal{A}$'s challenger in the data-privacy experiment for PKE-PEKS as follows: given input κ and the verification key vk^* (output by OT.$\mathsf{KeyGen}(\kappa)$), $\mathcal{F}$ first runs IBE.$\mathsf{KeyGen}(\kappa)$ to obtain (pk, sk), and then runs $\mathcal{A}(pk)$. Note that $\mathcal{F}$ can answer any token queries, test queries, decryption queries with sk. If $\mathcal{A}$ happens to submit a valid ciphertext (vk^*, c, s, σ) to decryption oracle before requesting the challenge ciphertext, then $\mathcal{F}$ simply outputs $(c||s, \sigma)$ to its own challenger and stops. Otherwise, when $\mathcal{A}$ outputs two messages m_0^* and m_1^* and a keyword w^* where it wants to be challenged on, $\mathcal{F}$ proceeds as follows: it chooses a random bit b, computes $c^* \leftarrow$ IBE.$\mathsf{Encrypt}(pk, 0||vk^*, m_b^*)$, $s^* \leftarrow$ IBE.$\mathsf{Encrypt}(pk, 1||w^*, vk^*)$, then obtains a signature σ^* of message $c^*||s^*$ via calling its signing oracle. Finally, $\mathcal{F}$ sends the challenge ciphertext $(vk^*, c^*, s^*, \sigma^*)$ to $\mathcal{A}$. If $\mathcal{A}$ submits a valid decryption query (vk^*, c, s, σ) note that we must have $(c, s, \sigma) \neq (c^*, s^*, \sigma^*)$. In this case, $\mathcal{F}$ simply outputs $(c||s, \sigma)$ as its forgery. It is easy to see that $\mathcal{F}$'s success probability is exactly $\Pr[\mathsf{Forge}]$. Security of OT implies this claim. □

Proof of Claim 5.2 We use $\mathcal{A}$ to construct an adversary $\mathcal{D}$ against IND-IBE-CPA security in selective-identity sense of the IBE scheme. $\mathcal{D}$ simulates $\mathcal{A}$'s challenger in the data-privacy experiment for PKE-PEKS as follows:

Setup: $\mathcal{D}$ runs OT.$\mathsf{KeyGen}(\kappa)$ to generate (vk^*, sk_σ^*), commits $id^* = 0||vk^*$ to its own challenger as the target identity, and is then given mpk of the IBE scheme. $\mathcal{D}$ sets $pk = mpk$ and runs $\mathcal{A}(pk)$.

Phase 1: Upon receiving token queries, test queries, and decryption queries issued by $\mathcal{A}$, $\mathcal{D}$ responds as follows:

- Token query $\langle w \rangle$: $\mathcal{D}$ issues decryption key query $\langle 1||w \rangle$ to its own challenger and forwards the reply to $\mathcal{A}$.
- Test query $\langle u, w \rangle$: $\mathcal{D}$ first obtains a token t_w for w in the same way as it answers the token query $\langle w \rangle$, then responds with $\mathsf{Test}(t_w, u)$.
- Decryption query $\langle u \rangle$: $\mathcal{D}$ parses u as (vk, c, s, σ). If OT.$\mathsf{Verify}(vk, c||s, \sigma) = 0$ then $\mathcal{D}$ rejects this decryption query with $\perp$. Otherwise $\mathcal{D}$ proceeds as follows:
  - Case $vk = vk^*$: event Forge occurs, $\mathcal{D}$ aborts and outputs a random bit.
  - Case $vk \neq vk^*$: $\mathcal{D}$ obtains a decryption key dk for "identity" $0||vk$ (by issuing decryption key extraction query $\langle 0||vk \rangle$ to its own challenger), then responds with IBE.$\mathsf{Decrypt}(dk, c)$.

Challenge: $\mathcal{A}$ outputs two messages m_0^* and m_1^* and a keyword w^* where it wants to be challenged on. $\mathcal{D}$ proceeds as follows: it submits m_0^* and m_1^* to its own challenger, and is then given $c^* \leftarrow$ IBE.$\mathsf{Encrypt}(pk, 0||vk^*, m_b^*)$, where b is a random bit chosen by $\mathcal{D}$'s challenger. $\mathcal{D}$ then compute $s^* \leftarrow$ IBE.$\mathsf{Encrypt}(pk, 1||w^*, vk^*)$, $\sigma^* \leftarrow$ OT.$\mathsf{Sign}(sk_\sigma^*, c^*||s^*)$. Finally, $\mathcal{D}$ sends $u^* = (vk^*, c^*, s^*, \sigma^*)$ to $\mathcal{A}$ as the challenge ciphertext.

Phase 2: $\mathcal{A}$ can adaptively make more token queries, test queries, and decryption queries. Note that $\mathcal{D}$ can answer all the token queries and test queries correctly since decryption key queries of the form $\langle 1||w \rangle$ are always permitted by $\mathcal{D}$'s challenger. $\mathcal{D}$ proceeds decryption queries in the same way as in Phase 1 except that it will directly reject $\langle u^* \rangle$ with $\perp$.

Guess: As soon as $\mathcal{A}$ outputs its guess b' for b. $\mathcal{D}$ outputs b' to its own challenger.

Note that $\mathcal{D}$ represents a legal strategy for attacking the selective-identity IND-IBE-CPA security of the underlying IBE scheme. Furthermore, $\mathcal{D}$ provides a perfect simulation for $\mathcal{A}$

conditioned on the event Forge never occurs. Let SuccD denote the event of $\mathcal{D}$ outputting the correct bit in the selective-identity IND-IBE-CPA security experiment. It is easy to see that:

$$|\Pr[\mathsf{SuccD}] - \tfrac{1}{2}| = |\Pr[\mathsf{SuccA} \wedge \overline{\mathsf{Forge}}] + \tfrac{1}{2}\Pr[\mathsf{Forge}] - \tfrac{1}{2}|.$$

The assumed security of the underlying IBE implies this claim. □

Based on Claim 5.1 and Claim 5.2, we can derive a concrete bound of ϵ. Note that:

$$\begin{aligned} |\Pr[\mathsf{SuccA}] - \tfrac{1}{2}| &\leq |\Pr[\mathsf{SuccA} \wedge \mathsf{Forge}] - \tfrac{1}{2}\Pr[\mathsf{Forge}]| + |\Pr[\mathsf{SuccA} \wedge \overline{\mathsf{Forge}}] + \tfrac{1}{2}\Pr[\mathsf{Forge}] - \tfrac{1}{2}| \\ &\leq \tfrac{1}{2}\Pr[\mathsf{Forge}] + |\Pr[\mathsf{SuccA} \wedge \overline{\mathsf{Forge}}] + \tfrac{1}{2}\Pr[\mathsf{Forge}] - \tfrac{1}{2}|, \end{aligned}$$

we have $\epsilon \leq \frac{1}{2}\epsilon_3 + \epsilon_1$. For the time complexity, it is easy to check that the running time of $\mathcal{F}$ is at most $t + t_b + q_w t_k + (q_t + q_d)(t_k + t_v + t_d) + 2t_e$, and the running time of $\mathcal{D}$ is at most $t + t_g + (q_t + q_d)(t_v + t_d) + t_e + t_s$. During the simulation, $\mathcal{D}$ asks at most $q_w + q_t + q_d$ decryption key extraction queries. This proves Lemma 5.1. ■

Lemma 5.2. *Assume the IBE scheme is $(t_2, q_{k_2}, q_{d_2}, \epsilon_2)$ ANO-IBE-CCA anonymous and also $(t_4, q_{k_4}, \epsilon_4)$ weakly robust, and the signature scheme is $(t_3, 1, \epsilon_3)$ sEUF-CMA secure. Then the PKE-PEKS scheme has $(t, q_w, q_t, q_d, \epsilon)$ keyword privacy such that:*

$$\begin{aligned} &\epsilon \geq \epsilon_2 + \epsilon_4 + \tfrac{1}{2}\epsilon_3, q_w \leq q_{k_2}, q_t + q_d \leq q_{d_2}, q_w + q_t + q_d \leq q_{k_4}, \\ &t \leq \min\{t_3 - t_b - q_w t_k - (q_t + q_d)(t_k + t_v + t_d) - 2t_e, t_4 - t_g, t_2 - (q_w + q_t)t_k\}. \end{aligned}$$

Proof. Suppose there is an adversary $\mathcal{A}$ has advantage $\mathrm{Adv}_{\mathcal{A}}$ against the keyword privacy of the PKE-PEKS construction in running time t. Let m^*, w_0^* and w_1^* be the target message-keyword output by $\mathcal{A}$. Let $(vk^*, c^*, s^*, \sigma^*)$ be the challenge PKE-PEKS ciphertext of (m^*, w_b^*). Let Forge denote the event that $\mathcal{A}$ submits a valid test query $\langle u, w\rangle$ in Phase 2 where $u = (vk^*, c, s, \sigma)$ and w is either w_0^* or w_1^*. Let Break denote the event that the decryption of s^* under identity $1||w_b^*$ is not $\perp$. We prove the following claims:

Claim 5.3. $\Pr[\mathsf{Forge}] \leq \epsilon_3$.

Claim 5.4. $\Pr[\mathsf{Break}] \leq \epsilon_4$.

Claim 5.5. $|\Pr[\mathsf{SuccA} \wedge \overline{\mathsf{Forge} \vee \mathsf{Break}}] + \frac{1}{2}\Pr[\mathsf{Forge}] - \frac{1}{2}| \leq \epsilon_2$.

Proof of Claim 5.3 We use $\mathcal{A}$ to construct a forger $\mathcal{F}$ against sEUF-CMA security of the signature scheme OT. $\mathcal{F}$ simulates $\mathcal{A}$'s challenger in the keyword-privacy experiment of PKE-PEKS as follows: given input κ and the verification key vk^* (output by OT.$\mathsf{KeyGen}(\kappa)$), $\mathcal{F}$ first runs IBE.$\mathsf{KeyGen}(\kappa)$ to obtain (pk, sk), and then runs $\mathcal{A}(pk)$. Note that $\mathcal{F}$ can answer any token queries, test queries, and decryption queries with sk. When $\mathcal{A}$ outputs two keywords w_0^* and w_1^* and a message m^* where it wants to be challenged on, $\mathcal{F}$ proceeds as follows: it chooses a random bit b, computes $c^* \leftarrow$ IBE.$\mathsf{Encrypt}(pk, 0||vk^*, m^*)$, $s^* \leftarrow$ IBE.$\mathsf{Encrypt}(pk, (1||w_b^*), vk^*), vk^*)$, then obtains a signature σ^* of message $c^*||s^*$ via calling its signing oracle. Finally, $\mathcal{F}$ sends the challenge ciphertext $(vk^*, c^*, s^*, \sigma^*)$ to $\mathcal{A}$. If $\mathcal{A}$ submits a valid test query $\langle u, w\rangle$ in Phase 2 where $u = (vk^*, c, s, \sigma)$ and w is either w_0^* or w_1^* (now we must have $(c, s, \sigma) \neq (c^*, s^*, \sigma^*)$), $\mathcal{F}$ simply outputs $(c||s, \sigma)$ as its forgery. It is easy to see that $\mathcal{F}$'s success probability is exactly $\Pr[\mathsf{Forge}]$. Security of OT implies this claim. □

Proof of Claim 5.4 We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ against weak robustness of the IBE scheme. $\mathcal{B}$ simulates $\mathcal{A}$'s challenger in the keyword-privacy experiment of PKE-PEKS as follows: given mpk of the IBE, $\mathcal{B}$ sets $pk = mpk$ and runs $\mathcal{A}(pk)$. Note that in Phase 1 the decryption key extraction queries for "identities" of the form $1||w$ and $0||vk$ are always permitted by $\mathcal{B}$'s challenger, thereby $\mathcal{B}$ is able to handle any token queries, test queries, and decryption queries in Phase 1. In the challenge stage, when $\mathcal{A}$ outputs a message m^* and two keywords w_0^* and w_1^* where it wants to be challenged on, $\mathcal{B}$ runs OT.$\mathsf{KeyGen}(\kappa)$ to generate (vk^*, sk_σ^*), chooses a random bit b, then sends two identities $1||w_b^*$ and $1||w_{\bar{b}}^*$ and a message vk^* to its own challenger and halts. Suppose $s^* \leftarrow$ IBE.$\mathsf{Encrypt}(pk, 1||w_b^*, vk^*)$ is an IBE ciphertext honestly-generated by $\mathcal{B}$'s challenger, then it also serves as a valid component of an honestly-generated PKE-PEKS ciphertext. Thereby, $\mathcal{B}$'s success probability in the WROB experiment of IBE is exactly $\Pr[\mathsf{Break}]$. The assumed weak robustness of IBE implies this claim. □

Proof of Claim 5.5 We use $\mathcal{A}$ to construct an adversary $\mathcal{D}$ against ANO-IBE-CCA anonymity of the IBE scheme. $\mathcal{D}$ simulates $\mathcal{A}$'s challenger in the keyword-privacy experiment of PKE-PEKS as follows:

Setup: $\mathcal{D}$ is given mpk of the IBE scheme. $\mathcal{D}$ sets $pk = mpk$ and runs $\mathcal{A}(pk)$.

Phase 1: Upon receiving the token queries, test queries, and decryption queries issued by $\mathcal{A}$, $\mathcal{D}$ responds as follows:

- Token query $\langle w \rangle$: $\mathcal{D}$ issues decryption key query $\langle 1||w \rangle$ to its own challenger and forwards the reply to $\mathcal{A}$.
- Test query $\langle u, w \rangle$: $\mathcal{D}$ parses u as (vk, c, s, σ). If OT.$\mathsf{Verify}(vk, c||s, \sigma) = 0$, $\mathcal{D}$ outputs 0. Else, $\mathcal{D}$ issues decryption query $\langle 1||w, s \rangle$ to its own challenger. $\mathcal{D}$ outputs 1 if the reply is vk and 0 otherwise.
- Decryption query $\langle u \rangle$: $\mathcal{D}$ parses u as (vk, c, s, σ). If OT.$\mathsf{Verify}(vk, c||s, \sigma) = 0$, then $\mathcal{D}$ rejects this decryption query with $\perp$. If not, $\mathcal{D}$ issues decryption query $(0||vk, c)$ to its own challenger and forwards the reply to $\mathcal{A}$.

Challenge: When $\mathcal{A}$ outputs a message m^* and two keywords w_0^* and w_1^* where it wants to be challenged on, $\mathcal{D}$ proceeds as follows:

1. Run $(vk^*, sk_\sigma^*) \leftarrow$ OT.$\mathsf{KeyGen}(\kappa)$.
2. Compute $c^* \leftarrow$ IBE.$\mathsf{Encrypt}(pk, 0||vk^*, m)$.
3. Submit vk^* (as the message) and two target identities $1||w_0^*$ and $1||w_1^*$ to its own challenger, and get the response $s^* \leftarrow$ IBE.$\mathsf{Encrypt}(pk, 1||w_b^*, vk^*)$, where b is a random bit chosen by $\mathcal{D}$'s challenger.
4. Compute $\sigma^* \leftarrow$ OT.$\mathsf{Sign}(sk_\sigma^*, c^*||s^*)$.
5. Send $u^* = (vk^*, c^*, s^*, \sigma^*)$ to $\mathcal{A}$ as the challenge ciphertext .

Phase 2: $\mathcal{A}$ may issue more token queries, decryption queries, and test queries in Phase 2, $\mathcal{D}$ responds as follows:

- Token query $\langle w \rangle$: as long as $w \neq w_0^*, w_1^*$, $\mathcal{D}$ can always issue decryption key extraction query for identity $1||w$, then forwards the reply to $\mathcal{A}$.
- Test query $\langle u, w \rangle$: test queries $\langle u^*, w_0^* \rangle$ and $\langle u^*, w_1^* \rangle$ will be rejected. For a legal test query, $\mathcal{D}$ parses u as (vk, c, s, σ), first checks if OT.$\mathsf{Verify}(vk, c||s, \sigma) = 1$. If not, $\mathcal{D}$ returns 0. If so, when w is not w_0^* or w_1^*, $\mathcal{D}$ issues decryption query $\langle 1||w, s \rangle$ to its own challenger, returns 1 if the reply is vk and 0 if not. Otherwise, $\mathcal{D}$ proceeds as follows:
  - Case $vk = vk^*$: event Forge occurs (now w is either w_0^* or w_1^*, thus for a legal query we must have $u \neq u^*$), $\mathcal{D}$ aborts and outputs a random bit.
  - Case $vk \neq vk^*$: if $s \neq s^*$, $\mathcal{D}$ makes decryption query $\langle 1||w, s \rangle$ to its own challenger, then returns 1 if the reply is vk and 0 otherwise. If $s = s^*$, $\mathcal{D}$ returns 0. We stress that $\mathcal{D}$'s responses for test queries belong to this subbranch ($vk \neq vk^*$, $s = s^*$, and $w = w_0^*$ or $w = w_1^*$) are always correct conditioned on Break never occurs. To see this, note that if $w = w_b^*$, the decryption of s^* under "identity" $1||w$ is exactly vk^*

(differs from vk); if $w = w^*_{\bar{b}}$, the decryption of s^* under "identity" $1||w^*_{\bar{b}}$ is $\perp$ (Break does not happen). Thus the PEKS test fails in both cases.

- Decryption query $\langle u \rangle$: $\mathcal{D}$ responds to the decryption queries as it did in Phase 1. Since decryption queries of the form $\langle 0||vk, c\rangle$ are always permitted, $\mathcal{D}$ can answer all the decryption queries correctly.

Guess: As soon as $\mathcal{A}$ outputs its guess b' for b, $\mathcal{D}$ outputs b' to its own challenger.
Note that $\mathcal{D}$ represents a legal strategy for attacking ANO-IBE-CCA anonymity of the IBE scheme. Furthermore, $\mathcal{D}$ provides a perfect simulation for $\mathcal{A}$ in the keyword privacy experiment of PKE-PEKS conditioned on neither Forge nor Break occurs. Let SuccD denote the event of $\mathcal{D}$ outputting the correct bit in the ANO-IBE-CCA anonymity experiment of the IBE, we have:

$$|\Pr[\mathsf{SuccD} - \tfrac{1}{2}| = |\Pr[\mathsf{SuccA} \wedge \overline{\mathsf{Forge} \vee \mathsf{Break}}] + \tfrac{1}{2}\Pr[\mathsf{Forge}] - \tfrac{1}{2}|.$$

The assumed anonymity of the IBE implies this claim. □

Based on Claim 5.3, Claim 5.4, and Claim 5.5, we can derive a concrete bound of ϵ. Note that:

$$\begin{aligned} |\Pr[\mathsf{SuccA}] - \tfrac{1}{2}| &\leq |\Pr[\mathsf{SuccA} \wedge (\mathsf{Forge} \vee \mathsf{Break})] - \tfrac{1}{2}\Pr[\mathsf{Forge}]| + \\ &\quad |\Pr[\mathsf{SuccA} \wedge \overline{\mathsf{Forge} \vee \mathsf{Break}}] + \tfrac{1}{2}\Pr[\mathsf{Forge}] - \tfrac{1}{2}| \\ &\leq \tfrac{1}{2}\Pr[\mathsf{Forge}] + \Pr[\mathsf{Break}] + |\Pr[\mathsf{SuccA} \wedge \overline{\mathsf{Forge} \vee \mathsf{Break}}] + \tfrac{1}{2}\Pr[\mathsf{Forge}] - \tfrac{1}{2}|, \end{aligned}$$

we have $\epsilon \geq \frac{1}{2}\epsilon_3 + \epsilon_4 + \epsilon_2$. For the time complexity, it is easy to check that the running time of $\mathcal{F}$ is at most $t + t_b + q_w t_k + (q_t + q_d)(t_k + t_v + t_d) + 2t_e$, the running time of $\mathcal{B}$ is at most $t + t_g$, and the running time of $\mathcal{D}$ is at most $t + (q_w + q_t)t_k$. During the simulation, $\mathcal{B}$ asks at most $q_w + q_t + q_d$ decryption key extraction queries; $\mathcal{D}$ asks at most q_w decryption key extraction queries and at most $q_t + q_d$ decryption queries. This proves Lemma 5.2. ■

Theorem 5.1 immediately follows from Lemma 5.1 and Lemma 5.2. ■

Remark 5.1. In our two constructions, we use vk as the "message" for the PEKS encryption to further reduce the size of the overall ciphertext. Our two constructions could be potentially more efficient by employing two tricks as below: 1) observe that the PKE component only need to be CPA-secure, thus we may achieve better efficiency by employing the selective CPA-secure version encrypt algorithm of the underlying (H)IBE to build the PKE component; 2) observe that the user knows the "master secret key" sk of the underlying (H)IBE scheme, then it is very likely to speed up the decryption by directly using sk. See the instantiation presented in Section 7.3 for such an example.

Remark 5.2. It seems unlikely that one can improve our constructions by adapting the optimization suggested by [BCHK07] which replaces the one-time signature scheme by a combination of a message-authentication code (MAC) and a weak form of commitment. We sketch the reasons as below. One attempt is to encrypt the decommitment string dec in the PKE component. In this case, the gateway is unable to check the well-formedness of a PKE-PEKS ciphertext since the MAC can only be verified by the user holding the secret key sk. The other attempt is to encrypt dec in the PEKS component. In this case, the gateway is able to check the well-formedness of a PKE-PEKS ciphertext since it can recover the MAC key k and then verify the MAC. However, with k the gateway can also manipulate the PKE-PEKS ciphertext and thus break the data privacy. We omit the technical details here.

6 Consistency for PEKS Revisited

For a cryptographic scheme, except its security requirement, we also need to pay attention to its consistency requirement, which ensures that the primitive fulfills its functionality. As for PEKS, its consistency requirement set by [BDOP04] is that for two keywords w, w' and a PEKS ciphertext s of w, $\mathsf{Test}(t_{w'}, s)$ outputs 1 if $w = w'$ and outputs 0 if $w \neq w'$. This condition can be divided into two pieces. The former might be viewed as an analogy of correctness for IBE (decryption reverses encryption), which indicates that $\mathsf{Test}(t_w, s) = 1$ when s encrypts w. Adopting the treatment of [ABC$^+$08], we view it as a built-in property for PEKS and reserve the term consistency to the latter, namely $\mathsf{Test}(t_{w'}, s) = 0$ when s encrypts w and $w \neq w'$. Abdalla et al. [ABC$^+$08] formally defined perfect, statistical, and computational consistency for PEKS. In this work, we only focus on computational consistency.

6.1 Consistency for PEKS

We consider two levels of consistency for PEKS schemes, namely weak consistency (WCON) and strong consistency (SCON), which are defined by the following experiment:

Initialize: $\mathcal{CH}$ runs $\mathsf{KeyGen}(\kappa)$ to generate (pk, sk) and gives pk to $\mathcal{A}$.
Phase 1: $\mathcal{A}$ can adaptively make token queries $\langle w \rangle$. $\mathcal{CH}$ responds with $t_w \leftarrow \mathsf{TokenGen}(sk, w)$.
Finalize (weak consistency): $\mathcal{A}$ outputs two distinct keywords w and w'. $\mathcal{CH}$ honestly generates a PEKS ciphertext $s \leftarrow \mathsf{Encrypt}(pk, w)$. $\mathcal{A}$ succeeds if $\mathsf{Test}(t_{w'}, s) = 1$, where $t_{w'} \leftarrow \mathsf{TokenGen}(sk, w')$.
Finalize (strong consistency): $\mathcal{A}$ outputs two distinct keywords w and w' and a PEKS ciphertext s of w. Note that s might be generated using adversarially chosen random coins. $\mathcal{A}$ succeeds if $\mathsf{Test}(t_{w'}, s) = 1$, where $t_{w'} \leftarrow \mathsf{TokenGen}(sk, w')$.

We denote the event that $\mathcal{A}$ succeeds by SuccA and define $\mathcal{A}$'s advantage as $\mathrm{Adv}_{\mathcal{A}} \overset{\mathrm{def}}{=} \Pr[\mathsf{SuccA}]$. A PEKS scheme is said to be (t, q_w, ϵ) weakly consistent (resp. strong consistent) if for all t-time adversaries making at most q_w token queries have at most advantage ϵ in the corresponding experiment. We note that the weak consistency is exactly the original notion of consistency for PEKS defined in [ABC$^+$08, ABN10]. In what follows, we recall two notions of IBE schemes that are closely related to consistency of PEKS schemes.

Definition 6.1 (Robustness for IBE)**.** Abdalla et al. [ABN10] formally studied the notion of *robustness* of encryption schemes. In the IBE setting, robustness intuitively requires that a ciphertext c does not decrypt to a valid plaintext under the decryption keys for two distinct identities id and id', which is formally defined by the following experiment:

Initialize: $\mathcal{CH}$ runs $\mathsf{Setup}(\kappa)$ to generate (mpk, msk) and gives mpk to $\mathcal{A}$.
Phase 1: $\mathcal{A}$ can adaptively make decryption key extraction queries $\langle id \rangle$, and $\mathcal{CH}$ responds with $dk_{id} \leftarrow \mathsf{Extract}(msk, id)$.
Finalize (weak robustness): $\mathcal{A}$ outputs two distinct identities id and id' subject to the restriction that they had not been asked for decryption keys in Phase 1, and a message $m \in M$. $\mathcal{CH}$ honestly generates an IBE ciphertext $c \leftarrow \mathsf{Encrypt}(mpk, id, m)$. $\mathcal{A}$ succeeds if $m' \neq \perp$, where $m' \leftarrow \mathsf{Decrypt}(dk_{id'}, c)$ for $dk_{id'} \leftarrow \mathsf{Extract}(msk, id')$.
Finalize (strong robustness): $\mathcal{A}$ outputs two distinct identities id and id' subject to the restriction that they had not been asked for decryption keys in Phase 1, and a ciphertext c. Note that c may not be an honest encryption, say, generated using adversarially chosen random coins. $\mathcal{A}$ succeeds if $m \neq \perp$ and $m' \neq \perp$, where $m \leftarrow \mathsf{Decrypt}(dk_{id}, c)$ for $dk_{id} \leftarrow \mathsf{Extract}(msk, id)$ and $m' \leftarrow \mathsf{Decrypt}(dk_{id'}, c)$ for $dk_{id'} \leftarrow \mathsf{Extract}(msk, id')$.

We denote the event that $\mathcal{A}$ succeeds by SuccA and define $\mathcal{A}$'s advantage as $\mathrm{Adv}_{\mathcal{A}} \overset{\text{def}}{=} \Pr[\mathsf{SuccA}]$. An IBE scheme is said to be (t, q_k, ϵ) weakly robust (resp. strong robust) if for all t-time adversaries making at most q_k decryption key extraction queries have at most advantage ϵ in the corresponding experiment. Both weak robustness (WROB) and strong robustness (SROB) are also considered under chosen-ciphertext attacks [ABN10], yielding the notions of WROB-CCA and SROB-CCA.

Definition 6.2 (Collision-freeness for IBE)**.** Mohassel [Moh10] introduced the notion of *collision-freeness* for encryption schemes, which is a natural relaxation of robustness. In IBE setting, collision-freeness intuitively requires that a ciphertext does not decrypt to the same message under the decryption keys for two distinct identities, which is formally defined by the following experiment:

Initialize: same as the experiment for robustness.
Phase 1: same as the experiment for robustness.
Finalize (weak collision-freeness): $\mathcal{A}$ outputs two distinct identities id and id' subject to the restriction that they had not been asked for decryption keys in Phase 1, and a message $m \in M$. $\mathcal{CH}$ honestly generates an IBE ciphertext $c \leftarrow \mathsf{Encrypt}(mpk, id, m)$. $\mathcal{A}$ wins if $m = m'$, where $m' \leftarrow \mathsf{Decrypt}(dk_{id'}, c)$ for $dk_{id'} \leftarrow \mathsf{Extract}(msk, id')$.
Finalize (strong collision-freeness): $\mathcal{A}$ outputs two distinct identities id and id' subject to the restriction that they had not been asked for decryption keys in Phase 1, and a ciphertext c. Note that c may not be an honest encryption. $\mathcal{A}$ wins if $m = m'$, where $m \leftarrow \mathsf{Decrypt}(dk_{id}, c)$ for $dk_{id} \leftarrow \mathsf{Extract}(msk, id)$ and $m' \leftarrow \mathsf{Decrypt}(dk_{id'}, c)$ for $dk_{id'} \leftarrow \mathsf{Extract}(msk, id')$.

We denote the event that $\mathcal{A}$ succeeds by SuccA and define $\mathcal{A}$'s advantage as $\mathrm{Adv}_{\mathcal{A}} \overset{\text{def}}{=} \Pr[\mathsf{SuccA}]$. An IBE scheme is said to be (t, q_k, ϵ) weakly collision-free (resp. strong collision-free) if for all t-time adversaries making at most q_k decryption key extraction queries have at most advantage ϵ in the corresponding experiment. Similar to the notion of robustness, both weak collision-freeness (WCFR) and strong collision-freeness (SCFR) are also considered under chosen-ciphertext attacks [Moh10], yielding the notions of WCFR-CCA and SCFR-CCA.

6.2 Connection between Consistency, Robustness, and Collision-freeness

Addalla et al. [ABC+08] noticed that the original BDOP transform generally does not provide consistency, and thus proposed a new BDOP transform which is consistency-providing instead. After studying the robustness for encryption schemes, Abdalla et al. [ABN10] proved that even the original BDOP transform can guarantee the resulting PEKS scheme satisfies weak consistency if the underlying IBE scheme is SROB-CPA. However, strong robustness seems to be overkill for weak consistency. Note that Mohassel [Moh10] has pointed out that collision-freeness can be a sufficient requirement instead of robustness in some scenarios in practice. It is compelling to know if we can base the consistency for PEKS on collision-freeness for IBE. We answer this question affirmatively by presenting the following theorem:

Theorem 6.1. *For a PEKS scheme constructed using the BDOP transform, it is* (t, q_w, ϵ) *weakly (resp. strong) consistent if the underlying IBE scheme is* (t', q_k, ϵ') *weakly (resp. strong) collision-free, where* $\epsilon \geq \epsilon'$, $t \leq t'$, *and* $q_w \leq q_k$.

Proof. Suppose there is an adversary $\mathcal{A}$ has advantage $\mathrm{Adv}_{\mathcal{A}}$ against WCON/SCON of the PEKS scheme in running time t, we can construct an algorithm $\mathcal{D}$ that uses $\mathcal{A}$ to break WCFR/SCFR of the IBE scheme as follows:

Initialize: $\mathcal{D}$ is given (mpk, msk) of the IBE scheme. $\mathcal{D}$ forwards mpk to $\mathcal{A}$ as pk.

Phase 1: Upon receiving token query $\langle w \rangle$ issued by $\mathcal{A}$, $\mathcal{D}$ makes decryption key extraction query for "identity" w to its own challenger, and forwards the reply to $\mathcal{A}$.
Finalize (weak consistency): $\mathcal{A}$ outputs two distinct keywords w and w'. $\mathcal{D}$ outputs $(id = w, id' = w', m)$, where m is set to be the pre-fixed value specified by the BDOP transform. Let s_2 be an ciphertext of m under id which is honestly generated by $\mathcal{D}$'s challenger, then $s = (s_1 = m, s_2)$ is also an honestly-generated PEKS ciphertext.
Finalize (strong consistency): $\mathcal{A}$ outputs two distinct keywords w and w' and a PEKS ciphertext $s = (s_1, s_2)$ that encrypts w. $\mathcal{D}$ outputs $(id = w, id' = w', s_2)$ to its own challenger.

Since PEKS.$\mathsf{Test}(t_{w'}, s) = 1$ is equivalent to IBE.$\mathsf{Decrypt}(s_2, dk_{id'}) =$ IBE.$\mathsf{Decrypt}(s_2, dk_{id}) = m$, thus $\mathcal{D}$ breaks WCFR/SCFR of the IBE scheme with the same advantage as $\mathcal{A}$ breaks weak/strong consistency of the PEKS scheme. The running time of $\mathcal{D}$ is at most t. During the simulation, $\mathcal{D}$ asks at most q_w decryption key extraction queries. This proves the theorem. ■

It is straightforward to verify that this theorem also holds for the new BDOP transform. Note that Mohassel [Moh10] has shown that WCFR is strictly weaker than SROB, thus our result improves previous result due to [ABN10].

6.3 Consistency for PKE-PEKS

The notion of consistency of PEKS schemes extends naturally to PKE-PEKS schemes. We formally define consistency for PKE-PEKS schemes via the following experiment:

Initialize: $\mathcal{CH}$ runs $\mathsf{KeyGen}(\kappa)$ to generate (pk, sk) and sends pk to $\mathcal{A}$.
Phase 1: $\mathcal{A}$ can adaptively make token queries $\langle w \rangle$. $\mathcal{CH}$ responds with $t_w \leftarrow \mathsf{TokenGen}(sk, w)$.
Finalize (weak consistency): $\mathcal{A}$ outputs a message m and two distinct keywords w and w'. $\mathcal{CH}$ honestly generates a PKE-PEKS ciphertext $u \leftarrow \mathsf{Encrypt}(pk, m, w)$. $\mathcal{A}$ succeeds if $\mathsf{Test}(t_{w'}, u) = 1$, where $t_{w'} \leftarrow \mathsf{TokenGen}(sk, w')$.
Finalize (strong consistency): $\mathcal{A}$ outputs two distinct keywords w and w' and a PKE-PEKS ciphertext u encrypting (m, w) for some message m. Note that u might be generated using adversarially chosen random coins. $\mathcal{A}$ succeeds if $\mathsf{Test}(t_{w'}, u) = 1$, where $t_{w'} \leftarrow \mathsf{TokenGen}(sk, w')$.

We denote the event that $\mathcal{A}$ succeeds by SuccA and define $\mathcal{A}$'s advantage as $\mathrm{Adv}_{\mathcal{A}} \overset{\mathrm{def}}{=} \Pr[\mathsf{SuccA}]$. A PKE-PEKS scheme is said to be (t, q_w, ϵ) weakly consistent (resp. strong consistent) if for all t-time adversaries making at most q_w token queries have at most advantage ϵ in the corresponding experiment. Via a similar reduction as we did in the proof of Theorem 6.1, it is easy to verify our two constructions presented in Section 4 and Section 5 are weakly (resp. strong) consistent if the underlying (H)IBE scheme is weakly (resp. strong) collision-free.

Theorem 6.2. *For a PKE-PEKS scheme derived from the first generic construction, it is (t, q_w, ϵ) weakly (resp. strong) consistent if the underlying HIBE scheme is (t', q_k, ϵ') weakly (resp. strong) collision-free, where $\epsilon \geq \epsilon'$, $q_w \leq q_k$, and $t \leq t' - t_g$ (resp. $t \leq t'$).*

Proof. Suppose there is an adversary $\mathcal{A}$ has advantage $\mathrm{Adv}_{\mathcal{A}}$ against WCON/SCON of the PKE-PEKS scheme in running time t, we can build an adversary $\mathcal{D}$ against WCFR/SCFR of the underlying HIBE scheme as follows:

Initialize: $\mathcal{D}$ is given (mpk, msk) of the HIBE scheme. $\mathcal{D}$ forwards mpk to $\mathcal{A}$ as pk.
Phase 1: Upon receiving token query $\langle w \rangle$ issued by $\mathcal{A}$, $\mathcal{D}$ makes decryption key extraction query for "identity" $1||w$ to its own challenger, and forwards the reply with w to $\mathcal{A}$.

Finalize (weak consistency): $\mathcal{A}$ outputs a message m and two distinct keywords w and w'. $\mathcal{D}$ runs OT.$\mathsf{KeyGen}(\kappa)$ to generate (vk, sk_σ), then outputs $(id = (1||w, vk), id' = (1||w', vk), \tilde{m} = vk)$ to its own challenger.
Finalize (strong consistency): $\mathcal{A}$ outputs two distinct keywords w and w' and a PKE-PEKS ciphertext $u = (vk, c, s, \sigma)$ of (m, w) for some message m. Note that u might be generated using adversarially chosen random coins. $\mathcal{D}$ outputs $(id = (1||w, vk), id' = (1||w', vk), s)$ to its own challenger.

According to the first generic construction of PKE-PEKS from HIBE described in Section 4, we have PKE-PEKS.$\mathsf{Test}(t_{w'}, u) = 1 \Rightarrow$ HIBE.$\mathsf{Decrypt}(dk_{id'}, s) = vk$, where $u = (vk, c, s, \sigma)$ is a PKE-PEKS ciphertext of (m, w) and $id' = (1||w', vk)$. To see why this implication holds, we just need to decompose the left-hand side according to the PKE-PEKS construction from HIBE. More precisely, note that $t_{w'}$ is computed as (HIBE.$\mathsf{Extract}(sk, 1||w'), w'$) and the test algorithm outputs 1 if u is valid and $vk =$ HIBE.$\mathsf{Decrypt}(dk_{id'}, s)$, the implication immediately follows. Based on this implication, we conclude that:

- If $\mathcal{A}$ breaks WCON of the PKE-PEKS scheme with advantage $\mathrm{Adv}_{\mathcal{A}}$, which means event PKE-PEKS.$\mathsf{Test}(t_{w'}, u) = 1$ happens with probability $\mathrm{Adv}_{\mathcal{A}}$ for an honestly-generated PKE-PEKS ciphertext $u = (vk, c, s, \sigma)$, then event HIBE.$\mathsf{Decrypt}(dk_{id'}, s) = vk$ must happen with probability at least $\mathrm{Adv}_{\mathcal{A}}$ for an honestly-generated HIBE ciphertext $s \leftarrow$ HIBE.$\mathsf{Encrypt}(pk, id, \tilde{m})$, which is the third element from u. Thereby, $\mathcal{D}$ violates WCFR of the HIBE scheme with advantage at least $\mathrm{Adv}_{\mathcal{A}}$.
- If $\mathcal{A}$ breaks SCON of the PKE-PEKS scheme with advantage $\mathrm{Adv}_{\mathcal{A}}$, which means event PKE-PEKS.$\mathsf{Test}(t_{w'}, u) = 1$ happens with probability $\mathrm{Adv}_{\mathcal{A}}$ for an adversarially-generated PKE-PEKS ciphertext $u = (vk, c, s, \sigma)$, then event HIBE.$\mathsf{Decrypt}(dk_{id'}, s) = vk$ must happen with probability at least $\mathrm{Adv}_{\mathcal{A}}$. Thereby, $\mathcal{D}$ violates SCFR of the HIBE scheme with advantage at least $\mathrm{Adv}_{\mathcal{A}}$.

In the case of weak consistency, the running time of $\mathcal{D}$ is at most $t + t_g$. In the case of strong consistency, the running time of $\mathcal{D}$ is at most t. In both cases, $\mathcal{D}$ asks at most q_w decryption key extraction queries during the simulation. This concludes the proof of the theorem. ■

Theorem 6.3. *For a PKE-PEKS scheme derived from the second generic construction, it is* (t, q_w, ϵ) *weakly (resp. strong) consistent if the underlying IBE scheme is* (t', q_k, ϵ') *weakly (resp. strong) collision-free, where* $\epsilon \geq \epsilon'$, $q_w \leq q_k$, *and* $t \leq t' - t_g$ *(resp.* $t \leq t'$*).*

Proof. Suppose there is an adversary $\mathcal{A}$ has advantage $\mathrm{Adv}_{\mathcal{A}}$ against WCON/SCON of the PKE-PEKS scheme in running time t, we can build an adversary $\mathcal{D}$ breaking WCFR/SCFR of the underlying IBE scheme as follows:

Initialize: $\mathcal{D}$ is given (mpk, msk) of the IBE scheme. $\mathcal{D}$ forwards mpk to $\mathcal{A}$ as pk.
Phase 1: Upon receiving token query $\langle w \rangle$ issued by $\mathcal{A}$, $\mathcal{D}$ makes decryption key extraction query for "identity" $1||w$ to its own challenger, and forwards the reply to $\mathcal{A}$.
Finalize (weak consistency): $\mathcal{A}$ outputs a message m and two distinct keywords w and w'. $\mathcal{D}$ runs OT.$\mathsf{KeyGen}(\kappa)$ to generate (vk, sk_σ), outputs $(id = 1||w, id' = 1||w', \tilde{m} = vk)$ to its own challenger.
Finalize (strong consistency): $\mathcal{A}$ outputs two distinct keywords w and w' and a PKE-PEKS ciphertext $u = (vk, c, s, \sigma)$ of (m, w) for some message m. Note that u might be generated using adversarially chosen random coins. $\mathcal{D}$ outputs $(id = 1||w, id' = 1||w', s)$ to its challenger.

According to the second generic construction of PKE-PEKS from IBE described in Section 5, we have PKE-PEKS.$\mathsf{Test}(t_{w'}, u) = 1 \Rightarrow$ IBE.$\mathsf{Decrypt}(dk_{id'}, s) = vk$, where $u = (vk, c, s, \sigma)$ is a

PKE-PEKS ciphertext of (m, w), $id' = 1||w'$. To see why this implication holds, we just need to decompose the left-hand side according to the PKE-PEKS construction from IBE. More precisely, note that $t_{w'}$ is computed as $dk_{id'} \leftarrow \text{IBE.}\mathsf{Extract}(sk, 1||w')$ and the test algorithm outputs 1 if u is valid and $vk = \text{IBE.}\mathsf{Decrypt}(t_{w'}, s)$, the implication immediately follows. Based on this implication, we conclude that:

- If $\mathcal{A}$ breaks WCON of the PKE-PEKS scheme with advantage $\text{Adv}_{\mathcal{A}}$, which means event $\text{PKE-PEKS.}\mathsf{Test}(t_{w'}, u) = 1$ happens with probability $\text{Adv}_{\mathcal{A}}$ for an honestly-generated PKE-PEKS ciphertext $u = (vk, c, s, \sigma)$, then event $\text{IBE.}\mathsf{Decrypt}(dk_{id'}, s) = vk$ must happen with probability at least $\text{Adv}_{\mathcal{A}}$ for an honestly-generated IBE ciphertext $s \leftarrow \text{IBE.}\mathsf{Encrypt}(pk, id, \tilde{m})$, which is the third element from u. Thereby, $\mathcal{D}$ violates WCFR of the IBE scheme with advantage at least $\text{Adv}_{\mathcal{A}}$.
- If $\mathcal{A}$ breaks SCON of the PKE-PEKS scheme with advantage $\text{Adv}_{\mathcal{A}}$, which means event $\text{PKE-PEKS.}\mathsf{Test}(t_{w'}, u) = 1$ must happen with probability $\text{Adv}_{\mathcal{A}}$ for an adversarially-generated PKE-PEKS ciphertext $u = (vk, c, s, \sigma)$, then event $\text{IBE.}\mathsf{Decrypt}(dk_{id'}, s) = vk$ happens with probability at least $\text{Adv}_{\mathcal{A}}$. Thereby, $\mathcal{D}$ violates SCFR of the IBE scheme with advantage at least $\text{Adv}_{\mathcal{A}}$.

In the case of weak consistency, the running time of $\mathcal{D}$ is at most $t + t_g$. In the case of strong consistency, the running time of $\mathcal{D}$ is at most t. In both cases, $\mathcal{D}$ asks at most q_w decryption key extraction queries during the simulation. This concludes the proof of the theorem. ■

6.4 Enhanced Notions of Consistency, Collision-freeness, and Robustness

Enhanced Notion of Consistency. Abdalla et al. [ABC+08] formulated the consistency for PEKS more like a security condition, via a security experiment involving an adversary. Just as they indicated, the adversary against consistency is not very "adversarial": it is modeled to capture some kind of worst case but not malicious behavior. They also noticed that the basic notion of consistency can be made stronger by giving the adversary a token oracle and/or a test oracle. Indeed, we would go so far as to say the consistency should retain even the secret key sk is exposed to the adversary. This interpretation agrees with the intuition of the notion of consistency for PEKS [BDOP04]. In our understanding, consistency, like correctness, should also be viewed as a built-in property for PEKS and PKE-PEKS. Therefore, we are motivated to enhance the notion of consistency for PEKS and PKE-PEKS schemes. The enhanced weak/strong consistency is same as the original weak/strong consistency, except that in the **Initialize** step, the adversary is given both pk and sk.

Enhanced Notion of Collision-freeness and Robustness. For the same reasoning as above, the original notion of collision-freeness can also be enhanced. Particularly, in the IBE setting the enhanced collision-freeness can be defined analogously as above. Namely, no adversary can break collision-freeness with non-negligible probability even it knows the master secret key msk. As an analogy of Theorem 6.1, we claim that a PEKS scheme is enhanced weak/strong consistent if the underlying IBE scheme is enhanced weak/strong collision-free. We omit the proof here due to its straightforwardness.

Schemes such as the ElGamal PKE [ElG85] and the Boyen-Waters IBE [BW06] are shown to be strong collision-free in [Moh10]. Interestingly, they are also strong collision-free in the enhanced sense. Moreover, we find that many encryption schemes are at least weak collision-freeness in the enhanced sense without any modification, such as the Cramer-Shoup PKE [CS02], the DHIES [ABR01], the Boneh-Franklin IBE [BF03], the Sakai-Kasahara IBE [SK03], the Boneh-Boyen IBE [BB04a], and the Waters IBE [Wat05]. For many encryption schemes, collision-freeness inherently relies on their internal structure and collision-resistance of hash functions being used, but not on the hardness of the underlying assumptions. Thereby, we consider the enhanced notion of collision-freeness as a reasonable strengthening. It might be fairly viewed as a built-in property for PKE/IBE. The same observation also holds for the case of robustness, thus we can enhance the notion of robustness similarly. We would like to mention that the enhanced notion of robustness is concurrently and independently proposed by Farshim et al. [FLPQ13], which they refer to as unrestricted strong robustness.

7 Instantiations

7.1 Pairing-based Instantiation of Construction 1

De Caro et al. [DIP10] constructed a CPA-secure and anonymous HIBE based on three newly introduced complexity assumptions in composite bilinear groups in the standard model. We give a two-level version of the HIBE scheme [DIP10] as below:

- $\mathsf{KeyGen}(\kappa, 2)$: take as input a security parameter κ, generate four primes p_1, p_2, p_3, p_4, two groups $\mathbb{G}$ and $\mathbb{G}_T$ of order $N = p_1p_2p_3p_4$, subgroups $\mathbb{G}_{p_i}$ of $\mathbb{G}$ with order p_i for $1 \le i \le 4$, and a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$; pick $Y_1, X_1, u_1, u_2 \xleftarrow{R} \mathbb{G}_{p_1}$, $Y_3 \xleftarrow{R} \mathbb{G}_{p_3}$, $X_4, Y_4 \xleftarrow{R} \mathbb{G}_{p_4}$ and $\alpha \xleftarrow{R} \mathbb{Z}_N$, output the master key pair
$$mpk = (N, Y_1, Y_3, Y_4, t = X_1X_4, u_1, u_2, W = e(Y_1, Y_1)^\alpha), \quad msk = (X_1, \alpha).$$
- $\mathsf{Extract}(msk, id)$: take as input msk and a level-1 identity $id = (id_1)$ where $id_1 \in \mathbb{Z}_{p_1}$, pick $r_1, r_2 \xleftarrow{R} \mathbb{Z}_N$, and $R_{1,1}, R_{1,2}, R_{2,1}, R_{2,2} \xleftarrow{R} \mathbb{G}_{p_3}$, then compute
$$\begin{aligned} &K_{1,1} = Y_1^{r_1}R_{1,1}, \quad K_{1,2} = Y_1^{\alpha}(u_1^{id}X_1)^{r_1}R_{1,2}, \quad E_{1,2} = u_2^{r_1}R_{1,2}, \\ &K_{2,1} = Y_1^{r_2}R_{2,1}, \quad K_{2,2} = (u_1^{id}X_1)^{r_2}R_{2,2}, \quad E_{2,2} = u_2^{r_2}R_{2,2}, \end{aligned}$$
output the decryption key $dk_{id} = (K_{1,1}, K_{1,2}, E_{1,2}, K_{2,1}, K_{2,2}, E_{2,2})$.
- $\mathsf{Derive}(dk_{id}, id')$: take as input a decryption key $dk_{id} = (K_{1,1}, K_{1,2}, E_{1,2}, K_{2,1}, K_{2,2}, E_{2,2})$ for a level-1 identity $id = (id_1)$ and a level-2 identity $id = (id_1, id_2)$, pick $\tilde{r}_1 \xleftarrow{R} \mathbb{Z}_N$, and and $\tilde{R}_{1,1}, \tilde{R}_{1,2} \xleftarrow{R} \mathbb{G}_{p_3}$, then compute
$$K'_{1,1} = K_{1,1}(K_{2,1})^{\tilde{r}_1}\tilde{R}_{1,1}, \quad K'_{1,2} = K_{1,2}(K_{2,2})^{\tilde{r}_1}(E_{1,2})^{id_2}(E_{2,2})^{\tilde{r}_1 id_2}\tilde{R}_{1,2},$$
output the decryption key $dk_{id} = (K'_{1,1}, K'_{1,2})$. Note that for a level-2 identity, it's not necessary to generate any other secret elements, say E_i's as in the output of $\mathsf{Extract}$ algorithm, since we do not need to derive a decryption key for the next level (no such a level exists).
- $\mathsf{Encrypt}(mpk, id, m)$: take as input mpk, an identity $id = (id_1, id_2)$, and a message $m \in \mathbb{G}_T$, choose $s \xleftarrow{R} \mathbb{Z}_N$ and $Z, Z' \xleftarrow{R} \mathbb{G}_{p_4}$, then compute
$$c_0 = m \cdot W^s, \qquad c_1 = (u_1^{id_1}u_2^{id_2}t)^s Z, \qquad c_2 = Y_1^s Z',$$
output the ciphertext $c = (c_0, c_1, c_2)$.
- $\mathsf{Decrypt}(dk_{id}, c)$: take as input a decryption key dk_{id} for identity id and a ciphertext $c = (c_0, c_1, c_2)$, output the message
$$m = c_0 \cdot \frac{e(K_{1,1}, c_1)}{e(K_{1,2}, c_2)}.$$

Collision-freeness of the above scheme is easy to verify. As for the candidate of one-time signature, we recommend the Boneh and Boyen short signature scheme [BB08], which is sEUF-CMA secure based on the strong Diffie-Hellman (SDH) assumption in the bilinear groups in the standard model. By employing these two schemes as the underlying primitives for our first generic construction, we obtain a pairing-based PKE-PEKS scheme which is consistent and jointly CCA-secure in the standard model.

7.2 Lattice-based Instantiation of Construction 1

Agrawal et al. [ABB10] proposed a CPA-secure and anonymous HIBE scheme based on the learning with errors (LWE) assumption in the standard model. Their original scheme encrypts just one bit at one time and is only selective-identity secure and anonymous. As the authors of [ABB10] pointed out, their scheme can be simply extended to support multi-bit encryption by reusing the randomness of the encryption algorithm, and can be bootstrapped to an adaptive-identity secure and anonymous scheme by applying a similar technique used in [Wat05, Boy10].

Before we present the two-level version of the HIBE scheme [ABB10], we first fix some notations and recall several related algorithms on lattices in the sense of functionality (since we try not to involve too much knowledge on lattices here). For integers n, m and q, we treat a matrix in $\mathbb{Z}_q^{n\times m}$ simply as the set of its column vectors in $\mathbb{Z}_q^n$, and denoted by bold capital letters (e.g., $\mathbf{A}$). We use bold lower-case letters to denote vectors $\mathbf{x} \in \mathbb{Z}_q^n$, and the l_2 norm (namely, the Euclidean norm) of a vector $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{Z}_q^n$ is denoted by $\|\mathbf{x}\| = \sqrt{\sum_{i=1}^n x_i^2}$. In addition, the norm of a matrix $\mathbf{X}$ is defined as the norm of its longest column (i.e., $\|\mathbf{X}\| = \max_i \|\mathbf{x}_i\|$).

- $\mathsf{TrapGen}(n, q)$: take as input two integers n and q, output a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and a trapdoor $\mathbf{T_A} \in \mathbb{Z}^{m\times m}$, such that $\mathbf{AT_A} = \mathbf{0} \mod q$ and $\|\mathbf{T_A}\| \le O(n \log q)$.
- $\mathsf{SampleBasisLeft}(\mathbf{A}, \mathbf{B}, \mathbf{T_A}, \sigma)$: take as input two matrices $\mathbf{A} \in \mathbb{Z}_q^{n\times m_1}$, $\mathbf{B} \in \mathbb{Z}_q^{n\times m_2}$, a trapdoor $\mathbf{T_A} \in \mathbb{Z}^{m_1\times m_1}$ of $\mathbf{A}$, and a real σ, output a trapdoor $\mathbf{T_F}$ of $\mathbf{F} = (\mathbf{A}\|\mathbf{B}) \in \mathbb{Z}_q^{n\times(m_1+m_2)}$.
- $\mathsf{SamplePre}(\mathbf{A}, \mathbf{T_A}, \mathbf{u}, \tau)$: take as input a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, a trapdoor $\mathbf{T_A} \in \mathbb{Z}^{m\times m}$ of $\mathbf{A}$, a vector $\mathbf{u} \in \mathbb{Z}_q^n$ and a real τ, output a short vector $\mathbf{e} \in \mathbb{Z}^{m\times m}$ such that $\mathbf{Ae} = \mathbf{u} \mod q$.

We refer the readers to [ABB10] for more details about the parameters and the algorithms.

For simplicity, we assume the level-2 identity space is $\{-1, 1\}^l \times \{-1, 1\}^l$, the adaptive-identity secure and anonymous HIBE encrypting one bit is given below:

- $\mathsf{KeyGen}(\kappa, 2)$: take as input a security parameter κ, generate a $n \times m$ matrix $\mathbf{A}_0 \in \mathbb{Z}^{n\times m}$ together with its trapdoor $\mathbf{T}_{\mathbf{A}_0}$ using the $\mathsf{TrapGen}$ algorithm; randomly choose $2(l + 1)$ matrices $\{\mathbf{A}_i, \{\mathbf{B}_{i,j}\}_{j=1,\ldots,l}\}_{i=1,2}$ from $\mathbb{Z}_q^{n\times m}$ and a vector $\mathbf{u}$ from $\mathbb{Z}_q^n$, and finally output the master key pair

$$mpk = (\mathbf{A}_0, \{\mathbf{A}_i, \{\mathbf{B}_{i,j}\}_{j=1,\ldots,l}\}_{i=1,2}, \mathbf{u}), \quad msk = (\mathbf{T}_{\mathbf{A}_0}) \in \mathbb{Z}^{m\times m}.$$

- $\mathsf{Extract}(msk, id)$: take as input msk and an identity $id = (id_1)$ where $id_1 = (b_{1,1}, \ldots, b_{1,l}) \in \{-1, 1\}^l$, define matrix $\mathbf{F}_{id} = (\mathbf{A}_0\|\mathbf{A}_1 + \sum_{j=1}^l b_{1,j}\mathbf{B}_{1,j})$, pick a real σ, then output a decryption key dk_{id} for id as

$$dk_{id} \leftarrow \mathsf{SampleBasisLeft}\big(\mathbf{A}_0, \mathbf{A}_1 + \sum_{j=1}^{l} b_{1,j}\mathbf{B}_{1,j}, msk, \sigma\big).$$

- $\mathsf{Derive}(dk_{id}, id')$: take as input a decryption key dk_{id} for identity $id = (id_1)$ and an identity $id' = (id_1, id_2)$ where $id_2 = (b_{2,1}, \ldots, b_{2,l}) \in \{-1, 1\}^l$, define $\mathbf{F}_{id'} = (\mathbf{F}_{id} || \mathbf{A}_2 + \sum_{j=1}^{l} b_{2,j}\mathbf{B}_{2,j})$, then output the decryption key $dk_{id'}$ for id' as

$$dk_{id'} \leftarrow \mathsf{SampleBasisLeft}\big(\mathbf{F}_{id'}, \mathbf{A}_2 + \sum_{j=1}^{l} b_{2,j}\mathbf{B}_{2,j}, dk_{id}, \sigma\big).$$

- $\mathsf{Encrypt}(mpk, id, m)$: take as input mpk, an identity id, and a message $m \in \{0, 1\}$, first define the corresponding matrix $\mathbf{F}_{id}$ as in the algorithm $\mathsf{Extract}$ (i.e., a level-1 identity) or Derive (i.e., a level-2 identity); randomly choose a vector $\mathbf{s} \in \mathbb{Z}_q^n$ and a matrix $\mathbf{R} \in \{-1, 1\}^{m \times km}$, where $k = 1$ for level-1 identity, and $k = 2$ for level-2 identity, randomly choose a noise vector $x \in \chi, \mathbf{y} \in \chi^m$, and finally output the ciphertext

$$c = \big(c_0 = \mathbf{u}^T\mathbf{s} + x + m\Big\lfloor\frac{q}{2}\Big\rfloor, c_1 = \mathbf{F}_{id}^T\mathbf{s} + \mathbf{R}^T\mathbf{y}\big).$$

- $\mathsf{Decrypt}(dk_{id}, c)$: take as input a decryption key dk_{id} for identity id and a ciphertext $c = (c_0, c_1)$, first set $\tau = \sigma\sqrt{(k+1)m\omega(\sqrt{\log(km)})}$, then compute

$$\mathbf{e}_{id} \leftarrow \mathsf{SamplePre}(\mathbf{F}_{id}, dk_{id}, \mathbf{u}, \tau).$$

  Let $w = c_0 - \mathbf{e}_{id}^T c_1$. If $|w - \lfloor\frac{q}{2}\rfloor| < \lfloor\frac{q}{4}\rfloor$, output 1. Else, output 0.

The CPA-security and anonymity of the above two-level HIBE scheme is implied by the original results of [ABB10], while collision-freeness is easy to be verified, Very recently, Micciancio and Peikert [MP12] constructed a short signature scheme, which is *static* sEUF-CMA secure based on the small integer solution (SIS) assumption on lattices in the standard model. We can bootstrap it to standard sEUF-CMA security by leveraging the generic transform [KR00] with chameleon hash functions. A suitable type of chameleon hash function [CHKP10] has been constructed under a weak hardness of the SIS assumption. By employing these two schemes as the underlying primitives for our first generic construction, we obtain a lattice-based PKE-PEKS scheme which is consistent and jointly CCA-secure in the standard model.

7.3 Pairing-based Instantiation of Construction 2

We now give an instantiation of our second generic construction by employing the Gentry's IBE [Gen06] together with the Boneh-Boyen signature scheme [BB08].

$\mathsf{KeyGen}(\kappa)$: generate two groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p along with a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, pick $g, h_1, h_2, h_3 \xleftarrow{R} \mathbb{G}^*$ and $\alpha \xleftarrow{R} \mathbb{Z}_p$, set $g_1 = g^\alpha$; choose a collision-resistant hash function H from $\{0, 1\}^*$ to $\mathbb{Z}_p$; output public key $pk = (g, g_1, h_1, h_2, h_3, \mathsf{H})$ and secret key $sk = \alpha$.

$\mathsf{Encrypt}(pk, m, w)$:

1. Pick $t \xleftarrow{R} \mathbb{G}^*$ and $x, y \xleftarrow{R} \mathbb{Z}_p^*$, compute $u = t^x$, $v = t^y$, and then set $vk = (t, u, v)$, $sk_\sigma = (x, y)$.
2. Compute $id_0 \leftarrow \mathsf{H}(0||vk)$, choose $r_0 \xleftarrow{R} \mathbb{Z}_p$, and compute $c_1 = g_1^{r_0}g^{-r_0 \cdot id_0}$, $c_2 = m \cdot e(g, h_1)^{-r_0}$; set $c = (c_1, c_2)$.
3. Compute $id_1 \leftarrow \mathsf{H}(1||w)$, choose $r_1 \xleftarrow{R} \mathbb{Z}_p$, and compute $s_1 = g_1^{r_1}g^{-r_1 \cdot id_1}$, $s_2 = e(g, g)^{r_1}$, $s_3 = \mathsf{H}(vk) \cdot e(g, h_1)^{-r_1}$, $s_4 = e(g, h_2)^{r_1}e(g, h_3)^{r_1\beta}$ for $\beta \leftarrow \mathsf{H}(s_1||s_2||s_3)$; set $s = (s_1, s_2, s_3, s_4)$.

4. Compute $z \leftarrow \mathsf{H}(c||s)$, pick $\sigma_1 \xleftarrow{\mathrm{R}} \mathbb{Z}_p \backslash \{ -\frac{x+z}{y} \}$, and compute $\sigma_2 = t^{1/(x+z+y\sigma_1)}$, here the inverse is computed modulo p; set the signature $\sigma = (\sigma_1, \sigma_2)$.
5. Output the final ciphertext $u = (vk, c, s, \sigma)$.

$\mathsf{Decrypt}(sk, u)$:

1. Parse u as (vk, c, s, σ), where $vk = (t, u, v)$, and $\sigma = (\sigma_1, \sigma_2)$.
2. Compute $z \leftarrow \mathsf{H}(c||s)$. If $e(\sigma_2, u \cdot t^z \cdot v^{\sigma_1}) \neq e(t, t)$, output $\perp$. Otherwise, compute $id_0 \leftarrow \mathsf{H}(0||vk)$ and output $m = c_2 \cdot e(c_1, h^{1/(\alpha - id_0)})$.

$\mathsf{TokenGen}(sk, w)$:

1. Compute $id_1 \leftarrow \mathsf{H}(1||w)$, and then choose $dk_{i,1} \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, compute $dk_{i,2} = (h_i g^{-dk_{i,1}})^{1/(\alpha - id_1)}$ and set $dk_i = (dk_{i,1}, dk_{i,2})$ for $1 \leq i \leq 3$. If $\alpha = id_1$, the algorithm aborts. Otherwise, output $t_w = (dk_1, dk_2, dk_3)$.

$\mathsf{Test}(t_w, u)$:

1. Parse t_w as (dk_1, dk_2, dk_3), u as (vk, c, s, σ).
2. Compute $z \leftarrow \mathsf{H}(c||s)$. If $e(\sigma_2, u \cdot t^z \cdot v^{\sigma_1}) \neq e(t, t)$, output 0. Otherwise, compute $\beta \leftarrow \mathsf{H}(s_1||s_2||s_3)$, test if $s_4 = e(s_1, dk_{2,2} dk_{3,2}^{\beta}) s_2^{dk_{2,1} + dk_{3,1}\beta}$. If the check fails, output 0. Else, continue to check if $\mathsf{H}(vk) = s_3 \cdot e(s_1, dk_{1,2}) s_2^{dk_{1,1}}$. If so, output 1. Else output 0.

The Gentry's IBE [Gen06] is ANO-IBE-CCA based on the truncated q-ABDHE assumption, and the Boneh-Boyen signature scheme [BB04b] is sEUF-CMA secure based on the SDH assumption. Besides, the Gentry's IBE [Gen06] is weak robust against CCA-attack and thus it is certainly weak collision-free. Therefore, according to Theorem 5.1 this above construction is consistent and jointly CCA-secure in the standard model.

8 Extensions

Keyword Hiding. As noticed in [BW07, HW08, Nis12], the IND-PEKS-CPA/CCA notions only ask that searchable ciphertext does not reveal any information about the keyword, but neglect to capture keyword privacy against the gateway, namely a token does not reveal information about its corresponding keyword. We refer to such privacy as keyword hiding, which is desired in scenarios that the user want to keep the specific keyword hidden from the gateway. It is often noted that the standard indistinguishable-style keyword hiding cannot be achieved in non-interactive setting.[7] To see this, note that given two different keywords w_0 and w_1 and a token t_{w_b}, one can always learn b by using "encrypt-then-test" method [HW08, Nis12]. In fact, due to the functionality of PEKS schemes, no meaningful notion of keyword hiding is possible beyond the minimum assumption that the keywords corresponding to the given tokens are sampled from distributions with certain amount of min-entropy (which must be at least super-logarithmic in the security parameter κ). Recently, Boneh et al. [BRS13] proposed a new notion named "function privacy" for IBE schemes, which captures the intuition that secret key sk_{id} hides its functionality. They formalized this notion via an indistinguishable-style definition, which requires that sk_{id} for an identity id sampled from any sufficiently unpredictable distribution is indistinguishable from $sk_{id'}$ for an independently and uniformly sampled identity id'. They also developed an approach called "extract-augment-combine" for designing IBE

[7]Here, "non-interactive" means both token generation and keyword search are done in a non-interactive manner. We emphasize that indistinguishable-style keyword hiding is possible in interactive setting. Boneh et al. [BKOI07] constructed a PEKS scheme allowing Private Information Retrieval [KO97], which hides all the information including the keyword.

schemes that satisfy the notion of function privacy. Noting the connection between IBE and PEKS, keyword hiding for PEKS is in spirit of function privacy for IBE. Following [BRS13], we can formalize an indistinguishable-style definition of keyword hiding for PEKS schemes as well as PKE-PEKS schemes. It seems plausible that one could achieve indistinguishable keyword hiding via the extract-augment-combine approach.

Multi-Keyword Setting. For brevity, we described our construction in single keyword setting. Our construction extends easily to multi-keyword setting as follows: to encrypt message m with a string of keywords $(w_1, \dots, w_n)$ intended to Alice, Bob generates a key pair (vk, sk) for a one-time signature scheme and creates a PKE ciphertext $c \leftarrow \mathsf{Encrypt}(pk, 1||vk, m)$ as in the single keyword setting; then creates a PEKS ciphertext $s = s_1||\dots||s_n$, where $s_i \leftarrow \mathsf{Encrypt}(pk, (0||w_i, vk), vk)$ for $1 \leq i \leq n$, and compute the signature σ of $c||s$ with sk. The final PKE-PEKS ciphertext is $(vk, c||s, \sigma)$. The security proof is omitted here since it is very similar to that for the construction in the single keyword setting.

Integrated IBE and IBEKS. Abdalla et al. [ABC+08] put forwarded the concept of Identity-Based Encryption with Keyword Search (IBEKS). Naturally, we can define integrated (H)IBE and (H)IBEKS scheme and its joint security notion analogously. Both our constructions extend to give transforms from (H)IBE schemes with appropriate properties to jointly CCA-secure integrated (H)IBE-(H)IBEKS schemes.

Acknowledgments

We are grateful to Zongyang Zhang, Qiong Huang, and Sherman S.M. Chow for helpful discussions. We also thank the anonymous DCC reviewers for many useful comments.

References

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (h)ibe in the standard model. In *Advances in Cryptology - EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 553–572. Springer, 2010.

[ABC+08] 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. *J. Cryptology*, 21(3):350–391, 2008.

[ABN10] Michel Abdalla, Mihir Bellare, and Gregory Neven. Robust encryption. In *TCC 2010*, volume 5978 of *LNCS*, pages 480–497. Springer, 2010.

[ABR01] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. The oracle diffie-hellman assumptions and an analysis of dhies. In *Topics in Cryptology - CT-RSA 2001*, volume 2020 of *LNCS*, pages 143–158. Springer, 2001.

[BB04a] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity based encryption without random oracles. In *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer, 2004.

[BB04b] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73, 2004.

[BB08] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the sdh assumption in bilinear groups. *J. Cryptology*, 21(2):149–177, 2008.

[BCHK07] Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. *SIAM Journal on Computation*, 36(5):1301–1328, 2007.

[BDOP04] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *Advances in Cryptology - EUROCRYPT 2004*, volume 3621 of *LNCS*, pages 506–522. Springer, 2004.

[BF03] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. *SIAM Journal on Computation*, 32:586–615, 2003.

[BGH07] Dan Boneh, Craig Gentry, and Michael Hamburg. Space-efficient identity based encryption without pairings. In *48th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2007*, pages 647–657. IEEE Computer Society, 2007.

[BKOI07] Dan Boneh, Eyal Kushilevitz, Rafail Ostrovsky, and William E. Skeith III. Public key encryption that allows pir queries. In *Advances in Cryptology - CRYPTO 2007*, volume 4622 of *LNCS*, pages 50–67. Springer, 2007.

[Boy10] Xavier Boyen. Lattice mixing and vanishing trapdoors: A framework for fully secure short signatures and more. In *Public Key Cryptography - PKC 2010*, volume 6056 of *LNCS*, pages 499–517. Springer, 2010.

[BRS13] Dan Boneh, Ananth Raghunathan, and Gil Segev. Function-private identity-based encryption: Hiding the function in functional encryption. In *Advances in Cryptology - CRYPTO 2013*, volume 8043 of *LNCS*, pages 461–478. Springer, 2013.

[BSNS06] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. On the integration of public key data encryption and public key encryption with keyword search. In *Information Security, 9th International Conference, ISC 2006*, volume 4176 of *LNCS*, pages 217–232. Springer, 2006.

[BW06] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *Advances in Cryptology - CRYPTO 2006*, volume 4117 of *LNCS*, pages 290–307. Springer, 2006.

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007*, volume 4392 of *LNCS*, pages 535–554. Springer, 2007.

[CHK04] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity based encryption. In *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 207–222. Springer, 2004.

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In *Advances in Cryptology - EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 523–552. Springer, 2010.

[Coc01] Clifford Cocks. An indentity based encryption scheme based on quadratic residues. In *Cryptography and Coding, 8th IMA International Conference*, volume 2260 of *LNCS*, pages 360–363. Springer, 2001.

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *Advances in Cryptology - EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, 2002.

[DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM J. Comput.*, 30(2):391–437, 2000.

[DIP10] Angelo De Caro, Vincenzo Iovino, and Giuseppe Persiano. Fully secure anonymous hibe and secret-key anonymous ibe with short ciphertexts. In *4th International Conference - Pairing-Based Cryptography - Pairing 2010*, volume 6487 of *LNCS*, pages 347–366. Springer, 2010.

[DK05] Yevgeniy Dodis and Jonathan Katz. Chosen-ciphertext security of multiple encryption. In *Theory of Cryptography, TCC 2005*, volume 3378 of *LNCS*, pages 188–209. Springer, 2005.

[DS07] Giovanni Di Crescenzo and Vishal Saraswat. Public key encryption with searchable keywords based on jacobi symbols. In *Progress in Cryptology - INDOCRYPT 2007*, volume 4859 of *LNCS*, pages 282–296. Springer, 2007.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.

[FLPQ13] Pooya Farshim, Benoît Libert, Kenneth G. Paterson, and Elizabeth A. Quaglia. Robust encryption, revisited. In *Public-Key Cryptography - PKC 2013*, volume 7778 of *LNCS*, pages 352–368. Springer, 2013.

[FP07] Thomas Fuhr and Pascal Paillier. Decryptable searchable encryption. In *Provable Security, First International Conference, ProvSec 2007*, volume 4784 of *LNCS*, pages 228–236. Springer, 2007.

[Gen06] Craig Gentry. Practical identity-based encryption without random oracles. In *Advances in Cryptology - EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 445–464. Springer, 2006.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, STOC 2008*, pages 197–206. ACM, 2008.

[GS02] Craig Gentry and Alice Silverberg. Hierarchical id-based cryptography. In *Advances in Cryptology - ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 548–566. Springer, 2002.

[HL02] Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In *Advances in Cryptology - EUROCRYPT 2002*, volume 2322 of *LNCS*, pages 466–481. Springer, 2002.

[HP01] Stuart Haber and Benny Pinkas. Securely combining public-key cryptosystems. In *Proceedings of the 8th ACM Conference on Computer and Communications Security, CCS 2001*, pages 215–224. ACM, 2001.

[HW08] Dennis Hofheinz and Enav Weinreb. Searchable encryption with decryption in the standard model. IACR Cryptology ePrint Archive, Report 2008/423, 2008. http://eprint.iacr.org/2008/423.

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *38th Annual Symposium on Foundations of Computer Science, FOCS 1997*, pages 364–373. IEEE Computer Society, 1997.

[KR00] Hugo Krawczyk and Tal Rabin. Chameleon signatures. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2000.* The Internet Society, 2000.

[Moh10] Payman Mohassel. A closer look at anonymity and robustness in encryption schemes. In *Advances in Cryptology - ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 501–518. Springer, 2010.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Advances in Cryptology - EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, 2012.

[Nis12] Mototsugu Nishioka. Perfect keyword privacy in peks systems. In *Provable Security - 6th International Conference, ProvSec 2012*, volume 7496 of *LNCS*, pages 175–192. Springer, 2012.

[OP01] Tatsuaki Okamoto and David Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. In *Public Key Cryptography - PKC 2001*, volume 1992 of *LNCS*, pages 104–118. Springer, 2001.

[PSST11] Kenneth G. Paterson, Jacob C. N. Schuldt, Martijn Stam, and Susan Thomson. On the joint security of encryption and signature, revisited. In *Advances in Cryptology - ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 161–178. Springer, 2011.

[SC11] Jae Hong Seo and Jung Hee Cheon. Fully secure anonymous hierarchical identity-based encryption with constant size ciphertexts. IACR Cryptology ePrint Archive, Report 2011/021, 2011. http://eprint.iacr.org/2011/021.

[Sho01] Victor Shoup. A proposal for an iso standard for public key encryption. IACR Cryptology ePrint Archive, Report 2001/112, 2001. http://eprint.iacr.org/2001/112.

[SK03] Ryuichi Sakai and Masao Kasahara. Id based cryptosystems with pairing on elliptic curve. Cryptology ePrint Archive, Report 2003/054, 2003. http://eprint.iacr.org/2003/054.

[Wat05] Brent Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer, 2005.

[ZI07] Rui Zhang and Hideki Imai. Generic combination of public key encryption with keyword search and public key encryption. In *Cryptology and Network Security, 6th International Conference, CANS 2007*, volume 4856 of *LNCS*, pages 159–174. Springer, 2007.

A Review of Standard Definitions

A.1 Public-Key Encryption with Keyword Search

A non-interactive public-key encryption with keyword search scheme [BDOP04] consists of four PPT algorithms as follows:

- $\mathsf{KeyGen}(\kappa)$: take as input a security parameter κ, output a public/secret key pair (pk, sk). Let W be the set of all possible keywords.
- $\mathsf{Encrypt}(pk, w)$: take as input a public key pk and a keyword $w \in W$, output a ciphertext s.
- $\mathsf{TokenGen}(sk, w)$: take as input a secret key sk and a keyword $w \in W$, output a token t_w.
- $\mathsf{Test}(t_w, s)$: take as input a token t_w and a ciphertext $s \leftarrow \mathsf{Encrypt}(pk, w')$, output 1 if $w' = w$ and 0 otherwise.

The IND-PEKS-CPA security for PEKS schemes is defined by the following experiment:

Setup: $\mathcal{CH}$ runs $\mathsf{KeyGen}(\kappa)$ to generate (pk, sk) and gives $\mathcal{A}$ the public key pk.
Phase 1: $\mathcal{A}$ can adaptively make token queries $\langle w \rangle$. $\mathcal{CH}$ responds with $t_w \leftarrow \mathsf{TokenGen}(sk, w)$.
Challenge: $\mathcal{A}$ outputs two distinct keywords $w_0^*, w_1^* \in W$ subject to the restriction that they had not been asked for tokens in Phase 1. $\mathcal{CH}$ picks a random bit b and sends $c^* \leftarrow \mathsf{Encrypt}(pk, w_b^*)$ to $\mathcal{A}$ as the challenge ciphertext.
Phase 2: $\mathcal{A}$ can adaptively make more token queries $\langle w \rangle$ subject to the restriction that $w \neq w_0^*, w_1^*$. $\mathcal{CH}$ responds the same way as in Phase 1.
Guess: $\mathcal{A}$ outputs a guess b' for b and succeeds if $b' = b$. We denote this event by SuccA and define $\mathcal{A}$'s advantage as $\mathrm{Adv}_{\mathcal{A},\mathrm{PEKS}}^{\mathrm{IND\text{-}CPA}}(\kappa) \overset{\mathrm{def}}{=} |\Pr[\mathsf{SuccA}] - 1/2|$.

Definition A.1. A PEKS scheme is (t, q_w, ϵ) IND-PEKS-CPA secure if for all t-time adversaries making at most q_w token queries have advantage at most ϵ in the above experiment.

The IND-PEKS-CCA security for PEKS schemes can be defined by a similar experiment by giving the adversary access to an additional test oracle which can determine if c is an encryption of w. To avoid triviality, test queries $\langle c^*, w_0^* \rangle$ and $\langle c^*, w_1^* \rangle$ are not allowed in Phase 2.

Definition A.2. A PEKS scheme is (t, q_w, q_t, ϵ) IND-PEKS-CCA secure if for all t-time adversaries making at most q_w token queries and at most q_t test queries have advantage at most ϵ in the IND-PEKS-CCA experiment.

A.2 Hierarchical Identity-Based Encryption

Hierarchical identity-based encryption (HIBE) [HL02, GS02] is a generalization of IBE [BF03] to identities supporting hierarchical structures. In an HIBE scheme, identities are hierarchical and take the form $id = (id_1, id_2, \dots)$. Each user in the hierarchy can act as a local key-generation authority for all subordinate hierarchical identities. An HIBE scheme consists of five PPT algorithms as follows:

- $\mathsf{KeyGen}(\kappa, \ell)$: take as input a security parameter κ and a parameter ℓ for the maximum depth of the HIBE, output a master public/secret key pair (mpk, msk). Let I be the identity space, M be the message space, and C be the ciphertext space. We assume mpk is used as an implicit input for algorithms $\mathsf{Extract}$, Derive, as well as $\mathsf{Decrypt}$,
- $\mathsf{Extract}(msk, id)$: take as input msk and an identity $id \in I$, output a decryption key dk_{id}.[8]
- $\mathsf{Derive}(dk_{id}, id')$: take as input a decryption key dk_{id} for identity $id = (id_1, \dots, id_{j-1})$ of depth $j-1$ and an identity $id' = (id_1, \dots, id_j)$ of depth j, output a decryption key $dk_{id'}$ for id'.
- $\mathsf{Encrypt}(mpk, id, m)$: take as input mpk, an identity $id \in I$, and a message $m \in M$, output a ciphertext $c \in C$.
- $\mathsf{Decrypt}(dk_{id}, c)$: take as input a decryption key dk_{id} for identity id and a ciphertext $c \in C$, output a message $m \in M$ or a reject symbol $\perp$ indicating c is invalid.

The basic security notion for HIBE schemes is indistinguishability against adaptive chosen-plaintext attack (IND-HIBE-CPA), which is defined by the following experiment:

Setup: $\mathcal{CH}$ runs $\mathsf{KeyGen}(\kappa, \ell)$ to generate (pk, sk) and gives $\mathcal{A}$ the master public key mpk.
Phase 1: $\mathcal{A}$ can adaptively make decryption key extraction queries $\langle id \rangle$. $\mathcal{CH}$ responds with $dk_{id} \leftarrow \mathsf{Extract}(msk, id)$.
Challenge: $\mathcal{A}$ outputs two distinct messages m_0^*, m_1^* and an identity id^* subject to the restriction that any prefix of id^* had not been queried for decryption keys in Phase 1. $\mathcal{CH}$ randomly picks a bit b and sends $c^* \leftarrow \mathsf{Encrypt}(mpk, id^*, m_b^*)$ to $\mathcal{A}$ as the challenge ciphertext .
Phase 2: $\mathcal{A}$ can adaptively make more decryption key extraction queries $\langle id \rangle$ subject to the restriction that id is not a prefix of id^*. $\mathcal{CH}$ responds the same way as in Phase 1.
Guess: $\mathcal{A}$ outputs a guess b' for b and succeeds if $b' = b$. We denote this event by SuccA and define $\mathcal{A}$'s advantage as $\mathrm{Adv}_{\mathcal{A},\mathrm{HIBE}}^{\mathrm{IND\text{-}CPA}}(\kappa) \overset{\mathrm{def}}{=} |\Pr[\mathsf{SuccA}] - 1/2|$.

Definition A.3. A HIBE scheme is (t, q_k, ϵ) IND-HIBE-CPA secure if for all t-time adversaries making at most q_k decryption key extraction queries have advantage at most ϵ in the above experiment.

Selective-identity IND-HIBE-CPA security can be defined similarly. The difference is that the adversary has to commit a target identity id^* before seeing mpk and is not allowed to query decryption keys for any prefix of id^* throughout the experiment.

Orthogonal to security, anonymity is introduced to capture identity privacy, which ensures the ciphertext reveals no information about the recipient's identity. To allow a fine-grained

[8]According to the convention of IBE, decryption key dk is usually denoted by sk and referred to as private key. In this work we reserve the symbol sk for secret key of PKE, PEKS, and PKE-PEKS schemes and call private key in IBE schemes as decryption key to avoid confusion.

treatment, Abdalla et al. [ABC$^+$08] defined anonymity regarding to a set L of levels, meaning that in the anonymous experiment the adversary is challenged to distinguished two distinct identities differing only at levels $l \in L$. For ease of notation, we will write l rather than $\{l\}$ when $L = \{l\}$ is a singleton set. Let $\mathsf{diff}(\cdot, \cdot)$ be a function outputting the set of levels at which the two input identities differ. We recall the definition of anonymity for HIBE schemes as follows:

Setup: $\mathcal{CH}$ runs $\mathsf{KeyGen}(\kappa)$ to generate (pk, sk) and gives $\mathcal{A}$ the master public key mpk.
Phase 1: $\mathcal{A}$ can adaptively make decryption key queries $\langle id \rangle$. $\mathcal{CH}$ responds with $dk_{id} \leftarrow \mathsf{Extract}(msk, id)$.
Challenge: $\mathcal{A}$ outputs a message m^* and two distinct identities id_0^*, id_1^* subject to the restrictions that any prefix of id_0^* and id_1^* had not been asked for decryption keys and $\mathsf{diff}(id_0^*, id_1^*) \subset L$. $\mathcal{CH}$ picks a random $b \in \{0, 1\}$ and gives $c^* \leftarrow \mathsf{Encrypt}(mpk, id_b^*, m^*)$ to $\mathcal{A}$ as the challenge ciphertext.
Phase 2: $\mathcal{A}$ can adaptively make more decryption key extraction queries for any identity id subject to the restriction that id is not a prefix of id_0^* or id_1^*. $\mathcal{CH}$ responds the same way as in Phase 1.
Guess: $\mathcal{A}$ outputs a guess b' for b and succeeds if $b' = b$. We denote this event by SuccA and define $\mathcal{A}$'s advantage as $\mathrm{Adv}_{\mathcal{A},\mathrm{HIBE}}^{\text{ANO-CPA}}(\kappa) \stackrel{\text{def}}{=} |\Pr[\mathsf{SuccA}] - 1/2|$.

Definition A.4. A HIBE scheme is (t, q_k, ϵ) ANO-HIBE-CPA $[L]$-anonymous if for all t-time adversaries making at most q_k decryption key extraction queries have advantage at most ϵ in the above experiment.

When considering anonymity for HIBE schemes in the presence of chosen-ciphertext attack, we obtain the ANO-PEKS-CCA security [ABN10], which is defined by a similar experiment as above by giving the adversary additional access to a decryption oracle subject the natural restriction that decryption queries $\langle id_0^*, c^* \rangle$ and $\langle id_1^*, c^* \rangle$ are not allowed in Phase 2.

Definition A.5. A HIBE scheme is (t, q_k, q_d, ϵ) ANO-HIBE-CCA $[L]$-anonymous if for all t-time adversaries making at most q_k decryption key extraction queries and at most q_d decryption queries have advantage at most ϵ in the ANO-HIBE-CCA experiment.

A.3 Signatures

A signature scheme consists of three PPT algorithms as follows:

- $\mathsf{KeyGen}(\kappa)$: take as input a security parameter κ, output a verification key vk and a signing key sk_σ. Let M be the message space.
- $\mathsf{Sign}(sk_\sigma, m)$: take as input a signing key sk_σ and a message $m \in M$, output a signature σ.
- $\mathsf{Verify}(vk, m, \sigma)$: take as input a verification key vk, a message m, and a signature σ, output 1 indicates "acceptance" and 0 indicates "rejection".

For the correctness of a signature scheme, we require that for all $(vk, sk_\sigma) \leftarrow \mathsf{KeyGen}(\kappa)$ and all $m \in M$, $\mathsf{Verify}(vk, m, \mathsf{Sign}(sk_\sigma, m)) = 1$ always holds. If (σ, m) satisfies $\mathsf{Verify}(vk, m, \sigma) = 1$, then σ is said to be a valid signature of message m under the verification key vk.

A strong notion of security for signature schemes is strong existential unforgeability under adaptive chosen-message attack (sEUF-CMA), which is defined by the following experiment:

Setup: $\mathcal{CH}$ runs $\mathsf{KeyGen}(\kappa)$ to generate (vk, sk_σ) and gives $\mathcal{A}$ the verification key vk.
Forgery: $\mathcal{A}$ may do one of the following:

- $\mathcal{A}$ adaptively make signing queries $\langle m_i \rangle$, and is then given in return $\sigma_i \leftarrow \mathsf{Sign}(sk_\sigma, m_i)$. After this, $\mathcal{A}$ outputs (m^*, σ^*).
- $\mathcal{A}$ outputs (m^*, σ^*) without making any signing queries. In this case (m_i, σ_i) are undefined.

We denote the event that $\mathcal{A}$ outputs (m^*, σ^*) such that $\mathsf{Verify}(vk, m^*, \sigma^*) = 1$ but $(m^*, \sigma^*) \neq (m_i, \sigma_i)$ (if (m_i, σ_i) are defined) as SuccA, and define $\mathcal{A}$'s advantage as $\mathrm{Adv}^{\mathrm{sEUF\text{-}CMA}}_{\mathcal{A},\mathrm{SIG}} \overset{\mathrm{def}}{=} \Pr[\mathsf{SuccA}]$.

Definition A.6. A signature scheme is (t, q_s, ϵ) sEUF-CMA secure if for all t-time adversaries making at most q_s signing queries have advantage at most ϵ in the above experiment. Particularly, a signature scheme is one-time strongly unforgeable if it is $(t, 1, \epsilon)$ sEUF-CMA secure.