IACR Communications in Cryptology
ISSN 3006-5496, Vol. 1, No. 2, 28 pages.
https://doi.org/10.62056/a63z11zn4

Public-Key Authenticated Encryption with Keyword Search Made Easy

Qinyi Li[1] and Xavier Boyen[2]

[1] Griffith University, Brisbane, Australia
[2] QUT, Brisbane, Australia

Abstract. Public-key searchable encryption allows keyword-associated tokens to be used to test if a ciphertext contains specific keywords. Due to the low entropies of keywords, the token holder can create ciphertexts from candidate keywords and test them using the token in hand to recover the keywords, known as inside keyword guessing attacks (IKGA). Public-key authenticated encryption with keyword search is a searchable encryption proposed to defend against such attacks. It ensures the sender's private key protects the ciphertexts from the IKGA. PAEKS schemes with reasonable security and practical efficiency remain elusive despite many proposals. This work provides a simple generic PAEKS scheme from non-interactive key exchange (NIKE) and symmetric-key equality-predicate encryption with three new constructions for the latter, respectively from pseudorandom functions (PRFs), the decision bilinear Diffie-Hellman assumption, and the learning-with-errors assumption. Instantiating our generic scheme, we derive several PAEKS schemes from the most well-known assumptions, with some of them achieving full cipher-keyword indistinguishability and full token indistinguishability in the standard model, for the first time. Our instantiated schemes allow practical implementations and outperform the existing PAEKS schemes under the same assumptions.

Keywords: Public-Key Authenticated Encryption · Keywords Search · Post-quantum · Token Privacy · Generic Constructions

1 Introduction

Public-key encryption with keyword search (PEKS) [BCOP04] enables the private key holder to generate a token tk_w for a keyword w. The token can be used by an agent to test if a ciphertext ct_w encrypts the same keyword. However, PEKS only works when the token holding agent is trusted, but not when it is malicious, since the encryption of PEKS can be done publicly, the token holder can test its token against the ciphertext of keywords to know the keywords embedded in the tokens – known as inside keyword guessing attack (IKGA). Due to the low entropy of keyword distributions, IKGA can be very effective.

To defend against IKGA, public-key *authenticated* encryption with keyword search (PAEKS) was introduced and has recently received much attention. PAEKS has limited encryption ability. To create a ciphertext whose keyword can be tested, the sender's private key is needed, i.e., the privacy of the keyword of a ciphertext also relies on the secrecy of the sender's private key. As a result, the testing token holder, who does not have the sender's private key, cannot generate testable ciphertexts, and hence, IKGA is prevented. Albeit being less flexible than the traditional PEKS, PAEKS enables many applications that need malicious agents to locate encrypted content that contains specific keywords.

The security of PAEKS is formalised via a series of works [QCH+20, QCZZ21, LTT+22, CM22, Emu22], primarily focusing on two aspects, namely ciphertext privacy and token

E-mail: qinyi.li@griffith.edu.au (Qinyi Li), xavier.boyen@qut.edu.au (Xavier Boyen)

Received: 2024-04-08 Accepted: 2024-06-03

privacy. The widely accepted security notions for ciphertext privacy and token privacy are full cipher-keyword indistinguishability (FCI) and token indistinguishability (TI) [QCZZ21, LTT+22, Emu22, XWC+24]. The FCI requires that no one can link between ciphertexts from the same keyword. This security notion is defined by allowing the attackers to get ciphertexts (resp. tokens) from any keywords they choose. The TI guarantees token privacy when the attacker sees tokens with different keywords. FCI should be required for practice as ciphertexts on the same keyword may appear often. TI is acceptable as multiple tokens on the same keyword (under the same sender and receiver) may not be needed.

Most PAEKS schemes do not have FCI security, e.g.,[NE19, QCH+20, LHHS22]. Thus, the attacker of these schemes may be able to link two tokens if they are from the same keyword. This is not sufficient for applications of encryption with keyword searches, and we do not discuss these schemes in detail.

A stronger notion for token privacy, named full token indistinguishability (FTI), ensures that the adversary cannot distinguish among tokens generated from the same keyword. As noted by Qin et al. [QCZZ21], a randomised token generation algorithm is necessary to achieve FTI. PAEKS schemes with FTI and FCI are rare: as far as we know, so far, there are only two constructions [CM22, CQFM23]. We note that the multi-user CI and TI notions exist in the literature, which are implied by the FCI and FTI notions [QCZZ21].

1.1 Motivations

Various constructions of PAEKS have been proposed. Most of the existing PAEKS schemes apply a PEKS scheme on encrypted keywords $E_K(w)$ – the ciphertext/token are the PEKS ciphertext/token on $E_K(w)$; the secrets from both the sender and receiver are used to compute K. For example, K is derived by smooth projective hash functions (e.g., Emura [Emu22] and Liu et al. [LTT+22]) or non-interactive key exchange (e.g., Xiang et al. [XWC+24]). The primary disadvantage of this approach is that PEKS schemes are from anonymous identity-based encryption (IBE) [BF01, BW06, ABB10] and smooth projective hash functions (SPHFs), whose instantiations may not be very efficient (e.g., pairing calculations are involved or large public keys used to define certain hard lattices) compared to the commonly used key exchange and encryption. For example, the pairing-based schemes by Qin et al. [QCH+20, QCZZ21] under-perform the BDOP-PEKS [BF01] on which they are based. The lattice-based instantiations of the generic constructions of Liu et al. [LTT+22] and Emura [Emu22] require smooth projective hash functions (e.g., [BBDQ18]) and lattice IBE schemes (e.g., [ABB10, DLP14]).

One exception that deviates from the above approach is the schemes by Cheng and Meng [CM22], which uses tricks unique to certain families of lattices. However, the schemes seem quite inefficient – The concrete LWE parameters provided in [CM22] give only 41-bit security, yet leading to around 260Mb public-key size and around 10Mb ciphertext/token size.[1] To get a reasonable security level, e.g., 100 bits, a substantial blowup of parameter sizes is expected, making the schemes of little use in practice. We would ask:

[RQ1] – "*How can we construct secure and practical PAEKS schemes?*"

The above PEKS-with-encryted-keywords approach and the lattice-based construction by Cheng and Meng have two limitations. First, we cannot use them to construct efficient PAEKS from other well-studied computational problems, e.g., the DH problem, the RSA problem, and the isogeny-based problems, because PEKS does not have constructions from those computational problems. On the other hand, the DH and RSA problems have

[1]The LWE parameters provided include the lattice dimension $n = 256$, number of LWE samples $m = 2^{16}$, the LWE modulus $q = 2^{43}$ and the discrete Gaussian parameter $\sigma = 2^{16}$. The security level was estimated using the LWE problem estimator maintained by Albrecht et al. at https://github.com/malb/lattice-estimator on 16/11/2023.

been well-studied for a long time; isogeny-based cryptography represents a significant and promising direction of post-quantum cryptography. Hence, answering the following question is of both practical and theoretical interest:

> [RQ2] – "*How can we design PAEKS so that it can be instantiated efficiently with various (well-studied or post-quantum) computational assumptions?*"

Second, the aforementioned approaches do not guide constructing PAEKS with the strongest token privacy, i.e., FTI. The techniques used by the two FCI-secure PAEKS schemes seem rather ad-hoc. Hence, the following research question is left open by Emura [Emu22]:

> [RO3] – "*How to provide FCI and FTI security in a generic construction (of PAEKS)?*"

Our work is motivated by the above three research questions.

1.2 Our Contributions

This work builds PAEKS from non-interactive key exchange (NIKE) and symmetric-key equality-predicate encryption (EPE). NIKE allows two users to establish a shared key using each other's public keys without interactions. EPE is a special case of symmetric-key predicate encryption (PE) [SSW09]. EPE supports equality predicates: Let K be the secret key. A query key on keyword w, i.e., dk_w, is generated using K. The ciphertext $\sigma_{w'}$ under the keyword w' is encrypted using K. The query algorithm applies dk_w to $\sigma_{w'}$ and returns one if $w = w'$; otherwise, it returns zero.[2] Our construction is straightforward. To send a ciphertext on keyword w, the sender S computes the NIKE shared key K with the receiver and uses K to generate the EPE ciphertext ct_w as the PAEKS ciphertext. To get the token for w, the receiver runs the NIKE scheme to get the shared key K and uses K to generate the EPE query key dk_w. dk_w is given to a testing agent. Testing applies dk_w, σ_w, and the testing procedure of EPE. Security-wise, we use the standard security definitions for NIKE, which requires that if the attacker does not get one of the private keys of the two target users, the shared key looks random. We define ciphertext indistinguishability, query-key privacy, and weak query-key privacy for EPE. We prove that if the NIKE scheme is secure, and the EPE scheme has ciphertext indistinguishability and query-key privacy (resp. weak query-key privacy), the PAEKS has FCI and FTI (resp. TI).

Our generic construction reduces the problem of constructing efficient PAEKS schemes into the problem of making efficient NIKE and EPE schemes. The former has been well-studied – NIKE can be built with practical efficiency from most commonly used computational problems, including the DH problem, factoring problem, the LWE problem and the isogeny-based problems [DH76, CKS09, FHKP13, CLM+18, dK18, HHK18, GdKQ+24]. We provide several practical constructions for the latter, i.e., EPE, from pseudorandom functions (PRFs), decision bilinear Diffie-Hellman (DBDH) problem, and the learning-with-errors (LWE) problem, in the *standard model.*

Solution to RQ1: Most notably, our NIKE-PRF construction is simple and efficient. It achieves the standard PAEKS security, i.e., FCI and TI. When instantiating the construction using an efficient NIKE scheme and the standard HMAC-based PRFs, the token and ciphertext sizes of the derived PAEKS scheme can be less than 100 bytes for the standard security level, e.g., 128 bits. If the sender and receiver cache the NIKE shared key, encryption, token generation, and testing in the derived PAEKS scheme only involve symmetric-key operations, making our construction at least thousands of times faster than the existing PAEKS schemes. We emphasise that the standard HMAC-based

[2]Note here describes the so-called predicate-only systems [SSW09]. Full encryption can be easily obtained from it but is not needed.

Table 1: PAEKS Schemes with FCI and TI Security ($\geq$ 128-bit Security)

	Parameter Size	Assump.	SM	QR
[QCZZ21]:	$\lvert\mathsf{pk}\rvert$: $1 \times \hat{\mathbb{G}} + 1 \times$ hash (e.g., 96 bytes) $\lvert\mathsf{tk}\rvert$: $1 \times \mathbb{G} + 1 \times$ hash (e.g., 96 bytes) $\lvert\mathsf{ct}\rvert$: $1 \times \mathbb{G} + 1 \times$ hash (e.g., 96 bytes)	DBDH	✗	✗
[Emu22]: ([DLP14]+[BBDQ18])	$\lvert\mathsf{pk}\rvert$: SPHF-$\lvert\mathsf{pk}\rvert$ + IBE-$\lvert\mathsf{pk}\rvert$ ($> 10^8\times$ bytes) $\lvert\mathsf{tk}\rvert$: SPHF-$\lvert\mathsf{sk}\rvert$ + id-$\lvert\mathsf{sk}\rvert$ ($> 10^7$ $\times$ bytes $\lvert\mathsf{ct}\rvert$: SPHF-$\lvert$hash$\rvert$ + IBE-$\lvert\mathsf{ct}\rvert$ ($> 10^7$ $\times$ bytes)	NTRU LWE	✔	✔
[LTT$^+$22]: ([LW19]+[ABB10])	$\lvert\mathsf{pk}\rvert$: SPHF-$\lvert\mathsf{pk}\rvert$ + IBE-$\lvert\mathsf{pk}\rvert$ ($> 10^8\times$ bytes) $\lvert\mathsf{tk}\rvert$: SPHF-$\lvert\mathsf{sk}\rvert$ + id-$\lvert\mathsf{sk}\rvert$ ($> 10^7$ $\times$ bytes $\lvert\mathsf{ct}\rvert$: SPHF-$\lvert$hash$\rvert$ + IBE-$\lvert\mathsf{ct}\rvert$ ($> 10^7$ $\times$ bytes)	NTRU LWE	✔	✔
Ours: ([DH76]+HMAC)	$\lvert\mathsf{pk}\rvert$: $1 \times \mathbb{G}$ (e.g., 64 bytes) $\lvert\mathsf{tk}\rvert$: 32 bytes $\lvert\mathsf{ct}\rvert$: 64 bytes	DDH	✔	✗
Ours: ([CKS09]+HMAC)	$\lvert\mathsf{pk}\rvert$: $2 \times \mathbb{G}$ (e.g., 128 bytes) $\lvert\mathsf{tk}\rvert$: 32 bytes) $\lvert\mathsf{ct}\rvert$: 64 bytes	CDH	✔ (DKR)	✗
Ours: ([FHKP13]+HMAC)	$\lvert\mathsf{pk}\rvert$: $2 \times \mathbb{G} + 1 \times \mathbb{Z}_p$ (e.g., 192 bytes) $\lvert\mathsf{tk}\rvert$: 32 bytes) $\lvert\mathsf{ct}\rvert$: 64 bytes	DBDH	✔	✗
Ours: ([FHKP13]+HMAC)	$\lvert\mathsf{pk}\rvert$: $1 \times \mathbb{Z}_N$ (e.g., 96 bytes) $\lvert\mathsf{tk}\rvert$: 32 bytes) $\lvert\mathsf{ct}\rvert$: 64 bytes	Factoring	✗ (DKR)	✗
Ours: ([CLM$^+$18]+HMAC)	$\lvert\mathsf{pk}\rvert$: $1 \times \mathbb{Z}_p$ (e.g., 64 bytes) $\lvert\mathsf{tk}\rvert$: 32 bytes $\lvert\mathsf{ct}\rvert$: 64 bytes	CSIDH	✔	✔
Ours: ([GdKQ$^+$24]+HMAC)	$\lvert\mathsf{pk}\rvert$: $2 \times R_q^N$ (e.g., $\approx 2.3 \times 10^5$ bytes) $\lvert\mathsf{tk}\rvert$: 32 bytes $\lvert\mathsf{ct}\rvert$: 64 bytes	MLWE	✗	✔
Ours: ([GdKQ$^+$24]+HMAC)	$\lvert\mathsf{pk}\rvert$: $2 \times R_q^N$ (e.g., $\approx 4 \times 10^5$ bytes) $\lvert\mathsf{tk}\rvert$: 32 bytes $\lvert\mathsf{ct}\rvert$: 64 bytes	MLWE	✔	✔

* Notations: $\lvert\mathsf{pk}\rvert$, $\lvert\mathsf{tk}\rvert$, $\lvert\mathsf{ct}\rvert$ denote the sizes of the public key, token, and ciphertext; $\mathbb{G}$ and $\hat{\mathbb{G}}$ are the source and target groups in a bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \hat{\mathbb{G}}$; "SM" and "QR" are abbreviatons of "Standard Model" and "Quantum Resistant"; "HMAC-SHA256" means HAMC uses SHA256 hash function; The integer N in $\mathbb{Z}_N$ denotes the RSA modulus; "MLWE" and "RLWE" denote "Module-LWE' and "Ring-LWE" Assumptions; "DKR" denotes "dishonest key registration" and the DDH/factoring-based NIKE schemes from [CKS09, FHKP13] are secure in the DKR model, see [FHKP13]; $\mathcal{R}_q$ denotes the polynomial ring $\mathbb{Z}_q[x]/(x^n + 1)$ where n is the power of 2; $N = 32$ for [GdKQ$^+$24].

PRFs are naturally quantum-resistant. When using quantum-resistant NIKE schemes, e.g., [CLM$^+$18, dK18, HHK18, GdKQ$^+$24], the derived PAEKS schemes are quantum resistant.

We compare the instantiations of our constructions with the three known PAEKS schemes with FCI and TI [QCZZ21, LTT$^+$22, Emu22] in Table 1. Compared with Qin et al.'s [QCZZ21], our DBDH-based scheme is more efficient and does not require random oracles. Compared with the lattice instantiations of the generic constructions [LTT$^+$22, Emu22], our lattice-based constructions are much more efficient (We couldn't use the concrete parameters provided in [LTT$^+$22] as there are inconsistencies as explained in Appendix C.1), and no concrete parameters are provided in [Emu22]). We estimate the minimum order of parameters using their constructions' underlying lattice primitives, e.g., lattice SPHFs [BBDQ18] and lattice IBE [ABB10, DLP14]). As a result, we provide a solution to RQ1, with post-quantum security.

Solution to RQ2: Meanwhile, Table 1 shows that most commonly used cryptographic assumptions can be used to get PAEKS schemes with FCI and TI, using our generic

Table 2: PAEKS Schemes with FCI and FTI Security ($\geq$ 128-bit Security)

	Parameter Size	Assump.	SM	QR
	$\lvert\mathsf{pk}\rvert$: $\mathbb{G}$ (e.g., 64 bytes)			
[CQFM23]:	$\lvert\mathsf{tk}\rvert$: $3 \times \mathbb{G} + 1 \times \hat{\mathbb{G}}$ (e.g., 256 bytes)	DBDH	✗	✗
	$\lvert\mathsf{ct}\rvert$: $3 \times \mathbb{G} + 1 \times \hat{\mathbb{G}}$ (e.g., 256 bytes)			
CM22[CM22]:	$\lvert\mathsf{pk}\rvert$: $(3m+n) \times \mathbb{Z}_q^n$ ($\approx 2.6 \times 10^8$ bytes)			
(Parameters with	$\lvert\mathsf{tk}\rvert$: $(3m+n) \times \mathbb{Z}_q^m + \mathbb{Z}^{n\times n}$ ($\approx 8.8 \times 10^7$ bytes)	LWE	✗	✔
41-bit security)	$\lvert\mathsf{ct}\rvert$: $(3m+n) \times \mathbb{Z}_q^m + \mathbb{Z}^{n\times n}$ ($\approx 8.8 \times 10^7$ bytes)			
Ours:	$\lvert\mathsf{pk}\rvert$: $1 \times \mathbb{G}$ (e.g., 64 bytes)			
Pairing-EPE	$\lvert\mathsf{tk}\rvert$: $2 \times \mathbb{G}$ (e.g., 192 bytes)	DBDH	✔	✗
+ DH76[DH76]	$\lvert\mathsf{ct}\rvert$: $2 \times \mathbb{G} + 1 \times \hat{\mathbb{G}}$ (e.g., 192 bytes)			
Ours:	$\lvert\mathsf{pk}\rvert$: NIKE-$\lvert\mathsf{pk}\rvert$+ $1 \times \mathcal{R}_q$ ($\approx 2.3 \times 10^5$ bytes)	MLWE		
Lattice-EPE +	$\lvert\mathsf{tk}\rvert$: $3 \times \mathcal{R}_q$ ($\approx 2.3 \times 10^5$ bytes)	NTRU	✗	✔
[GdKQ+24]+[DLP14]	$\lvert\mathsf{ct}\rvert$: $3 \times \mathcal{R}_q$ ($\approx 2.3 \times 10^3$ bytes)			
Ours:	$\lvert\mathsf{pk}\rvert$: NIKE-$\lvert\mathsf{pk}\rvert$+ $1 \times \mathcal{R}_q$ ($\approx 4 \times 10^5$ bytes)	MLWE		
Lattice-EPE +	$\lvert\mathsf{tk}\rvert$: $3 \times \mathcal{R}_q$ ($\approx 4 \times 10^5$ bytes)	NTRU	✔	✔
[dK18]+[DLP14]	$\lvert\mathsf{ct}\rvert$: $3 \times \mathcal{R}_q$ ($\approx 4 \times 10^3$ bytes)			

*Notations: $\lvert\mathsf{pk}\rvert$, $\lvert\mathsf{tk}\rvert$, $\lvert\mathsf{ct}\rvert$ denote the sizes of public key, token, and ciphertext; "SM" and "QR" are abbreviations of "Standard Model" and "Quantum Resistant"; "MLWE" denotes "Module-LWE Assumption"; $\mathcal{R}_q$ denotes the polynomial ring $\mathbb{Z}_q[x]/(x^n+1)$ where n is power of 2; $\mathbb{G}$ and $\hat{\mathbb{G}}$ are the source and target groups in a bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \hat{\mathbb{G}}$.
*The example parameters, except the ones explicitly given in [CM22] are conservative.

construction. Hence, we give a solution to the research question RQ2. In particular, we can base PAEKS on the assumptions that the factoring and the isogeny-based problems are hard. It was unknown how to construct efficient PAEKS schemes from these assumptions.

Solution to RQ3: Our generic construction paves the way to construct PAEKS with FTI, i.e., constructing EPE with query-key privacy. Our concrete constructions of EPE from the DBDH assumption and the LWE assumption demonstrate the applicability of our solution. Combining with secure NIKE schemes (e.g., [DH76, CKS09, FHKP13, dK18, GdKQ+24]) we obtain PAEKS schemes with FCI and FTI from the DBDH assumptions or the LWE assumption, which are more efficient than the existing two PAEKS schemes [CM22, CQFM23], as shown in Table 2. [3]

Besides solving the research questions, our generic constriction can be extended to support more expressive search queries using expressive symmetric-key predicate encryption and can be adapted to identity-based settings with identity-based NIKE. We believe that this work helps shield the light of research on the cryptographic primitive PAEKS.

1.3 Organisation

The rest of the paper is organised as follows. Definitions of the cryptographic primitives are given in Section 2. Section 3 defines PAEKS. Section 4 presents our generic construction of PAEKS and the security proofs. In Section 5, we construct three symmetric-key equality-predicate encryption schemes with security proofs. Section 6 discusses extensions of our approach. We conclude in Section 7.

[3]We note that the oracle DH assumption is used to prove FCI and FTI in [CQFM23]. In Appendix Section C.2, we explain that the DBDH assumption is needed.

2 Preliminaries

2.1 Pseudorandom Functions

Definition 1. Let $\lambda \in \mathbb{N}$ be the security parameter. Let $\mathcal{K}$, $\mathcal{X}$ and $\mathcal{Y}$ be countable sets, $\mathcal{F} = \{f : \mathcal{X} \to \mathcal{Y}\}$ is the set of all functions with domain $\mathcal{X}$ and range $\mathcal{Y}$. A deterministic function $\mathsf{PRF} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is a secure pseudorandom function if PRF can be computed in polynomial time in λ, and for all p.p.t adversary $\mathcal{A}$, $f \leftarrow U(\mathcal{F})$, $K \leftarrow \mathcal{K}$, the advantage

$$\mathsf{Adv}_{\mathsf{PRF},\mathcal{A}}(\lambda) := \left|\Pr[\mathsf{Exp}_{\mathsf{PRF},\mathcal{A}}(\lambda) = 1] - 1/2\right| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}_{\mathsf{PRF},\mathcal{A}}(\lambda)$ is defined in Figure 1.

```
Experiment $\mathsf{Exp}_{\mathsf{PRF},\mathcal{A}}(\lambda)$:
1. $K \leftarrow U(\mathcal{X})$, $\mu \leftarrow U(\{0,1\})$
2. If $\mu = 0$, $\mathcal{O}(\cdot) = \mathsf{PRF}(K, \cdot)$, else, $\mathcal{O}(\cdot) = f(\cdot)$
3. $\mu' \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(1^\lambda)$
4. Return $(\mu' == \mu)$
```

Figure 1: Security Experiment of Pseudorandom Functions

2.2 Non-Interactive Key Exchange (NIKE)

A NIKE scheme Λ with identity space $\mathcal{I}$, shared key space $\mathcal{K}$ has four p.p.t algorithms:

1. $\mathsf{Setup}(1^\lambda)$: On input the security parameter λ, it returns public parameters pub.
2. $\mathsf{Gen}(\mathsf{ID})$: Given an identity $\mathsf{ID} \in \mathcal{I}$, the key generation algorithm returns a public/private key pair $(\mathsf{pk}, \mathsf{sk})$.
3. $\mathsf{Skey}(\mathsf{ID}_1, \mathsf{pk}_1, \mathsf{ID}_2, \mathsf{sk}_2)$: The shared key establishment algorithm Skey takes as two identities $\mathsf{ID}_1, \mathsf{ID}_2 \in \mathcal{I}$ and their corresponding public keys $\mathsf{pk}_1, \mathsf{pk}_2$, and returns a shared key $k \in \mathcal{K}$, or an error symbol $\perp$.[4]

Definition 2. A non-interactive key exchange $\Lambda = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Skey})$ is correct if for all $\mathsf{pub} \leftarrow \mathsf{Setup}(1^\lambda)$ with $\lambda \in \mathbb{N}$, $\mathsf{ID}_1, \mathsf{ID}_2 \in \mathcal{I}$, $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{Gen}(\mathsf{ID}_1)$, $(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow \mathsf{Gen}(\mathsf{ID}_2)$, $K' \leftarrow \mathsf{Skey}(\mathsf{ID}_1, \mathsf{pk}_1, \mathsf{ID}_2, \mathsf{sk}_2)$, $K \leftarrow \mathsf{Skey}(\mathsf{ID}_2, \mathsf{pk}_2, \mathsf{ID}_1, \mathsf{sk}_1)$,

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

where the probability is over the random coins of Setup and Gen.

Cash et al. (CKS) [CKS09] initialise formal definitions for NIKE. We follow Freire et al. [FHKP13] to define NIKE's passive and active security. The active security model, known as the DKR-CKS model, allows the adversary to register public keys that she does not know the corresponding private keys, whereas the passive model, known as the HKR-CKS model, does not allow this.[5] A NIKE scheme secure in the DKR-CKS model is also secure in the HKR-CKS model, and the reverse is not true.

Definition 3. Let λ be the security parameter, $\mathsf{atk} = \{\mathsf{hkr}, \mathsf{dkr}\}$. We say the NIKE scheme Λ is secure if

$$\mathsf{Adv}_{\Lambda,\mathcal{A}}^{\mathsf{atk-cks}}(\lambda) := |\Pr[\mathsf{Exp}_{\Lambda,\mathcal{A}}^{\mathsf{atk-cks}}(\lambda) = 1] - 1/2|$$

is negligible in λ for all p.p.t adversary $\mathcal{A}$, where the experiment $\mathsf{Exp}_{\Lambda,\mathcal{A}}^{\mathsf{atk-cks}}(\lambda)$ is defined in Figure 2 in which the highlighted part is only used by the DKR-CKS model.

[4]It is assumed that Skey returns $\perp$ if $\mathsf{ID}_1 = \mathsf{ID}_2$.

[5]We note that there are multiple versions of the DKR-CKS and HKR-CKS model, which are shown to be polynomially equivalent [FHKP13].

```
Experiment $\mathsf{Exp}^{\mathsf{atk-cks}}_{\Pi,\mathcal{A}}(\lambda)$: // $\mathsf{atk} = \{\mathsf{hkr}, \mathsf{dkr}\}$
1. $\mathsf{pub} \leftarrow \mathsf{Setup}(1^\lambda)$, $\mathcal{L} = \emptyset$
2. $\mu' \leftarrow \mathcal{A}^{\mathsf{RegHon}(\cdot),\ \mathsf{RegCorr}(\cdot),\mathsf{Reavel}(\cdot)\ ,\mathsf{Chall}(\cdot)}(\mathsf{pub})$
3. Return $(\mu' == \mu)$

Oracle $\mathsf{RegHon}(\mathsf{ID})$:
1. If $(corr, \mathsf{ID}, \perp, \cdot) \in \mathcal{L}$, return $\perp$
2. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{ID})$,
   $\mathcal{L} \leftarrow \{(honest, \mathsf{ID}, \mathsf{sk}, \mathsf{pk})\} \cup \mathcal{L}$
3. Return $\mathsf{pk}$

Oracle $\mathsf{RegCorr}(\mathsf{ID}, \mathsf{pk})$:
1. If $(corr, \mathsf{ID}, \perp, \cdot) \in \mathcal{L}$, set $(corr, \mathsf{ID}, \perp, \cdot) =$
   $(corr, \mathsf{ID}, \perp, \mathsf{pk})$
2. Else, $\mathcal{L} \leftarrow \mathcal{L} \cup (corr, \mathsf{ID}, \perp, \mathsf{pk})$
```

```
Oracle $\mathsf{Reveal}(\mathsf{ID}_1, \mathsf{ID}_2)$:
1. If $(honest, \mathsf{ID}_1, \mathsf{sk}_1, \mathsf{pk}_1) \in \mathcal{L}$ and
   $(corr, \mathsf{ID}_2, \perp, \mathsf{pk}_2) \in \mathcal{L}$,
   return $\mathsf{Skey}(\mathsf{ID}_2, \mathsf{pk}_2, \mathsf{ID}_1, \mathsf{sk}_1)$
2. If $(corr, \mathsf{ID}_1, \perp, \mathsf{pk}_1) \in \mathcal{L}$ and
   $(honest, \mathsf{ID}_2, \perp, \perp) \in \mathcal{L}$,
   return $\mathsf{Skey}(\mathsf{ID}_1, \mathsf{pk}_1, \mathsf{ID}_2, \mathsf{sk}_2)$
3. If $(honest, \mathsf{ID}_1, \mathsf{sk}_1, \mathsf{pk}_1) \in \mathcal{L}$ and
   $(honest, \mathsf{ID}_2, \mathsf{sk}_2, \mathsf{pk}_2) \in \mathcal{L}$,
   return $\mathsf{Skey}(\mathsf{ID}_1, \mathsf{pk}_1, \mathsf{ID}_2, \mathsf{sk}_2)$
4. Otherwise, return $\perp$

Oracle $\mathsf{Chall}(\mathsf{ID}_1, \mathsf{ID}_2)$:
1. If $\mathsf{ID}_1 = \mathsf{ID}_2$ or $(corr, \mathsf{ID}_1, \cdot, \cdot) \in \mathcal{L}$
   or $(corr, \mathsf{ID}_2, \cdot, \cdot) \in \mathcal{L}$, return $\perp$
2. $\mu \leftarrow U(\{0,1\})$, $K_1^* \leftarrow U(\mathcal{K})$
3. $K_0^* \leftarrow \mathsf{Skey}(\mathsf{ID}_1, \mathsf{pk}_1, \mathsf{ID}_2, \mathsf{sk}_2)$
4. Return $K_\mu$
```

Figure 2: Security Experiments of NIKE

The Diffie-Hellam key-exchange protocol [DH76] is HKS-CKS secure. DKR-CKS secure NIKE schemes exist from standard assumptions, e.g., factoring [FHKP13], CDH [CKS09], DBDH [FHKP13], and LWE [GdKQ+24].

2.3 Symmetric-Key Equality-Predicate Encryption

This section defines equality-predicate encryption (EPE) with predicate privacy. An EPE system $\Sigma = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Qry})$ with key space $\mathcal{K}$ and keyword space W is defined as follows. Let $\lambda \in \mathbb{N}$ be the security parameter and $K \leftarrow U(\mathcal{K})$.

1. $\mathsf{KG}(K, w)$: The query-key generation algorithm takes as input K and a keywords $w \in \mathcal{W}$, and returns a query key dk_w.

2. $\mathsf{Enc}(K, w)$: The encryption algorithm returns a ciphertext σ_w.

3. $\mathsf{Qry}(\mathsf{dk}_w, \sigma_w)$: The querying algorithm return 1 or 0.

For correctness of Σ, we require for $K \in \mathcal{K}$ and $w = w'$,

$$\Pr[\mathsf{Qry}(\mathsf{KG}(K, w), \mathsf{Enc}(K, w')) \to 1] \geq 1 - \mathsf{negl}(\lambda)$$

where the probability is taken over the randomness used by KG and Enc.

Regarding the security of EPE, we consider formalising two notions: query key privacy and ciphertext indistinguishability.

Definition 4 (Query key privacy)**.** Let λ be the security parameter. We say an EPE scheme $\Sigma = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Qry})$ has query-key privacy if

$$\mathsf{Adv}^{\mathsf{qk-ind}}_{\Sigma,\mathcal{A}}(\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{qk-ind}}_{\Sigma,\mathcal{A}}(\lambda) = 1] - 1/2|$$

is negligible in λ for all p.p.t adversary $\mathcal{A}$. We say Σ has weak query-key privacy if

$$\mathsf{Adv}^{\mathsf{qk-priv}}_{\Sigma,\mathcal{A}}(\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{qk-priv}}_{\Sigma,\mathcal{A}}(\lambda)] = 1 - 1/2|$$

is negligible in λ for all p.p.t adversary $\mathcal{A}$. The experiments $\mathsf{Exp}^{\mathsf{qk-ind}}_{\Sigma,\mathcal{A}}(\lambda)$ and $\mathsf{Exp}^{\mathsf{qk-priv}}_{\Sigma,\mathcal{A}}(\lambda)$ are defined in Figure 3.[6]

[6]Note that weak query-key privacy model prohibits queries $w = w_0^*$ or $w = w_1^*$ to the oracle $\mathsf{K}_K(\cdot)$.

Definition 5 (Ciphertext Indistinguishability)**.** Let λ be the security parameter. We say an EPE scheme $\Sigma = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Qry})$ has ciphertext indistinguishability if

$$\mathsf{Adv}^{\mathsf{ct-ind}}_{\Sigma,\mathcal{A}}(\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{ct-ind}}_{\Sigma,\mathcal{A}}(\lambda)] = 1 - 1/2|$$

is negligible in λ for all p.p.t adversary $\mathcal{A}$, where $\mathsf{Exp}^{\mathsf{ct-ind}}_{\Sigma,\mathcal{A}}(\lambda)$ is defined in Figure 3.

```
Experiment $\mathsf{Exp}^{\mathsf{ct-ind}}_{\Sigma,\mathcal{A}}(\lambda)$:
1. $K \leftarrow U(\mathcal{K})$
2. $(w_0^*, w_1^*) \leftarrow \mathcal{A}^{\mathsf{K}_K(\cdot),\mathsf{E}_K(\cdot)}$
3. $b \leftarrow U(\{0,1\})$
4. $\sigma^* \leftarrow \mathsf{Enc}(K, w_b^*)$
5. $b' \leftarrow \mathcal{A}^{\mathsf{K}_K(\cdot),\mathsf{E}_K(\cdot)}(\sigma^*)$
6. Return $(b' == b)$
Oracle $\mathsf{E}_K(w)$:
1. $\sigma_w \leftarrow \mathsf{Enc}(K, w)$
2. Return $\sigma_w$
Oracle $\mathsf{K}_K(w)$: // $w \neq w_0^*, w_1^*$
1. $\mathsf{dk}_w \leftarrow \mathsf{KG}(K, w)$
2. Return $\mathsf{dk}_w$
```

```
Experiments $\mathsf{Exp}^{\mathsf{qk-ind}}_{\Sigma,\mathcal{A}}(\lambda)$, $\mathsf{Exp}^{\mathsf{qk-priv}}_{\Sigma,\mathcal{A}}(\lambda)$:
1. $K \leftarrow U(\mathcal{K})$
2. $(w_0^*, w_1^*) \leftarrow \mathcal{A}^{\mathsf{K}_K(\cdot),\mathsf{E}_K(\cdot)}$
3. $b \leftarrow U(\{0,1\})$
4. $\mathsf{dk}^* \leftarrow \mathsf{KG}(K, w_b^*)$
5. $b' \leftarrow \mathcal{A}^{\mathsf{K}_K(\cdot),\mathsf{E}_K(\cdot)}(\mathsf{dk}^*)$
6. Return $(b' == b)$
Oracle $\mathsf{E}_K(w)$: // $w \neq w_0^*, w_1^*$
1. $\sigma_w \leftarrow \mathsf{Enc}(K, w)$
2. Return $\sigma_w$
Oracle $\mathsf{K}_K(w)$: // $w \neq w_0^*, w_1^*$
1. $\mathsf{dk}_w \leftarrow \mathsf{KG}(K, w)$
2. Return $\mathsf{dk}_w$
```

Figure 3: Security Experiments for Symmetric-Key EPE

3 Public-Key Authenticated Searchable Encryption

A PAEKS system Π consists of six p.p.t algorithms:

1. $\mathsf{Setup}(1^\lambda)$: The setup algorithm Setup takes as input the security parameter λ and returns public parameter param.

2. $\mathsf{SGen}(\mathsf{param}, S)$: Given the sender's identity S and param, the sender key generation algorithm returns sender's public-/private-key pair $(\mathsf{pk}_S, \mathsf{sk}_S)$.

3. $\mathsf{RGen}(\mathsf{param}, R)$: Given the receiver's identity R and param, the receiver key generation algorithm returns receiver's public-/private-key pair $(\mathsf{pk}_R, \mathsf{sk}_R)$

4. $\mathsf{PAEKS}(\mathsf{sk}_S, \mathsf{pk}_R, w)$: The encryption algorithm takes as input the sender's private key sk_S, receiver's public key pk_R, and a keyword w, and outputs a ciphertext ct_w.

5. $\mathsf{Token}(\mathsf{sk}_R, \mathsf{pk}_s, w)$: The receiver runs the token generation algorithm, which takes as input the receiver's private key sk_R, sender's public key pk_S, and a keyword w, and produces a testing token tk_w. tk_w is sent to the testing server.

6. $\mathsf{Test}(\mathsf{pk}_S, \mathsf{pk}_R, \mathsf{tk}_w, \mathsf{ct}_{w'})$: The testing algorithm inputs the sender's public key and the testing token tk_w, and a ciphertext $\mathsf{ct}_{w'}$, and outputs 1 if $w = w'$, or 0 otherwise.

Definition 6 (Full Cipher-Keyword Indistinguishability (FCI))**.** Let λ be the security parameter. We say that a PAEKS scheme Π has cipher-keyword indistinguishability under full chosen-keyword attacks if

$$\mathsf{Adv}^{\mathsf{fci}}_{\Pi,\mathcal{A}}(\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{fci}}_{\Pi,\mathcal{A}}(\lambda) = 1] - 1/2|$$

```
**Experiment** $\mathsf{Exp}^{\mathsf{fci}}_{\Pi,\mathcal{A}}(\lambda)$:
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$
2. $(\mathsf{pk}_S, \mathsf{sk}_S) \leftarrow \mathsf{SGen}(\mathsf{param}, S)$
3. $(\mathsf{pk}_S, \mathsf{sk}_R) \leftarrow \mathsf{SGen}(\mathsf{param}, R)$
4. $(w_0^*, w_1^*) \leftarrow \mathcal{A}^{\mathcal{OE}_{\mathsf{sk}_S}(\cdot,\cdot),\mathcal{OT}_{\mathsf{sk}_R}(\cdot,\cdot)}(\mathsf{pk}_S, \mathsf{pk}_R)$
5. $b \leftarrow U(\{0,1\})$
6. $\mathsf{ct}^* \leftarrow \mathsf{PAEKS}(\mathsf{sk}_S, \mathsf{pk}_R, w_b^*)$
7. $b' \leftarrow \mathcal{A}^{\mathcal{OE}_{\mathsf{sk}_S}(\cdot,\cdot),\mathcal{OT}_{\mathsf{sk}_R}(\cdot,\cdot)}(\mathsf{pk}_S, \mathsf{pk}_R, \mathsf{ct}^*)$
8. Return $(b' == b)$

Oracle $\mathcal{OE}_{\mathsf{sk}_S}(\mathsf{pk}, w)$:
1. $\mathsf{ct}_w \leftarrow \mathsf{PAEKS}(\mathsf{sk}_S, \mathsf{pk}, w)$
2. Return $\mathsf{ct}_w$

Oracle $\mathcal{OT}_{\mathsf{sk}_R}(\mathsf{pk}, w)$:
// $(\mathsf{pk}, w) \neq (\mathsf{pk}_S, w_0^*)$, $(\mathsf{pk}, w) \neq (\mathsf{pk}_S, w_1^*)$
1. $\mathsf{tk}_w \leftarrow \mathsf{Token}(\mathsf{sk}_R, \mathsf{pk}, w)$
2. Return $\mathsf{tk}_w$
```

```
**Experiments** $\mathsf{Exp}^{\mathsf{fti}}_{\Pi,\mathcal{A}}(\lambda)$, $\boxed{\mathsf{Exp}^{\mathsf{ti}}_{\Pi,\mathcal{A}}(\lambda)}$:
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$
2. $(\mathsf{pk}_S, \mathsf{sk}_S) \leftarrow \mathsf{SGen}(\mathsf{param}, S)$
3. $(\mathsf{pk}_S, \mathsf{sk}_R) \leftarrow \mathsf{SGen}(\mathsf{param}, R)$
4. $(w_0^*, w_1^*) \leftarrow \mathcal{A}^{\mathcal{OE}_{\mathsf{sk}_S}(\cdot,\cdot),\mathcal{OT}_{\mathsf{sk}_R}(\cdot,\cdot)}(\mathsf{pk}_S, \mathsf{pk}_R)$
5. $b \leftarrow U(\{0,1\})$
6. $\mathsf{tk}^* \leftarrow \mathsf{Token}(\mathsf{sk}_R, \mathsf{pk}_S, w_b^*)$
7. $b' \leftarrow \mathcal{A}^{\mathcal{OE}_{\mathsf{sk}_S}(\cdot,\cdot),\mathcal{OT}_{\mathsf{sk}_R}(\cdot,\cdot)}(\mathsf{pk}_S, \mathsf{pk}_R, \mathsf{tk}^*)$
8. Return $(b' == b)$

Oracle $\mathcal{OE}_{\mathsf{sk}_S}(\mathsf{pk}, w)$:
// $(\mathsf{pk}, w) \neq (\mathsf{pk}_R, w_0^*)$, $(\mathsf{pk}, w) \neq (\mathsf{pk}_R, w_1^*)$
1. $\mathsf{ct}_w \leftarrow \mathsf{PAEKS}(\mathsf{sk}_S, \mathsf{pk}, w)$
2. Return $\mathsf{ct}_w$

Oracle $\mathcal{OT}_{\mathsf{sk}_R}(\mathsf{pk}, w)$:
$\boxed{// (\mathsf{pk}, w) \neq (\mathsf{pk}_S, w_0^*),\ (\mathsf{pk}, w) \neq (\mathsf{pk}_S, w_1^*)}$
1. $\mathsf{tk}_w \leftarrow \mathsf{Token}(\mathsf{sk}_R, \mathsf{pk}, w)$
2. Return $\mathsf{tk}_w$
```

Figure 4: FCI, FTI and TI Security Experiments for PAEKS

is negligible in λ, where the experiment $\mathsf{Exp}^{\mathsf{fci}}_{\Pi,\mathcal{A}}(\lambda)$ is defined in Figure 4. Without loss of generality, we assume that the $\mathcal{A}$ does not make query $\mathcal{OE}_{\mathsf{sk}_S}(\mathsf{pk}_R, w_b^*)$ before she sends them for the challenge ciphertext, but can do it after.[7]

Definition 7 (Token Indistinguishability (TI))**.** Let λ be the security parameter. A PAEKS scheme Π has *full* trapdoor indistinguishability under chosen-keyword attacks if

$$\mathsf{Adv}^{\mathsf{fti}}_{\Pi,\mathcal{A}}(\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{fti}}_{\Pi,\mathcal{A}}(\lambda) = 1] - 1/2|$$

is negligible in λ where the experiment $\mathsf{Exp}^{\mathsf{fti}}_{\Pi,\mathcal{A}}(\lambda)$ is defined in Figure 4. Without loss of generality, we assume that the $\mathcal{A}$ did not query w_0^* and w_1^* to $\mathcal{OT}_{\mathsf{sk}_R}(\mathsf{pk}_s, \cdot)$ before she sends them for the challenge token. We say Π has trapdoor indistinguishability under chosen-keyword attacks if

$$\mathsf{Adv}^{\mathsf{ti}}_{\Pi,\mathcal{A}}(\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{ti}}_{\Pi,\mathcal{A}}(\lambda) = 1] - 1/2|$$

is negligible in λ where the experiment $\mathsf{Exp}^{\mathsf{ti}}_{\Pi,\mathcal{A}}(\lambda)$ is defined in Figure 4, which differs from $\mathsf{Exp}^{\mathsf{ti}}_{\Pi,\mathcal{A}}(\lambda)$ by the boxed highlight.

4 The Proposed PAEKS Scheme

We present our new PAEKS scheme based on non-interactive key exchange and symmetric-key equality-predicate encryption.

4.1 The Construction

Our construction, which is given in Figure 5, uses the following building blocks: 1) A NIKE scheme $\Lambda = (\Lambda.\mathsf{Setup}, \Lambda.\mathsf{Gen}, \Lambda.\mathsf{Skey})$ with a share-key space $\mathcal{K}$, and 2) A symmetric-key EPE scheme $\Sigma = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Qry})$ with a key space $\mathcal{K}$ and a keyword space W.

[7]Considering the adversary asks $\mathcal{OE}_{\mathsf{sk}_S}(\mathsf{pk}_R, w_b^*)$ at some point before the challenge point for which there are at most $\mathsf{poly}(\lambda) + 1$ choices. A reduction from this model to our model would simply guess such a point, which would be correct with a non-negligible probability. A correct guess makes the two models equivalent.

```
$\Pi.\mathsf{Setup}(1^\lambda)$:
  1. $\mathsf{pub} \leftarrow \Lambda.\mathsf{Setup}(1^\lambda)$
  2. $\mathsf{param} := \mathsf{pub}$

$\Pi.\mathsf{SGen}(\mathsf{param}, S)$:
  1. $\mathsf{ID} = S$
  2. $(\mathsf{pk}_S, \mathsf{sk}_S) \leftarrow \Lambda.\mathsf{Gen}(\mathsf{ID})$

$\Pi.\mathsf{RGen}(\mathsf{param}, R)$:
  1. $\mathsf{ID} = R$
  2. $(\mathsf{pk}_R, \mathsf{sk}_R) \leftarrow \Lambda.\mathsf{Gen}(\mathsf{ID})$

$\Pi.\mathsf{PAEKS}(\mathsf{sk}_S, \mathsf{pk}_R, w)$:
  1. $K_{RS} \leftarrow \Lambda.\mathsf{Skey}(R, \mathsf{pk}_R, S, \mathsf{sk}_S)$
  2. $\sigma_w \leftarrow \Sigma.\mathsf{Enc}(K_{RS}, w)$
  3. $\mathsf{ct}_w := \sigma_w$

$\Pi.\mathsf{Token}(\mathsf{sk}_R, \mathsf{pk}_S, w)$:
  1. $K_{SR} \leftarrow \Lambda.\mathsf{Skey}(S, \mathsf{pk}_S, R, \mathsf{sk}_R)$
  2. $\mathsf{dk}_w \leftarrow \Sigma.\mathsf{KG}(K_{RS}, w)$
  3. $\mathsf{tk}_w := \mathsf{dk}_w$

$\Pi.\mathsf{Test}(\mathsf{pk}_S, \mathsf{pk}_R, \mathsf{tk}_w, \mathsf{ct}_{w'})$:
  1. Return $\mathsf{Qry}(\mathsf{tk}_w, \mathsf{ct}_{w'})$
```

Figure 5: The Construction of PAEKS Scheme Π

Theorem 1 (Correctness)**.** *The PAEKS scheme* Π *is correct.*

Proof. First of all, $\Lambda.\mathsf{Skey}(R, \mathsf{sk}_R, S, \mathsf{sk}_S) = \Lambda.\mathsf{Skey}(S, \mathsf{pk}_S, R, \mathsf{sk}_R)$ with overwhelming probability due to the correctness of the NIKE scheme. So, we have $K_{RS} = K_{SR}$ with overwhelming probability. Under the condition $K_{RS} = K_{SR} = K$ and $w = w'$,

$$\begin{aligned}\mathsf{Qry}(\mathsf{tk}_w, \mathsf{ct}_{w'}) &= \mathsf{Qry}(\mathsf{dk}_w, \sigma_{w'}) \\ &= \mathsf{Qry}(\Sigma.\mathsf{KG}(K_{SR}, w), \Sigma.\mathsf{Enc}(K_{RS}, w')) \\ &= \mathsf{Qry}(\Sigma.\mathsf{KG}(K, w), \Sigma.\mathsf{Enc}(K, w))) \\ &= 1\end{aligned}$$

holds with overwhelming probability. □

4.2 Security

We prove the following theorem, which states that our proposed scheme has full cipher-keyword indistinguishability.

Theorem 2. *Provided the NIKE scheme* Λ *and the EPE scheme* Σ *are secure, per Definition 3 and Definition 5, the PAEKS scheme* Π *has full cipher-keyword indistinguishability (per Definition 6). More specifically, given a p.p.t adversary* $\mathcal{A}$ *against* Π*, we have*

$$\mathsf{Adv}^{\mathsf{fci}}_{\Pi,\mathcal{A}}(\lambda) \leq 2 \cdot \mathsf{Adv}^{\mathsf{atk-cks}}_{\Lambda,\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{ct-ind}}_{\Sigma,\mathcal{B}_2}(\lambda)$$

for p.p.t algorithms $\mathcal{B}_1$ *and* $\mathcal{B}_2$ *where* $\mathsf{atk} = \{\mathsf{hkr}, \mathsf{dkr}\}$.

We prove the case $\mathsf{atk} = \mathsf{hkr}$ which also proves the case $\mathsf{atk} = \mathsf{dkr}$, because a NIKE scheme secure in DKR-CKS model is also secure in the HKR-CKS model.

Proof. We define two hybrid security games Hyb_0, Hyb_1. At the end of each game, a well-defined binary value is output. We denote the event that the hybrid game Hyb_i outputs 1 as $\mathsf{Hyb}_i \Rightarrow 1$. The hybrid games are defined as follows.

Hyb_0**:** This game is identical to the experiment $\mathsf{Exp}^{\mathsf{fci}}_{\Pi,\mathcal{A}}(\lambda)$ where Π is the proposed PAEKS.

Hyb_1**:** This hybrid game is identical to Hyb_0 except that it changes the way of setting K_{SR} and K_{RS} for the oracles $\mathcal{OE}_{\mathsf{sk}_S}(\mathsf{pk}_R, \cdot)$ and $\mathcal{OT}_{\mathsf{sk}_R}(\mathsf{pk}_S, \cdot)$. In particular, instead of computing K_{RS} and K_{SR} using $\Lambda.\mathsf{Skey}$, a random key is selected and assigned to K_{RS} and K_{SR}, i.e., $K_{RS} := K^*$ and $K_{SR} := K^*$ where $K^* \leftarrow U(\mathcal{K})$.

Let $\mathcal{A}$ be an adversary against Π. Since Hyb_0 is identical to $\mathsf{Exp}^{\mathsf{fci}}_{\Pi,\mathcal{A}}(\lambda)$, we have

$$\mathsf{Adv}^{\mathsf{fci}}_{\Pi,\mathcal{A}}(\lambda) + 1/2 = \Pr[\mathsf{Hyb}_0 \Rightarrow 1] \tag{1}$$

The only difference between Hyb_0 and Hyb_1 is how the shared key K_{RS} (and K_{SR}) is set. We show this difference can be turned into a security breach of the NIKE system. We build an algorithm $\mathcal{B}_1$ that simulates Hyb_0 or Hyb_1 to the adversary $\mathcal{A}$ as follows.

1. $\mathcal{B}_1$ receives pub from the NIKE challenger, and publishes $\mathsf{param} := \mathsf{pub}$. Then, $\mathcal{B}_1$ initialises an empty set $\mathcal{L}$. Moreover, it chooses the sender identity S and receiver identity R, calls $\mathsf{RegHon}(S)$ and $\mathsf{RegHon}(R)$ to get pk_S and pk_R, calls the oracle $\mathsf{Chall}(S, R)$ to receive K^*_μ, sets $K_{SR} = K_{RS} = K^*_\mu$, and publishes pk_S and pk_R.

2. $\mathcal{B}_1$ answers queries $\mathcal{OE}_{\mathsf{sk}_S}(\mathsf{pk}_{R'}, w)$ as follows: If $R' = R$, $\mathcal{B}_1$ returns $\mathsf{ct}_w := \sigma_w \leftarrow \Sigma.\mathsf{Enc}(K^*_\mu, w)$. Otherwise, if $R' \neq R$, $\mathcal{B}_1$ looks up $\mathcal{L}$: if $(R', \mathsf{pk}_{S'}, \mathsf{sk}_{S'}) \notin \mathcal{L}$, $\mathcal{B}_1$ runs $(\mathsf{pk}_{R'}, \mathsf{sk}_{R'}) \leftarrow \Lambda.\mathsf{Gen}(\mathsf{param}, R')$, and updates $\mathcal{L} \leftarrow \mathcal{L} \cup (R', \mathsf{pk}_{R'}, \mathsf{sk}_{R'})$. Then, $\mathcal{B}_1$ sets $K_{R'S} := \Lambda.\mathsf{Skey}(S, \mathsf{pk}_S, R, \mathsf{sk}_{R'})$ [8], and returns $\mathsf{ct}_w := \sigma_w \leftarrow \Sigma.\mathsf{Enc}(K_{R'S}, w)$.

3. $\mathcal{B}_1$ answers oracle queries $\mathcal{OT}_{\mathsf{sk}_R}(\mathsf{pk}_{S'}, w)$ as follows: If $S' = S$, $\mathcal{B}_1$ returns $\mathsf{tk}_w := \mathsf{dk}_w \leftarrow \Sigma.\mathsf{KG}(K^*_\mu, w)$. Otherwise, if $S' \neq S$, $\mathcal{B}_1$ looks up $\mathcal{L}$: if $(S', \mathsf{pk}_{S'}, \mathsf{sk}_{S'}) \notin \mathcal{L}$, $\mathcal{B}_1$ runs $(\mathsf{pk}_{S'}, \mathsf{sk}_{S'}) \leftarrow \Lambda.\mathsf{Gen}(\mathsf{param}, S')$, and updates $\mathcal{L} \leftarrow \mathcal{L} \cup (S', \mathsf{pk}_{S'}, \mathsf{sk}_{S'})$. Next, $\mathcal{B}_1$ sets $K_{S'R} := \mathsf{Skey}(R, \mathsf{pk}_R, S', \mathsf{sk}_{S'})$, and returns $\mathsf{tk}_w := \mathsf{dk}_w \leftarrow \Sigma.\mathsf{KG}(K_{S'R}, w)$.

4. When $\mathcal{A}$ issues two different keywords w^*_0 and w^*_1, $\mathcal{B}_1$ picks $b \leftarrow U(\{0,1\})$, and returns $\mathsf{ct}^* \leftarrow \mathcal{OE}_{\mathsf{sk}_S}(\mathsf{pk}_R, w^*_b)$.

5. Further oracle queries to $\mathcal{OE}_{\mathsf{sk}_S}(\cdot,\cdot)$ and $\mathcal{OT}_{\mathsf{sk}_R}(\cdot,\cdot)$ from $\mathcal{A}$ are answered as before. Finally, $\mathcal{B}_1$ outputs 1 if $\mathcal{A}$ outputs $b' = b$; Else, $\mathcal{B}_1$ outputs 0.

We analyse the simulation. It is easy to see param, the public keys created by $\mathcal{B}_1$ have the correct distributions, i.e., distributed as in the real system. We consider two situations.

When $\mu = 0$, i.e., $K^*_\mu = K^*_1 \leftarrow \mathsf{Skey}(S, \mathsf{pk}_S, R, \mathsf{sk}_R)$. Since Λ is correct, $K_{SR} = \mathsf{Skey}(S, \mathsf{pk}_S, R, \mathsf{sk}_R) = \mathsf{Skey}(R, \mathsf{pk}_R, S, \mathsf{sk}_S) = K_{RS} = K^*_1$ except with negligible probability. Hence, the oracle responses $\mathcal{OE}_{\mathsf{sk}_S}(\mathsf{pk}_R, w)$ and $\mathcal{OT}_{\mathsf{sk}_R}(\mathsf{pk}_S, w)$ which uses K^*_1 are distribute as in Hyb_0. For the same reason, the oracle responses $\mathcal{OE}_{\mathsf{sk}_S}(\mathsf{pk}_{R'}, w)$ with $R' \neq R$, which are simulated using $K_{R'S} \leftarrow \Lambda.\mathsf{Skey}(S, \mathsf{pk}_S, R', \mathsf{sk}_{R'})$, and oracle responses $\mathcal{OT}_{\mathsf{sk}_R}(\mathsf{pk}_{S'}, w)$ with $S' \neq S$, which are simulated using $K_{S'R} \leftarrow \mathsf{Skey}(R, \mathsf{pk}_R, S', \mathsf{sk}_{S'})$ are distributed as in Hyb_0, except with a negligible probability that is bounded by the correctness of Λ.

Second, we consider $\mu = 1$, i.e., $K^*_\mu = K^*_1 \leftarrow U(\mathcal{K})$. The responses to the oracles queries $\mathcal{OE}_{\mathsf{sk}_S}(\mathsf{pk}_R, w)$ and $\mathcal{OT}_{\mathsf{sk}_R}(\mathsf{pk}_S, w)$ are distributed as in Hyb_1.

Let E be the event that at least one of the shared keys produced by $\mathcal{OE}_{\mathsf{sk}_S}(\mathsf{pk}_{R'}, w)$ and $\mathcal{OT}_{\mathsf{sk}_R}(\mathsf{pk}_{S'}, w)$, i.e., $K_{R'S}$ and $K_{S'R}$, do not match with the their counterparts, i.e., $K_{SR'}$ and $K_{RS'}$ (including $S' = S$ and $R' = R$) due to the correctness failure of $\Lambda.\mathsf{Skey}$. By the correctness of Λ, $\Pr[\neg E] \leq 1 - \mathsf{negl}'(\lambda)$ where $\mathsf{negl}'(\lambda)$ is negligible in λ. We have

$$\begin{aligned}\Pr[\mathsf{Hyb}_0 \Rightarrow 1] &= \Pr[(b' = b|\mu = 0) \wedge \neg E]\\ &= \Pr[b' = b|\mu = 0, \neg E] - \mathsf{negl}''(\lambda)\\ &= \Pr[\mathcal{B}_1 \Rightarrow 1|\mu = 0] - \mathsf{negl}''(\lambda)\end{aligned}$$

where $\mathsf{negl}''(\lambda) = \mathsf{negl}'(\lambda) \cdot \Pr[\mathcal{B}_1 \Rightarrow 1|\mu = 0, \neg E]$ is negligible. This leads to

$$\begin{aligned}&|\Pr[\mathsf{Hyb}_1 \Rightarrow 1] - \Pr[\mathsf{Hyb}_0 \Rightarrow 1]| \qquad (2)\\ =&|\Pr[\mathcal{B}_1 \Rightarrow 1|\mu = 0] - \Pr[\mathcal{B}_1 \Rightarrow 1|\mu = 1] - \mathsf{negl}''(\lambda)|\\ =&2 \cdot |1/2 - \Pr[(\mathsf{Exp}^{\mathsf{hkr-cks}}_{\Lambda,\mathcal{B}_1}(\lambda) = 1)]| - 1/2 \cdot \mathsf{negl}''(\lambda)\\ =&2 \cdot \mathsf{Adv}^{\mathsf{hkr-cks}}_{\Lambda,\mathcal{B}_1}(\lambda) - \mathsf{negl}(\lambda)\end{aligned}$$

Next, we efficiently reduce the attack against the EPE Σ to an attack to Hyb_1. We construct a p.p.t algorithm $\mathcal{B}_2$ which uses adversary $\mathcal{A}$ to break ciphertext indistinguishability of Σ as follows.

[8]In the real scheme, the algorithm $\Pi.\mathsf{PAEKS}$ uses the sender S's private key for $\Lambda.\mathsf{Skey}$ to get $K_{SR'}$

1. $\mathcal{B}_2$ sets $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{pk}_S, \mathsf{sk}_S) \leftarrow \mathsf{SGen}(\mathsf{param}, S)$, $(\mathsf{pk}_R, \mathsf{sk}_R) \leftarrow \mathsf{SGen}(\mathsf{param}, R)$, sends $(\mathsf{pk}_S, \mathsf{pk}_R)$ to $\mathcal{A}$. $\mathcal{B}_2$ then initialises an empty set $\mathcal{L}$.
2. $\mathcal{B}_2$ answers $\mathcal{A}$'s query $\mathcal{OE}_{\mathsf{sk}_S}(\mathsf{pk}_{R'}, w)$ as follows: If $R' = R$, $\mathcal{B}_2$ makes an oracle query $\mathsf{E}_K(w)$ to get σ_w and returns $\mathsf{ct}_w := \sigma_w$. Otherwise, if $R' \neq R$, $\mathcal{B}_2$ looks up $\mathcal{L}$: if $(R', \mathsf{pk}_{S'}, \mathsf{sk}_{S'}) \notin \mathcal{L}$, $\mathcal{B}_2$ runs $(\mathsf{pk}_{R'}, \mathsf{sk}_{R'}) \leftarrow \Lambda.\mathsf{Gen}(\mathsf{param}, R')$ and updates $\mathcal{L} \leftarrow \mathcal{L} \cup (R', \mathsf{pk}_{R'}, \mathsf{sk}_{R'})$. $\mathcal{B}_2$ sets $K_{R'S} := \mathsf{Skey}(S, \mathsf{pk}_S, R', \mathsf{sk}_{R'})$, and returns $\mathsf{ct}_w := \sigma_w \leftarrow \Sigma.\mathsf{Enc}(K_{R'S}, w)$.
3. $\mathcal{B}_2$ answers $\mathcal{A}$'s query $\mathcal{OT}_{\mathsf{sk}_R}(\mathsf{pk}_{S'}, w)$ as follows: If $S' = S$, $\mathcal{B}_2$ calls the oracle $\mathsf{K}_K(w)$ to get $\mathsf{dk}_w \leftarrow \mathsf{K}_K(w)$ and returns $\mathsf{tk}_w := \mathsf{dk}_w$. Otherwise, if $S' \neq S$, $\mathcal{B}_2$ looks up $\mathcal{L}$: if $(S', \mathsf{pk}_{S'}, \mathsf{sk}_{S'}) \notin \mathcal{L}$, $\mathcal{B}_2$ runs $(\mathsf{pk}_{S'}, \mathsf{sk}_{S'}) \leftarrow \Lambda.\mathsf{Gen}(\mathsf{param}, S')$ and updates $\mathcal{L} \leftarrow \mathcal{L} \cup (S', \mathsf{pk}_{S'}, \mathsf{sk}_{S'})$. $\mathcal{B}_2$ computes $K_{S'R} := \mathsf{Skey}(R, \mathsf{pk}_R, S', \mathsf{sk}_{S'})$, $\mathsf{dk}_w \leftarrow \Sigma.\mathsf{KG}(K_{S'R}, w)$, and returns $\mathsf{tk}_w := \mathsf{dk}_w$.
4. When $\mathcal{A}$ issues two different keywords w_0^* and w_1^*, $\mathcal{B}_2$ passes them through to its own challenger and receives back $\sigma_b^* \leftarrow \mathsf{Enc}_K(w_b^*)$ where $b \leftarrow U(\{0,1\})$. $\mathcal{B}_2$ then forward $\mathsf{ct}^* := \sigma_b^*$ to $\mathcal{A}$.
5. For further queries to $\mathcal{OE}_{\mathsf{sk}_S}(\cdot,\cdot)$ and $\mathcal{OT}_{\mathsf{sk}_R}(\cdot,\cdot)$ are answered as before. Finally, when $\mathcal{A}$ outputs μ', $\mathcal{B}_2$ outputs $b' = \mu'$.

We analyse the simulation. It is easy to see that param and all public keys available to $\mathcal{A}$ are distributed properly as they are in Hyb_1. In this simulation, $\mathcal{B}_2$ implicitly set the shared key between the users S and R, i.e., K_{SR} and K_{RS}, as some (unknown) key K (which is chosen uniformly at random by the EPE challenger) as required in Hyb_1. Therefore, the output distributions of oracles $\mathcal{OE}_{\mathsf{sk}_S}(\mathsf{pk}_{R'}, \cdot)$, $\mathcal{OT}_{\mathsf{sk}_R}(\mathsf{pk}_{S'}, \cdot)$ are correct, as in Hyb_1. So, $\mathcal{B}_2$ simulates Hyb_1 correctly to $\mathcal{A}$. Hence,

$$\Pr[\mathsf{Hyb}_1 \Rightarrow 1] = \Pr[\mathsf{Exp}^{\mathsf{ct-ind}}_{\Sigma,\mathcal{B}_2}(\lambda) = 1] \tag{3}$$

We conclude the proof of the theorem by combining the inequalities (1), (2), and (3) and the definition of $\mathsf{Adv}^{\mathsf{ct-ind}}_{\Sigma,\mathcal{B}_2}(\lambda)$. □

Theorem 3. *Provided the NIKE scheme Λ is secure, and the EPE scheme has query-key privacy (resp. weak query-key privacy), the PAEKS scheme Π has full trapdoor indistinguishability (resp. trapdoor indistinguishability). More specifically, given any p.p.t adversary $\mathcal{A}$ against Π, we have for* $\mathsf{atk} = \{\mathsf{hkr}, \mathsf{dkr}\}$

$$\mathsf{Adv}^{\mathsf{fci}}_{\Pi,\mathcal{A}}(\lambda) \leq 2 \cdot \mathsf{Adv}^{\mathsf{atk-cks}}_{\Lambda,\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{ct-ind}}_{\Sigma,\mathcal{B}_2}(\lambda)$$

and

$$\mathsf{Adv}^{\mathsf{ti}}_{\Pi,\mathcal{A}}(\lambda) \leq 2 \cdot \mathsf{Adv}^{\mathsf{atk-cks}}_{\Lambda,\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{qk-priv}}_{\Sigma,\mathcal{B}'_2}(\lambda)$$

for some p.p.t algorithms $\mathcal{B}_1$, $\mathcal{B}_2$ and $\mathcal{B}'_2$.

We prove the case $\mathsf{atk} = \mathsf{hkr}$ which proves the case $\mathsf{atk} = \mathsf{dkr}$.

Proof. We define two hybrid security games Hyb_0 and Hyb_1. At the end of each game, a well-defined binary value is returned. Let $\mathsf{Hyb}_i \Rightarrow 1$ the event that Hyb_i returns 1. We defined Hyb_0 and Hyb_1 as follows.

Hyb_0**:** This game is identical to the experiment $\mathsf{Exp}^{\mathsf{fti}}_{\Pi,\mathcal{A}}(\lambda)$ (resp. $\mathsf{Exp}^{\mathsf{ti}}_{\Pi,\mathcal{A}}(\lambda)$) where Π is the proposed PAEKS.

Hyb_1**:** This hybrid game is identical to Hyb_0 except that way of setting K_{SR} and K_{RS} for the oracles $\mathcal{OE}_{\mathsf{sk}_S}(\mathsf{pk}_R, \cdot)$ and $\mathcal{OT}_{\mathsf{sk}_R}(\mathsf{pk}_S, \cdot)$. In particular, instead of computing K_{RS} and K_{SR} using $\Lambda.\mathsf{Skey}$, a random key is selected and assigned to K_{RS} and K_{SR}, i.e., $K_{RS} := K^*$ and $K_{SR} := K^*$ where $K^* \leftarrow U(\mathcal{K})$.

Given an adversary $\mathcal{A}$ against Π, we have

$$\mathsf{Adv}^{\mathsf{fti}}_{\Pi,\mathcal{A}}(\lambda) + 1/2 = \Pr[\mathsf{Hyb}_0 \Rightarrow 1] \tag{4}$$

Hyb_1 differs from Hyb_0 in how to set the shared key between the prescribed identities S and R: In Hyb_0, the shared keys are computed as $K_{SR} \leftarrow \Lambda.\mathsf{Skey}(R, \mathsf{sk}_R, S, \mathsf{sk}_S)$ and $K_{RS} \leftarrow \Lambda.\mathsf{Skey}(S, \mathsf{pk}_S, R, \mathsf{sk}_R)$. By the correctness of $\Lambda.\mathsf{Skey}$, $K_{SR} = K_{RS}$ with overwhelming probability. In Hyb_1, a random shared key is chosen from $\mathcal{K}$, the output space of $\Lambda.\mathsf{Skey}$ and its value is assigned to K_{SR} and K_{RS}. Telling this difference amounts to differentiating a shared key generated by $\Lambda.\mathsf{Skey}$ using non-corrupted users from a random key. So, using the same process in the proof of Theorem 2, we can get

$$\Pr[\mathsf{Hyb}_1 \Rightarrow 1] - \Pr[\mathsf{Hyb}_0 \Rightarrow 1] \le \mathsf{Adv}^{\mathsf{hkr-cks}}_{\Lambda,\mathcal{B}_1}(\lambda) \tag{5}$$

for some p.p.t adversary $\mathcal{B}_1$ against Λ.

In Hyb_1, for the prescribed identities S and R, the behaviours of the oracles $\mathcal{OT}_{\mathsf{sk}_R}(\mathsf{pk}_S, \cdot)$ and $\mathcal{OE}_{\mathsf{sk}_S}(\mathsf{pk}_R, \cdot)$ are identical to the behaviours of the oracles $\mathsf{K}_{K^*}(\cdot)$ and $\mathsf{E}_{K^*}(\cdot)$, respectively. In particular, they have the same restrictions on queries, e.g., whether the challenge keywords are allowed to be queried. Moreover, the challenge trapdoor of the PAEKS system Π was generated using $\mathcal{OT}_{\mathsf{sk}_R}(\mathsf{pk}_S, \cdot)$, and hence $\mathsf{K}_{K^*}(\cdot)$. So, breaking the challenges produced by $\mathcal{OT}_{\mathsf{sk}_R}(\mathsf{pk}_S, \cdot)$ in Hyb_1 amounts to breaking the challenges produced by $\mathsf{K}_{K^*}(\cdot)$. So, we obtain

$$\Pr[\mathsf{Hyb}_1 \Rightarrow 1] = \Pr[\mathsf{Exp}^{\mathsf{qk-ind}}_{\Sigma,\mathcal{B}_2}(\lambda) = 1] \tag{6}$$

for some adversary $\mathcal{B}_2$. Combining inequalities (4), (5), and (6) concludes the proof. $\square$

5 Constructions of EPE

To instantiate our generic PAEKS scheme, we provide three practical constructions of the symmetric-key equality-predicate encryption. They are based on pseudorandom functions (PRFs), the decision bilinear Diffie-Hellman (DBDH) problem, and the learning-with-errors problem over rings. All of them have ciphertext indistinguishability. The PRF-based construction has weak query-key privacy, while the other two have query-key privacy. The PRF-based construction is very efficient and, combined with the state-of-the-art NIKE scheme, outperforms all previous PAEKS schemes.

5.1 Construction from Pseudorandom Functions

We construct EPE based on pseudorandom functions (PRFs).

5.1.1 The Construction

Our constriction uses two PRFs: $\mathsf{PRF}_1 : \mathcal{K} \times \mathcal{W} \to \mathcal{K}'$ and $\mathsf{PRF}_2 : \mathcal{K}' \times \mathcal{R} \to \mathcal{S}$. Let the symmetric key $K \leftarrow U(\mathcal{K})$. The construction is given below.

– $\mathsf{KG}(K, w)$: Returns $\mathsf{dk} := K_w \leftarrow \mathsf{PRF}_1(K, w)$.

– $\mathsf{Enc}(K, w)$: Sets $K_w \leftarrow \mathsf{PRF}_1(K, w)$, $r \leftarrow U(\mathcal{R})$, $\sigma := (t \leftarrow \mathsf{PRF}_2(K_w, r), r)$.

– $\mathsf{Qry}(\mathsf{dk}, \sigma)$: Returns 1 if $t = \mathsf{PRF}_2(K_w)$; Otherwise, return 0.

It is easy to see that the construction satisfies correctness.

5.1.2 Security

We prove weak query-key privacy and ciphertext indistinguishability. Because the query-key generation algorithm is deterministic, the above construction cannot achieve stronger query-key privacy.

Theorem 4. *Our construction of EPE based on PRFs has ciphertext indistinguishability if the PRFs are secure.*

Proof. Let $\mathcal{A}$ be the adversary. We define four hybrid games Hyb_i for $i = \{0, 1, 2, 3\}$. At the end of each game, a well-defined binary value is output. The hybrid games are:

Hyb_0**:** Hyb_0 is identical to the experiment $\mathsf{Adv}^{\mathsf{ct-ind}}_{\Sigma,\mathcal{A}}(\lambda)$.

Hyb_1**:** Hyb_1 is identical to Hyb_0 except the that an empty list $\mathcal{L}$ is set, and oracles $\mathsf{K}_K(\cdot)$, $\mathsf{E}_K(\cdot)$ are described in Figure 6. To generate the challenge on w_b^*, instead of running $\mathsf{Enc}(K, w_b^*)$, it proceeds by choosing a random PRF key $K_{w_b^*} \leftarrow U(\mathcal{K}')$, updating $\mathcal{L} \leftarrow \mathcal{L} \cup \{(w_b^*, K_{w_b^*})\}$, and returning $\sigma_b^* := (t \leftarrow \mathsf{PRF}_2(K_{w_b^*}, r))$ where $r \leftarrow U(\mathcal{R})$.[9]

Hyb_2**:** Hyb_2 is identical to Hyb_1 except that on the challenge input w_0^*, the oracle $\mathsf{E}_K(w_0^*)$ samples $r \leftarrow U(\mathcal{R})$, $t \leftarrow U(S)$ and returns $\sigma_0^* := (t, r)$.

Hyb_3**:** Hyb_3 is identical to Hyb_2 except that on the challenge input w_1^*, the oracle $\mathsf{E}_K(w_1^*)$ samples $r \leftarrow U(\mathcal{R})$, $t \leftarrow U(S)$ and returns $\sigma_1^* := (t, r)$.

```
Oracle $\mathsf{K}_K(w)$: // $w \neq w_0^*, w_1^*$
1. Return $K_w$ if $(w, K_w) \in \mathcal{L}$
2. $K_w \leftarrow U(\mathcal{K}')$
3. $\mathcal{L} \leftarrow \mathcal{L} \cup \{(w, K_w)\}$
4. Return $\mathsf{dk}_w := K_w$

Oracle $\mathsf{E}_K(w)$:
1. $K_w \leftarrow \mathsf{K}_K(w)$
2. $r \leftarrow U(\mathcal{R})$
3. $t \leftarrow \mathsf{PRF}_2(K_w, r)$
4. Return $\sigma_w := (t, r)$
```

Figure 6: Oracles $\mathsf{K}_K(\cdot)$, $\mathsf{E}_K(\cdot)$ in Hyb_1, proof of Theorem 4

As Hyb_0 is the same as $\mathsf{Exp}^{\mathsf{ct-ind}}_{\Sigma,\mathcal{A}}(\lambda)$. By definition,

$$\Pr[\mathsf{Hyb}_0 \Rightarrow 1] = \Pr[\mathsf{Exp}^{\mathsf{ct-ind}}_{\Sigma,\mathcal{A}}(\lambda) = 1] \tag{7}$$

The only difference between Hyb_1 and Hyb_0 is that in Hyb_0, K_w is computed by $\mathsf{PRF}_1(K, w)$ while in Hyb_1, K_w is chosen uniformly at random. We construct an algorithm $\mathcal{B}_1$ who has access to oracle $\mathcal{O}$ and needs to decide if $\mathcal{O} = \mathsf{PRF}_1(K)$ (when $\mu = 0$) or $\mathcal{O} = f$ where $f : W \rightarrow \mathcal{K}'$ is a random function (when $\mu = 1$).

$\mathcal{B}_1$ follows Hyb_1 except simulating $\mathsf{K}_K(w)$ as follows: If $(w, K_w) \in \mathcal{L}$, returns K_w, otherwise, calls $K_w \leftarrow \mathcal{O}(w)$, sets $\mathcal{L} \leftarrow \mathcal{L} \cup (w, K_w)$, and returns $\mathsf{dk}_w := K_w$. Note that this change also modifies the oracle $\mathsf{E}_K(w)$, which uses $\mathsf{K}_K(w)$ as a subroutine. Finally, when $\mathcal{B}_1$ outputs 0 if $\mathcal{A}$ outputs $b' = b$; otherwise, $\mathcal{B}_1$ outputs 1.

It can be seen that if $\mathcal{O}(\cdot) = \mathsf{PRF}(K, \cdot)$ (for some random $K \in \mathcal{K}$ unknown to $\mathcal{B}_1$), $\mathcal{B}_1$

[9]Note, the way that σ_b^* was generated in the same way as $\mathsf{E}_K()$. However, we cannot say we use $\mathsf{E}_K(w_b^*)$ as a step to create the challenge as $\mathsf{E}_K(\cdot)$ uses $\mathsf{K}_K(\cdot)$ as a subroutine and we specified w_b^* cannot be used to query $\mathsf{K}_K(\cdot)$.

simulates Hyb_0 to $\mathcal{A}$; otherwise, $\mathcal{B}_1$ simulates Hyb_1. We have

$$\begin{aligned}
&|\Pr[\mathsf{Hyb}_1 \Rightarrow 1] - \Pr[\mathsf{Hyb}_0 \Rightarrow 1]| \\
=&|\Pr[\mathcal{A} \Rightarrow 1|\mu = 1] - \Pr[\mathcal{A} \Rightarrow 1|\mu = 0]| \\
=&|\Pr[\mathcal{B}_1 \Rightarrow 1|\mu = 1] - \Pr[\mathcal{B}_1 \Rightarrow 1|\mu = 0]| \\
=&|\Pr[\mathcal{B}_1 \Rightarrow 1|\mu = 1] - (1 - \Pr[\mathcal{B}_1 \Rightarrow 0|\mu = 0])| \\
=&2 \cdot |\Pr[\mathsf{Exp}_{\mathsf{PRF}_1,\mathcal{B}_1}(\lambda) = 1] - 1/2| \\
=&2 \cdot \mathsf{Adv}_{\mathsf{PRF}_1,\mathcal{B}_1}(\lambda)
\end{aligned} \tag{8}$$

Hyb_2 differs from Hyb_1 by generating a random tag σ_0^*. In Hyb_1, σ_0^* is computed using PRF_2 with key $\mathsf{PRF}_1(K, w_0^*)$. We build an algorithm $\mathcal{B}_2$ that uses the difference between Hyb_1 and Hyb_2 to break PRF_2.

$\mathcal{B}_2$ has access to oracle $\mathcal{O}$ and needs to decide if $\mathcal{O} = \mathsf{PRF}_2(K_{w_0^*},\cdot)$ (when $\mu = 0$) or $\mathcal{O} = f$ where $f : \mathcal{R} \to \mathcal{S}$ is a random function (when $\mu = 1$). Note that $K_{w_0^* \in \mathcal{K}'}$ is random and unknown to $\mathcal{B}_2$. $\mathcal{B}_2$ follows Hyb_1 except simulating $\mathsf{E}_K(w_0^*)$ after received the challenge keyword w_0^*: $\mathcal{B}_2$ samples $r \leftarrow U(\mathcal{R})$, calls $\mathcal{O}(w_0^*)$ to get t and set $\sigma_0^* := (t, r)$. Finally, $\mathcal{B}_2$ outputs 0 if $\mathcal{A}$'s output $b' = b$, otherwise, $\mathcal{B}_2$ outputs 1.

We can see that $\mathcal{B}_2$ cannot reply to the query $\mathsf{K}_K(w_0^*)$ which should be equal to $K_{w_0^*}$. However, $\mathcal{A}$ is not allowed to make such a query as required by the security model. So, $\mathcal{B}_2$ simulates Hyb_1 when $\mathcal{O}(\cdot) = \mathsf{PRF}_2(K^*_{w_0^*}, \cdot)$ (and $\mu = 0$) and Hyb_2 when $\mathcal{O}(\cdot) = f(\cdot)$ (and $\mu = 1$). Hence, a routine calculation (similar to (8)) gives

$$|\Pr[\mathsf{Hyb}_2 \Rightarrow 1] - \Pr[\mathsf{Hyb}_0 \Rightarrow 1]| = 2 \cdot \mathsf{Adv}_{\mathsf{PRF}_2,\mathcal{B}_2}(\lambda) \tag{9}$$

Hyb_3 differs from Hyb_2 in setting a random tag σ_1^*. This difference is essentially the same as the difference between Hyb_2 and Hyb_1. So, using the same strategy as above, we can construct an algorithm $\mathcal{B}_3$ such that

$$|\Pr[\mathsf{Hyb}_2 \Rightarrow 1] - \Pr[\mathsf{Hyb}_1 \Rightarrow 1]| = 2 \cdot \mathsf{Adv}_{\mathsf{PRF}_2,\mathcal{B}_3}(\lambda) \tag{10}$$

Since the outputs of $\mathsf{E}_K(w_b^*)$ is independent of b in Hyb_3, we get

$$|\Pr[\mathsf{Hyb}_3 \Rightarrow 1] \le 1/2 \tag{11}$$

Combining inequalities (7) to (11), we have

$$\mathsf{Adv}^{\mathsf{ct-ind}}_{\Sigma,\mathcal{A}}(\lambda) \le 2 \cdot (\mathsf{Adv}_{\mathsf{PRF}_1,\mathcal{B}_1}(\lambda) + \mathsf{Adv}_{\mathsf{PRF}_2,\mathcal{B}_2}(\lambda) + \mathsf{Adv}_{\mathsf{PRF}_2,\mathcal{B}_3}(\lambda))$$

where the right part is negligible under our hypotheses. □

Theorem 5. *The PRF-based EPE has weak query-key privacy if the PRFs are secure.*

Proof. Let $\mathcal{A}$ be the adversary. We define two hybrid security games Hyb_0, Hyb_1, each returns a well-defined binary value. The hybrid games are defined as follows:

Hyb_0**:** This game is identical to the experiment $\mathsf{Exp}^{\mathsf{qk-priv}}_{\Sigma,\mathcal{A}}(\lambda)$.

Hyb_1**:** Hyb_1 is identical to Hyb_0 except an empty set $\mathcal{L}$ is set and the oracle $\mathsf{K}_K(\cdot)$, $\mathsf{E}_K(\cdot)$ are described in Figure 7. Meanwhile, instead of running $\mathsf{KG}(K, w_b^*)$ to create the challenge, it proceeds by sampling $K_{w_b^*} \leftarrow U(\mathcal{K}')$, updating $\mathcal{L} \leftarrow \mathcal{L} \cup \{(w_b^*, K_{w_b^*})\}$, and return $K_{w_b^*}$.[10]

[10]Note, w_b^* is not queried $\mathsf{E}_K(\cdot)$ and $\mathsf{E}_K(\cdot)$ according to the security definition.

```
Oracle $\mathsf{K}_K(w)$: // $w \neq w_0^*, w_1^*$
1. Return $K_w$ if $(w, K_w) \in \mathcal{L}$
2. $K_w \leftarrow U(\mathcal{K}')$
3. $\mathcal{L} \leftarrow \mathcal{L} \cup \{(w, K_w)\}$
4. Return $\mathsf{dk}_w := K_w$
```

```
Oracle $\mathsf{E}_K(w)$: // $w \neq w_0^*, w_1^*$
1. $K_w \leftarrow \mathsf{K}_K(w)$
2. $r \leftarrow U(\mathcal{R})$
3. $t \leftarrow \mathsf{PRF}_2(K_w, r)$
4. Return $\sigma_w := (t, r)$
```

Figure 7: Oracles $\mathsf{K}_K(\cdot)$, $\mathsf{E}_K(\cdot)$ in Hyb_1, proof of Theorem 5

As Hyb_0 is the same as $\mathsf{Exp}^{\mathsf{qk-priv}}_{\Sigma,\mathcal{A}}(\lambda)$. By definition,

$$\Pr[\mathsf{Hyb}_0 \Rightarrow 1] = \Pr[\mathsf{Exp}^{\mathsf{ct-ind}}_{\Sigma,\mathcal{A}}(\lambda) = 1] \tag{12}$$

Hyb_1 differs from Hyb_0 by sampling a uniformly random value K_w for each keyword w (including $w = w_0^*, w_1^*$) in stead of computing using PRF_1. We construct an algorithm $\mathcal{B}$ which uses this difference to break PRF_1.

$\mathcal{B}$ has access to an oracle $\mathcal{O}$ and needs to decide whether $\mathcal{O}(\cdot) = \mathsf{PRF}_1(K, \cdot)$ or $O = f$ where $f : W \to \mathcal{K}'$ is a random function. To response to queries to $\mathsf{K}_K(w)$ (including the subroutine call to $\mathsf{E}_K(w)$), $\mathcal{B}$ makes a call to $\mathcal{O}(w)$ and receives K_w. For generating the challenge dk^* on w_b^*, $\mathcal{B}$ calls $\mathcal{O}(w_b^*)$ to get $K_{w_b^*}$ and returns $\mathsf{dk}^* := K_{w_b^*}$. Finally, if $\mathcal{A}$ returns $b' = b$, $\mathcal{B}$ outputs 0; otherwise, 1.

It is easy to see that $\mathcal{B}$ simulates Hyb_0 of $\mathcal{O}(\cdot) = \mathsf{PRF}_1(K, \cdot)$; else, $\mathcal{B}$ simulates Hyb_1. So, same to deriving inequality 13, we get

$$|\Pr[\mathsf{Hyb}_1 \Rightarrow 1] - \Pr[\mathsf{Hyb}_0 \Rightarrow 1]| = 2 \cdot \mathsf{Adv}_{\mathsf{PRF}_1,\mathcal{B}}(\lambda) \tag{13}$$

Finally, we note that since w_0^* and w_1^* cannot be used to query $\mathsf{E}_K(\cdot)$, hence, $K_{w_b^*}$ and b remain uniformly random under $\mathcal{A}$'s view. So, we conclude

$$|\Pr[\mathsf{Hyb}_1 \Rightarrow 1] \leq 1/2 \tag{14}$$

Combining inequalities (12) to (14), we have

$$\mathsf{Adv}^{\mathsf{qk-priv}}_{\Sigma,\mathcal{A}}(\lambda) \leq 2 \cdot \mathsf{Adv}_{\mathsf{PRF}_1,\mathcal{B}}(\lambda)$$

where the right part is negligible under our hypotheses. □

5.2 Construction from Bilinear Maps

We present our EPE scheme based on the DBDH (and the DDH) assumptions.

5.2.1 Bilinear Maps and Assumptions

Let $\mathbb{G}_0$, $\mathbb{G}_1$ and $\hat{\mathbb{G}}$ be three cyclic groups of order p where $\mathbb{G}_0 \neq \mathbb{G}_1$, $g_0 \in \mathbb{G}_0$ and $g_1 \in \mathbb{G}_1$ are generators. An efficiently computable function $\hat{e} : \mathbb{G}_0 \times \mathbb{G}_1 \to \hat{\mathbb{G}}$ is a (asymmetric) bilinear map or a pairing if the following properties are satisfied:

- $\hat{g} := \hat{e}(g, g)$ is the generator of $\hat{\mathbb{G}}$
- $\forall a, b \in \mathbb{Z}$, $\hat{e}(g_0^a, g_1^b) = \hat{e}(g_0, g_1)^{ab}$

Let λ be the security parameter, The DBDH assumption holds over $(\mathbb{G}_0, \mathbb{G}_1, \hat{\mathbb{G}}, \hat{e})$ if for all p.p.t algorithm $\mathcal{A}$, $T \leftarrow U(\hat{\mathbb{G}})$, $\alpha, \beta, \gamma \leftarrow U(\mathbb{Z}_p)$, the advantage $\mathsf{Adv}_{DBDH,\mathcal{A}}(\lambda)$ defined as

$$|\Pr[\mathcal{A}(g_0, g_0^\alpha, g_0^\beta, g_1, g_1^\alpha, g_1^\gamma, \hat{e}(g_0, g_1)^{\alpha\beta\gamma}) = 1] - \Pr[\mathcal{A}(g_0, g_0^\alpha, g_0^\beta, g_1, g_1^\alpha, g_1^\gamma, T) = 1]|$$

is negligible in λ. We say the DDH assumption holds over $\mathbb{G}_0$ if for all p.p.t algorithm $\mathcal{A}$, $T \leftarrow U(\mathbb{G}_0)$, $\alpha, c \leftarrow U(\mathbb{Z}_p)$ the advantage $\mathsf{Adv}_{DDH,\mathcal{A}}(\lambda)$ defined as

$$|\Pr[\mathcal{A}(g_0, g_0^\alpha, g_0^\beta, g_1^\alpha, g_0^{\alpha\beta}) = 1] - \Pr[\mathcal{A}(g_0, g_0^\alpha, g_0^\beta, g_1^\alpha, T) = 1]|$$

is negligible in λ. The above two assumptions hold simultaneously in Type-3 pairings.

5.2.2 The Construction

Let $z \leftarrow U(\mathbb{Z}_p)$, $g_0, u_0 = g_0^z, h_0 \in \mathbb{G}_0$, $g_1, v_1 = g_1^z \in \mathbb{G}_1$, $\hat{e} : \mathbb{G}_0 \times \mathbb{G}_1 \to \hat{\mathbb{G}}$ be an asymmetric bilinear map, and $\mathsf{PRF} : \mathcal{K} \times \mathcal{W} \to \mathbb{Z}_p$ be a PRF. $(g_0, u_0, h_0, g_1, v_1, \hat{e})$ are public parameters.

– $\mathsf{KG}(K, w)$: $r \leftarrow U(\mathbb{Z}_p)$, $t \leftarrow \mathsf{PRF}(K, w)$, $\mathsf{dk} := (k_1, k_2) = (h_0^t u_0^r, g_0^r)$.

– $\mathsf{Enc}(K, w)$: $s \leftarrow U(\mathbb{Z}_p)$, $t \leftarrow \mathsf{PRF}(K, w)$, $\sigma := (c_1, c_2, c_3) = (\hat{e}(h_0, g_1)^{ts}, v_1^s, g_1^s)$.

– $\mathsf{Qry}(\mathsf{dk}, \sigma)$: Returns 1 if $\hat{e}(c_3, k_1) = c_1 \cdot \hat{e}(k_2, c_2)$; Else, return 0.

For correctness, $\hat{e}(k_1, c_3) = \hat{e}(h_0^t u_0^r, g_1^s) = \hat{e}(h_0, g_1)^{st} \cdot \hat{e}(g_0, g_1)^{zsr} = c_1 \cdot \hat{e}(k_2, c_2)$, as required.

5.2.3 Security

The security of our construction of EPE from bilinear maps is stated in the following two theorems. We provide the proofs in Section A.1 and Section A.2.

Theorem 6. *Our construction based on bilinear maps has ciphertext indistinguishability, provided* PRF *is secure and the DBDH assumption holds.*

Theorem 7. *Our construction based on bilinear maps has query-key privacy, provided the DDH assumption in* $\mathbb{G}_0$ *holds.*

5.3 Constructions from (Ring) Lattices

This subsection describes an EPE from lattices with query-key privacy and ciphertext indistinguishability in the standard model. Hence, combining with the lattice-based NIKE scheme, e.g., [dK18, GdKQ+24], we obtain practical PAEKS schemes from lattices with FCI and FTI. Our construction is inspired by the PAEKS scheme from [CM22].

5.3.1 Lattice Preliminary

We use bold lower and upper case letters for vectors and matrices, e.g., $\mathbf{a}$, $\mathbf{A}$. Vectors are column vectors. We denote $\mathbf{a}^\intercal$ the transpose of $\mathbf{a}$. Let $\mathcal{R} = \mathbb{Z}[x]/(x^n + 1)$ where n is a power of two and q is a prime. The scheme works over the polynomial rings $\mathcal{R}$ and $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$. We note that the multiplication over $\mathcal{R}$ (and $\mathcal{R}_q$) is commutative. We denote by $D_{\mathcal{R},\tau}$ the discrete Gaussian distribution over $\mathcal{R}_q$ with Gaussian parameter τ (see [BEP+21]). For $\mathbf{a} \in \mathcal{R}$, $\|\mathbf{a}\|$ denotes the Euclidean norm of the coefficients vector of $\mathbf{a}$. Our construction uses lattice trapdoors. Let $k = \lceil \log_2 q \rfloor$ and $\bar{\mathbf{a}} \leftarrow U(\mathcal{R}_q^{m-k})$.

Lemma 1 ([GPV08, MP12, BEP+21])**.** *There exists a p.p.t algorithm* $\mathsf{TrapGen}(\bar{\mathbf{a}}; r)$ *that takes as input* $\bar{\mathbf{a}}$ *and randomness* $r \in \mathcal{S}$*, returns* $\mathbf{a}$ *and a trapdoor* $\mathbf{R}$ *where* $\mathbf{a}$ *and* $\mathbf{R}$ *are distributed statistically close to* $U(\mathcal{R}_q^m)$ *and* $D_{\mathcal{R}^{(m-k)\times k},\omega(\sqrt{\log_2 m})}$*, respectively.*

Lemma 2 ([GPV08, MP12])**.** *Let* $(\mathbf{a}, \mathbf{R})$ *be the output of* $\mathsf{TrapGen}$ *and* $u \in \mathcal{R}_q$*. There is a p.p.t algorithm* $\mathsf{SampleD}(\mathbf{a}, \mathbf{R}, u, \eta)$ *with* $u \in \mathcal{R}_q$, $\eta \geq \ell \cdot \omega(\sqrt{\log_2 m})$ *where* ℓ *is the operator norm of* $\mathbf{R}$ *(see [MP12]) that samples* $\mathbf{d}$ *according to a distribution that is statistically close to* $D_{\mathcal{R}^m,\eta}$*, conditioned on* $\mathbf{a}^\intercal \mathbf{d} = u$*.*

Lemma 3 (Lemma 7, [DM14])**.** *Let* $m \geq 2\log_2 q + 2$*. Let* $\mathbf{d} \leftarrow D_{\mathcal{R}^m,\eta}$*,* $\mathbf{a}^\intercal \mathbf{d}$ *is statistically close to* $U(\mathcal{R}_q)$*.*

Definition 8. Let λ be the security parameter. $\mathbf{a}, \mathbf{b} \leftarrow U(\mathcal{R}_q^m)$, $\mathbf{e} \leftarrow D_{\mathbb{Z}^m,\tau}$, and $s \leftarrow D_{\mathbb{Z},\tau}$, we say the LWE assumption holds if for all p.p.t adversary $\mathcal{A}$, the advantage

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{lwe}}(\lambda) = |\Pr[\mathcal{A}(\mathbf{a}, \mathbf{b}^\intercal) = 1] - \Pr[\mathcal{A}(\mathbf{a}, s\mathbf{a}^\intercal + \mathbf{e}^\intercal)]|$$

is negligible in λ.

5.3.2 The Construction

Our lattice-based construction has a key space $\mathcal{K}$, and a keywords space $\mathcal{W}$. It uses $\bar{\mathbf{a}} \leftarrow U(\mathcal{R}_q^{m-k})$, $u \leftarrow U(\mathcal{R}_q)$ as public parameters. It also uses a pseudorandom function $\mathsf{PRF} : \mathcal{K} \times \mathcal{W} \to \mathcal{S}$, i.e., PRF outputs the randomnesses used by $\mathsf{TrapGen}$. Let $\beta = O(\tau\eta m)$. The construction is given below.

– $\mathsf{KG}(K, w)$:

1. $r_w \leftarrow \mathsf{PRF}(K, w||0)$, $\tilde{r}_w \leftarrow \mathsf{PRF}(K, w||1)$
2. $(\mathbf{a}, \mathbf{R}_w) \leftarrow \mathsf{TrapGen}(\bar{\mathbf{a}}; r_w)$, $(\mathbf{b}, \tilde{\mathbf{R}}_w) \leftarrow \mathsf{TrapGen}(\bar{\mathbf{a}}; \tilde{r}_w)$
3. $\mathbf{e} \leftarrow D_{\mathcal{R}^m,\tau}$, $e \leftarrow D_{\mathcal{R},\tau}$, $s \leftarrow D_{\mathcal{R},\tau}$, $c \leftarrow su + e$, $\mathbf{c}^\intercal \leftarrow s\mathbf{b}^\intercal + \mathbf{e}^\intercal$
4. $\mathbf{d} \leftarrow \mathsf{SampleD}(\mathbf{a}, \mathbf{R}_w, c, \eta)$, $\mathsf{dk}_w := (\mathbf{d}, \mathbf{c})$

– $\mathsf{Enc}(K, w)$:

1. $r_w \leftarrow \mathsf{PRF}(K, w||0)$, $\tilde{r}_w \leftarrow \mathsf{PRF}(K, w||1)$
2. $(\mathbf{a}, \mathbf{R}_w) \leftarrow \mathsf{TrapGen}(\bar{\mathbf{a}}; r_w)$, $(\mathbf{b}, \tilde{\mathbf{R}}_w) \leftarrow \mathsf{TrapGen}(\bar{\mathbf{a}}; \tilde{r}_w)$
3. $\tilde{\mathbf{e}} \leftarrow D_{\mathcal{R}^m,\tau}$, $\tilde{e} \leftarrow D_{\mathcal{R},\tau}$, $\tilde{s} \leftarrow D_{\mathcal{R},\tau}$, $\tilde{c} \leftarrow \tilde{s}u + \tilde{e}$, $\tilde{\mathbf{c}}^\intercal \leftarrow \tilde{s}\mathbf{a}^\intercal + \tilde{\mathbf{e}}^\intercal$
4. $\tilde{\mathbf{d}} \leftarrow \mathsf{SampleD}(\mathbf{b}, \tilde{\mathbf{R}}_w, \tilde{c}, \eta)$, $\sigma_w := (\tilde{\mathbf{d}}, \tilde{\mathbf{c}})$

– $\mathsf{Qry}(\mathsf{dk}_w, \sigma_w)$:

1. Parse $\mathsf{dk}_w \to (\mathbf{d}, \mathbf{c})$, $\mathsf{ct}_w \to (\tilde{\mathbf{d}}, \tilde{\mathbf{c}})$, $\mathbf{v} \leftarrow \mathbf{c}^\intercal\tilde{\mathbf{d}} - (\tilde{\mathbf{c}})^\intercal\mathbf{d}$
2. Return 1 if $\|\mathbf{v}\| \leq \beta$; Otherwise, 0

For correctness, we have

$$\begin{aligned}\|\mathbf{v}\| &= \left\|\mathbf{c}^\intercal\tilde{\mathbf{d}} - (\tilde{\mathbf{c}})^\intercal\mathbf{d}\right\| = \left\|(s\mathbf{b}^\intercal + \mathbf{e}^\intercal)\tilde{\mathbf{d}} - (\tilde{s}\mathbf{a}^\intercal + \tilde{\mathbf{e}}^\intercal)\mathbf{d}\right\| \\ &\leq 2\tau\sqrt{m} \cdot \eta\sqrt{m} + 2\tau^2 n \leq O(\tau\eta m) \\ &= \beta\end{aligned}$$

Instantiations. We describe our lattice-based EPE using the polynomial ring $\mathcal{R}_q$ to make it slightly general and easy to read. The construction contains operations from the GPV identity-based encryption (GPV-IBE) scheme [GPV08], i.e., lattice preimage sampling and dual-Regev encryption. Therefore, our construction can be adapted to other lattices as long as they allow the implementations of GPV sampling and dual-Regev encryption. One such important example is NTRU lattices. Ducas et al. [DLP14] show how to build very efficient GPV-IBE from NTRU lattices.

5.3.3 Security

The security of our construction of EPE from lattices is stated in the following two theorems. We provide the proofs in Section B.1 and Section B.2.

Theorem 8. *Our construction from ring lattices has ciphertext indistinguishability if* PRF *is secure and the LWE assumption holds.*

Theorem 9. *Our construction from ring lattices has full query-key privacy, provided* PRF *is secure, and the LWE assumption holds.*

6 Extensions

Our generic construction offers several interesting extensions to expressive public-key authenticated encryption with keyword search.

Expressive PAEKS. We observe that EPE with query-key privacy is a special case of symmetric-key predicate encryption with predicate privacy or symmetric-key function-private functional encryption [SSW09, AAB$^+$13, BS18] which support richer classes of functions than the equality function. We can extend our generic construction to build expressive public-key authenticated searchable encryption by combining NIKE schemes with predicate-private predicate encryption to support expressive searching patterns, e.g., conjunctions/disjunctions or any polynomial-sized functions over keywords, hence enabling a wider range of applications.

Our construction can be extended to identity-based PAEKS by using identity-based NIKE [SOK00]. This will be beneficial to applications on small domains to ease certificate management. Finally, we can extend our construction to support token revocation, meaning that tokens can be revoked and disabled.

Token Revocation. To mitigate the situations where the tokens of the PAEKS system are compromised, or the key holders leave the domain, token revocation needs to be considered – the token is disabled from testing the ciphertext keywords once it is revoked. Similar to the identity key revocation from identity-based cryptography, discussed in [BF01], our generic construction of PAEKS can concatenate time information to the keywords, and new tokens are issued when the current token is expired. This way the ciphertext/token sizes are unchanged.

Notice that the tokens and ciphertexts in the PRF-based and DBDH-based constructions apply a PRF to the keywords. We can include the time slot t into the input, so the tokens with different time slots from that of the ciphertext will not be able to work. More specifically, let K be the shared key between the sender and the receiver. For the PRF-based construction, the token at time slot t is $\mathsf{tk}_{w,t} = \mathsf{PRF}_1(K, w||t)$, while the ciphertext at t' is computed as $\mathsf{ct}_{w,t'} := (\mathsf{PRF}_2(\mathsf{PRF}_1(K, w||t'), r), r)$. $\mathsf{tk}_{w,t}$ does not work on $\mathsf{ct}_{w,t'}$ if $t \neq t'$. For the DBDH-based construction, the token for w at time slot t is $\mathsf{tk}_{w,t} = (h^s \cdot u^r, g^r)$ where $s = \mathsf{PRF}(K, w||t)$ and the ciphertext for w at time slot t' is $\mathsf{ct}_{w,t'} = (\hat{e}(e, h)^{sr'}, u^s, g^s)$, where $s' = \mathsf{PRF}(K, w||t')$. Again, $\mathsf{tk}_{w,t}$ does not work on $\mathsf{ct}_{w,t'}$ if $t \neq t'$.

ID-Based Authenticated Encryption with Keywords Search For public-key cryptographic applications in small domains, the overhead of managing public-key certificates can be alleviated using identity-based cryptography [Sha85]. In an identity-based cryptosystem, user's public keys are their unique identity in the domain that they belong to, which can be any publicly known strings, e.g., company email addresses. A trusted entity, named the key generation center (KGC), generates private keys for users based on their identities. As users' public keys are unique and publicly known, they certify the users by default, and no public key certificates are needed.

Our generic constructions of PAEKS can be easily extended to identity-based settings. The only public-key primitive used by our constructions is NIKE. So, we can acquire an identity-based PAEKS construction by using identity-based non-interactive key exchange (ID-NIKE) while keeping the symmetric-key equality-predicate encryption unchanged. The security properties of identity-based PAEKS can be formalised straightforwardly by considering the ID-NIKE security models. Practical ID-NIKE schemes are known, e.g., [SOK00]. We also note that one can consider using an attribute-based NIKE to make a PAEKS attribute-based.

7 Conclusion

The cryptographic primitive public-key authenticated encryption with keyword search (PAEKS) recently emerged to deal with inside keyword guessing attacks (IKGA) against public-key encryption with keyword search (PEKS). PAEKS allows keyword-associated tokens to test if a ciphertext from a designated sender contains the keyword. PAEKS requires the tokens and ciphertexts not to leak information about the keywords embedded into them, known as token privacy and ciphertext indistinguishability. This paper presents a generic construction of PAEKS, featuring great simplicity and practicality. Instantiating our construction using efficient non-interactive key exchange and PRFs leads to the most efficient PAEKS schemes, with some having post-quantum security. We show how to instantiate our construction from pairings and lattices to get efficient PAEKS schemes with the strongest token privacy and ciphertext indistinguishability. To the best of our knowledge, only two PAEKS schemes have such strong token privacy yet substantially underperforming our ones under the same assumptions.

References

[AAB+13] Shashank Agrawal, Shweta Agrawal, Saikrishna Badrinarayanan, Abishek Kumarasubramanian, Manoj Prabhakaran, and Amit Sahai. Functional encryption and property preserving encryption: New definitions and positive results. Cryptology ePrint Archive, Paper 2013/744, 2013. URL: https://eprint.iacr.org/2013/744.

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (h)ibe in the standard model. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 553–572. Springer Berlin Heidelberg, 2010. doi:10.1007/978-3-642-13190-5_28.

[BBDQ18] Fabrice Benhamouda, Olivier Blazy, Léo Ducas, and Willy Quach. Hash proof systems over lattices revisited. In Michel Abdalla and Ricardo Dahab, editors, *Public-Key Cryptography – PKC 2018*, pages 644–674, Cham, 2018. Springer International Publishing. doi:10.1007/978-3-319-76581-5_22.

[BCOP04] Dan Boneh, Giovanni Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522. Springer Berlin Heidelberg, 2004. doi:10.1007/978-3-540-24676-3_30.

[BEP+21] Pauline Bert, Gautier Eberhart, Lucas Prabel, Adeline Roux-Langlois, and Mohamed Sabt. Implementation of lattice trapdoors on modules and applications. In *Post-Quantum Cryptography – PQCrypto 2021*, pages 195–214. Springer, 2021. doi:10.1007/978-3-030-81293-5_11.

[BF01] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In Joe Kilian, editor, *Advances in Cryptology–CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer Berlin Heidelberg, 2001. doi:10.1007/3-540-44647-8_13.

[BS18] Zvika Brakerski and Gil Segev. Function-private functional encryption in the private-key setting. *Journal of Cryptology*, 31:202–225, 2018. doi:10.1007/s00145-017-9261-0.

[BW06] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 290–307. Springer Berlin Heidelberg, 2006. doi:10.1007/11818175_17.

[CKS09] David Cash, Eike Kiltz, and Victor Shoup. The twin diffie–hellman problem and applications. *Journal of Cryptology*, 22:470–504, 2009. doi:10.1007/s00145-009-9041-6.

[CLM$^+$18] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. Csidh: an efficient post-quantum commutative group action. In *Advances in Cryptology–ASIACRYPT 2018*, pages 395–427. Springer, 2018. doi:10.1007/978-3-030-03332-3_15.

[CM22] Leixiao Cheng and Fei Meng. Public key authenticated encryption with keyword search from lwe. In *European Symposium on Research in Computer Security*, pages 303–324. Springer, 2022. doi:10.1007/978-3-031-17140-6_15.

[CQFM23] Leixiao Cheng, Jing Qin, Feng Feng, and Fei Meng. Security-enhanced public-key authenticated searchable encryption. *Information Sciences*, 647:119454, 2023. doi:10.1016/j.ins.2023.119454.

[DH76] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976. doi:10.1109/TIT.1976.1055638.

[dK18] Bor de Kock. *A non-interactive key exchange based on ring-learning with errors*. PhD thesis, Master's thesis. Master's thesis, Eindhoven University of Technology, 2018.

[DLP14] Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. Efficient identity-based encryption over ntru lattices. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 22–41. Springer, 2014. doi:10.1007/978-3-662-45608-8_2.

[DM14] Léo Ducas and Daniele Micciancio. Improved short lattice signatures in the standard model. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014*, pages 335–352, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. doi:10.1007/978-3-662-44371-2_19.

[Emu22] Keita Emura. Generic construction of public-key authenticated encryption with keyword search revisited: stronger security and efficient construction. In *Proceedings of the 9th ACM on ASIA Public-Key Cryptography Workshop*, pages 39–49, 2022. doi:10.1145/3494105.352623.

[FHKP13] Eduarda SV Freire, Dennis Hofheinz, Eike Kiltz, and Kenneth G Paterson. Non-interactive key exchange. In *Public-Key Cryptography–PKC 2013*, pages 254–271. Springer, 2013. doi:10.1007/978-3-642-36362-7_17.

[GdKQ$^+$24] Phillip Gajland, Bor de Kock, Miguel Quaresma, Giulio Malavolta, and Peter Schwabe. Swoosh: Efficient lattice-based non-interactive key exchange. In *USENIX Security Symposium – USENIX Security 2024*. USENIX Association, 2024. URL: https://www.usenix.org/system/files/sec24summer-prepub-883-gajland.pdf.

[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 '08, pages 197–206, New York, NY, USA, 2008. ACM. doi:10.1145/1374376.1374407.

[HHK18] Julia Hesse, Dennis Hofheinz, and Lisa Kohl. On tightly secure non-interactive key exchange. In *Annual International Cryptology Conference – CRYPTO 2018*, pages 65–94. Springer, 2018. doi:10.1007/978-3-319-96881-0_3.

[KY16] Shuichi Katsumata and Shota Yamada. Partitioning via non-linear polynomial functions: More compact ibes from ideal lattices and bilinear maps. In *Advances in Cryptology–ASIACRYPT 2016*, pages 682–712. Springer, 2016. doi:10.1007/978-3-662-53890-6_23.

[LHHS22] Hongbo Li, Qiong Huang, Jianye Huang, and Willy Susilo. Public-key authenticated encryption with keyword search supporting constant trapdoor generation and fast search. *IEEE Transactions on Information Forensics and Security*, 18:396–410, 2022. doi:10.1109/TIFS.2022.3224308.

[LTT$^+$22] Zi-Yuan Liu, Yi-Fan Tseng, Raylin Tso, Masahiro Mambo, and Yu-Chi Chen. Public-key authenticated encryption with keyword search: Cryptanalysis, enhanced security, and quantum-resistant instantiation. In *Proceedings of the 2022 ACM on Asia conference on computer and communications security*, pages 423–436, 2022. doi:10.1145/3488932.3497760.

[LW19] Zengpeng Li and Ding Wang. Achieving one-round password-based authenticated key exchange over lattices. *IEEE transactions on services computing*, 15(1):308–321, 2019. doi:10.1109/TSC.2019.2939836.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718. Springer Berlin Heidelberg, 2012. doi:10.1007/978-3-642-29011-4_41.

[NE19] Mahnaz Noroozi and Ziba Eslami. Public key authenticated encryption with keyword search: revisited. *IET Information Security*, 13(4):336–342, 2019. doi:10.1049/iet-ifs.2018.5315.

[QCH$^+$20] Baodong Qin, Yu Chen, Qiong Huang, Ximeng Liu, and Dong Zheng. Public-key authenticated encryption with keyword search revisited: Security model and constructions. *Information Sciences*, 516:515–528, 2020. doi:10.1016/j.ins.2019.12.063.

[QCZZ21] Baodong Qin, Hui Cui, Xiaokun Zheng, and Dong Zheng. Improved security model for public-key authenticated encryption with keyword search. In *Provable and Practical Security – ProvSec 2021*, pages 19–38. Springer, 2021. doi:10.1007/978-3-030-90402-9_2.

[Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In GeorgeRobert Blakley and David Chaum, editors, *Advances in Cryptology*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer Berlin Heidelberg, 1985. doi:10.1007/3-540-39568-7_5.

[SOK00] Ryuichi Sakai, Kiyoshi Ohgishi, and Masao Kasahara. Cryptosystems based on pairing. In *Symposium on Cryptography and Information Security*. Springer, 2000.

[SSW09] Emily Shen, Elaine Shi, and Brent Waters. Predicate privacy in encryption systems. In Omer Reingold, editor, *Theory of Cryptography*, pages 457–473, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. doi:10.1007/978-3-642-00457-5_27.

[XWC+24] Tao Xiang, Zhongming Wang, Biwen Chen, Xiaoguo Li, Peng Wang, and Fei Chen. Stopguess: A framework for public-key authenticated encryption with keyword search. *Computer Standards & Interfaces*, 88:103805, 2024. doi:10.1016/j.csi.2023.103805.

A Supplemental Proofs for Section 5.2

A.1 Security Proofs of Theorem 6

Proof. Let $\mathcal{A}$ be the adversary. We defined four hybrid games Hyb_i for $i = \{0, 1, 2, 3\}$. A well-defined binary value is returned at the end of each hybrid game. We denote $\mathsf{Hyb}_i \Rightarrow 1$ the event that Hyb_i outputs 1. We defined the hybrid games as follows:

Hyb_0: This is identical to $\mathsf{Exp}^{\mathsf{ct-ind}}_{\Sigma,\mathcal{A}}(\lambda)$.

Hyb_1: This is identical to Hyb_0 except that an empty set $\mathcal{L}$ is set and the oracles $\mathsf{E}_K(\cdot)$ and $\mathsf{K}_K(\cdot)$ are described in Figure 8. Meanwhile, the challenge σ^* on keyword w_b^* is also created using $\mathsf{E}_K(\cdot)$.[11]

Hyb_2: This is identical to Hyb_1 except for answering $\mathsf{E}_K(w_b^*)$ to create the challenge on query w_b^*. In stead of generating σ^* as in Hyb_1, Hyb_2 samples a random element $R \leftarrow U(\hat{\mathbb{G}})$, $s \leftarrow U(\mathbb{Z}_p)$, sets $(c_1, c_2, c_3) = (R, v_1^s, g_1^s)$, and returns $\sigma^* = (c_1, c_2, c_3)$. The same steps are followed for repeated queries on w_b^*.

Hyb_3: This is identical to Hyb_2 except for answering $\mathsf{E}_K(w_{1-b}^*)$. Hyb_2 samples a random element $R \leftarrow U(\hat{\mathbb{G}})$, $s \leftarrow U(\mathbb{Z}_p)$, sets $(c_1, c_2, c_3) = (R, v_1^s, g_1^s)$, and returns $\sigma = (c_1, c_2, c_3)$. The same steps are followed for repeated queries on w_{1-b}^*.

```
Oracle $\mathsf{K}_K(w)$: // $w \neq w_0^*, w_1^*$
1. Return $t$ if $(w, t) \in \mathcal{L}$
2. $t \leftarrow U(\mathbb{Z}_p)$, $\mathcal{L} \leftarrow \mathcal{L} \cup \{(w, t)\}$, $r \leftarrow U(\mathbb{Z}_p)$
3. $(k_1, k_2) = (h_0^t u_0^r, g_0^r)$
4. Return $\mathsf{dk}_w := (k_1, k_2)$
```

```
Oracle $\mathsf{E}_K(w)$:
1. Return $t$ if $(w, t) \in \mathcal{L}$
2. $t \leftarrow U(\mathbb{Z}_p)$, $\mathcal{L} \leftarrow \mathcal{L} \cup \{(w, t)\}$, $s \leftarrow U(\mathbb{Z}_p)$
3. $(c_1, c_2, c_3) = (\hat{e}(h_0, g_1)^{ts}, v_1^s, g_1^s)$
4. Return $\sigma_w := (c_1, c_2, c_3)$
```

Figure 8: Oracles $\mathsf{K}_K(\cdot)$, $\mathsf{E}_K(\cdot)$ in Hyb_1, proof of Theorem 6

By definition, we have

$$\Pr[\mathsf{Hyb}_0 \Rightarrow 1] = \Pr[\mathsf{Exp}^{\mathsf{ct-ind}}_{\Sigma,\mathcal{A}}(\lambda) = 1] \tag{15}$$

Since the only difference between Hyb_0 and Hyb_1 is that in Hyb_0, $t = \mathsf{PRF}(K, w)$ for all w in the queries $\mathsf{E}_K(w)$, $\mathsf{K}_K(w)$. Using the same reduction as we give in the proof of inequality 8, we can differentiate PRF from a random function and get

$$|\Pr[\mathsf{Hyb}_1 \Rightarrow 1] - \Pr[\mathsf{Hyb}_0 \Rightarrow 1]| \leq 2\mathsf{Adv}_{\mathsf{PRF},\mathcal{B}_1}(\lambda) \tag{16}$$

for some algorithm $\mathcal{B}_1$.

We show that Hyb_1 and Hyb_2 can not be efficiently distinguished if the DBDH assumption holds. We construct an efficient algorithm $\mathcal{B}_2$ to break the DBDH assumption using the difference between Hyb_1 and Hyb_2 shown in Figure 9.

We note that the parameters $(g_0, h_0, u_0, g_1, v_1)$ simulated $\mathcal{B}_2$ are randomly distributed over $\mathbb{Z}_p$ as required. For $w \neq w_b^*$, $\mathsf{K}_K(w)$ and $\mathsf{E}_K(w)$ simulated by $\mathcal{B}_2$ are distributed as

[11] Note, there are no restrictions from the security definition on the input to $\mathsf{E}_K(\cdot)$.

```
Algorithm $\mathcal{B}_2$:
Input: $(g_0, g_0^\alpha, g_0^\beta, g_1, g_1^\alpha, g_1^\gamma, T)$
1. $z \leftarrow U(\mathbb{Z}_p)$, $h = g^\beta$, $u_0 = g_0^z$, $v_1 = g_1^z$
2. $(w_0^*, w_1^*) \leftarrow \mathcal{A}^{\mathsf{K}_K(\cdot), \mathsf{E}_K(\cdot)}$, $b \leftarrow U(\{0,1\})$
3. $\sigma^* \leftarrow \tilde{\mathsf{E}}(w_b^*)$, $b' \leftarrow \mathcal{A}^{\mathsf{K}_K(\cdot), \mathsf{E}_K(\cdot)}(\sigma^*)$
4. Return $(b' == b)$

Oracle $\mathsf{K}_K(w)$: // $w \neq w_0^*, w_1^*$
1. Return $t$ if $(w,t) \in \mathcal{L}$
2. $t \leftarrow U(\mathbb{Z}_p)$, $\mathcal{L} \leftarrow \mathcal{L} \cup \{(w,t)\}$, $r \leftarrow U(\mathbb{Z}_p)$
3. $(k_1, k_2) = (h_0^t u_0^r, g_0^r)$
4. Return $\mathsf{dk}_w := (k_1, k_2)$

Oracle $\mathsf{E}_K(w)$:
1. Return $\tilde{\mathsf{E}}(w)$ if $w = w_b^*$
2. Return $t$ if $(w,t) \in \mathcal{L}$
3. $t \leftarrow U(\mathbb{Z}_p)$, $\mathcal{L} \leftarrow \mathcal{L} \cup \{(w,t)\}$, $s \leftarrow U(\mathbb{Z}_p)$
4. $(c_1, c_2, c_3) = (\hat{e}(h_0, g_1)^{ts}, v_1^s, g_1^s)$
5. Return $\sigma_w := (c_1, c_2, c_3)$

Oracle $\tilde{\mathsf{E}}(w_b^*)$:
1. $\tilde{s} \leftarrow U(\mathbb{Z}_p)$
2. $c_1 = T^{\tilde{s}}$, $c_2 = (g_1^\alpha)^{z\tilde{s}}$, $c_3 = (g_1^\alpha)^{\tilde{s}}$
3. Return $\sigma^* := (c_1, c_2, c_3)$
```

Figure 9: Algorithm $\mathcal{B}_2$ breaks DBDH, proof of Theorem 6

in Hyb_1 (and Hyb_2). Note that challenge σ^* is produced by $\tilde{\mathsf{E}}(w_b^*)$, and $\mathsf{E}_K(w_b^*) = \tilde{\mathsf{E}}(w_b^*)$. $\mathcal{B}$ implicitly sets the keyword w_b^*'s corresponding exponent $t = \gamma$ and the encryption randomness $s = \alpha\tilde{s}$. When $T = \hat{e}(g_0, g_1)^{\alpha\beta\gamma}$, we have $c_1 = T^{\tilde{s}} = \hat{e}(g_0, g_1)^{\alpha\beta\gamma\tilde{s}} = \hat{e}(h_0, g_1)^{ts}$, $c_2 = (g_1^\alpha)^{z\tilde{s}} = v_1^s$, and $c_3 = (g_1^\alpha)^{\tilde{s}} = g_1^s$, which is a challenge ciphertext distributed as in Hyb_1. Otherwise, if T is random, σ^* is random and independent of w_b^*, i.e., a distribution required by Hyb_2. We note that we implicitly set the encryption randomness $s = c\tilde{s}$ where $\tilde{s}$ is freshly chosen for each repeated challenge query. The ciphertexts generated by $\tilde{\mathsf{E}}(w_b^*)$ when T is random are random and independent of each other. Hence, we have

$$|\Pr[\mathsf{Hyb}_2 \Rightarrow 1] - \Pr[\mathsf{Hyb}_1 \Rightarrow 1]| \leq 2\mathsf{Adv}_{DBDH,\mathcal{B}_2}(\lambda) \tag{17}$$

for the algorithm $\mathcal{B}_2$.

Hyb_3 generates fresh random ciphertext for repeated $\mathsf{E}_K(w_{1-b}^*)$, the same as the way that Hyb_2 deal with the queries $\mathsf{E}_K(w_b^*)$, using the same argument, we have

$$|\Pr[\mathsf{Hyb}_3 \Rightarrow 1] - \Pr[\mathsf{Hyb}_2 \Rightarrow 1]| \leq 2\mathsf{Adv}_{DBDH,\mathcal{B}_2}(\lambda) \tag{18}$$

for the algorithm $\mathcal{B}_2$. Finally, we note that the response to $\mathsf{E}_K(w_b^*)$ and $\mathsf{E}_K(w_{1-b}^*)$ are independent of b. Thus, $\mathcal{A}$ has no advantage in winning the game, hence

$$|\Pr[\mathsf{Hyb}_3 \Rightarrow 1] = 1/2 \tag{19}$$

Combining the inequalities (15) to (19) and the hypotheses PRF is secure and DBDH assumption holds. □

A.2 Security Proofs of Theorem 7

Proof. We proceed with the proof via three hybrid games.

Hyb_0: This is identical to $\mathsf{Exp}_{\Sigma,\mathcal{A}}^{\mathsf{qk-ind}}(\lambda)$.

Hyb_1: This is identical to Hyb_0 except that changes the way of constructing the challenge dk^* and the oracle $\mathsf{K}_K(\cdot)$. In particular, to construct the challenge on w_b^* and (repeated) queries $\mathsf{E}_K(w_b*)$, it choose a random $R_1, R_2 \leftarrow U(\mathbb{G})$ and returns (R_1, R_2).

Hyb_2: This is identical to Hyb_1 except that for all (repeated) queries on getting the query keys for w_{1-b}^*, it chooses a random $R_1, R_2 \leftarrow U(\mathbb{G})$ and returns (R_1, R_2)

By definition, we have

$$\Pr[\mathsf{Hyb}_0 \Rightarrow 1] = \Pr[\mathsf{Exp}_{\Sigma,\mathcal{A}}^{\mathsf{ct-ind}}(\lambda) = 1] \tag{20}$$

```
Algorithm $\mathcal{B}_1$:
Input: $(g_0, g_0^\alpha, g_0^\beta, g_1, g_1^\alpha, T)$
1. $u_0 = g_0^\alpha$, $h_0 \leftarrow U(\mathbb{G}_0)$, $v_1 = g_1^\alpha$
2. $K \leftarrow U(\mathcal{K})$, $(w_0^*, w_1^*) \leftarrow \mathcal{A}^{\mathsf{K}_K(\cdot),\mathsf{E}_K(\cdot)}$
3. $b \leftarrow U(\{0,1\})$
4. $\sigma^* \leftarrow \tilde{\mathsf{K}}(w_b^*)$, $b' \leftarrow \mathcal{A}^{\mathsf{K}_K(\cdot),\mathsf{E}_K(\cdot)}(\sigma^*)$
5. Return $(b' == b)$

Oracle $\mathsf{K}_K(w)$:
1. If $w = w_b^*$, return $\tilde{\mathsf{K}}(w_b^*)$
2. $t \leftarrow \mathsf{PRF}(K, w)$, $r \leftarrow U(\mathbb{Z}_p)$
3. $(k_1, k_2) = (h_0^t u_0^r, g_0^r)$
4. Return $\mathsf{dk}_w := (k_1, k_2)$
```

```
Oracle $\mathsf{E}_K(w)$:
1. Return $\tilde{\mathsf{K}}(w)$ if $w = w_b^*$
2. $t \leftarrow \mathsf{PRF}(K, w)$, $s \leftarrow U(\mathbb{Z}_p)$
3. $(c_1, c_2, c_3) = (\hat{e}(h_0, g_1)^{ts}, v_1^s, g_1^s)$
4. Return $\sigma_w := (c_1, c_2, c_3)$

Oracle $\tilde{\mathsf{K}}(w_b^*)$:
1. $t \leftarrow \mathsf{PRF}(K, w_b^*)$
2. $\tilde{r} \leftarrow U(\mathbb{Z}_p)$
3. $(k_1, k_2) = (h_0^t T^{\tilde{r}}, (g_0^\beta)^{\tilde{r}})$
4. Return $\mathsf{dk} := (k_1, k_2)$
```

Figure 10: Algorithm $\mathcal{B}_1$ breaks DDH, proof of Theorem 7

We show that Hyb_0 and Hyb_1 are indistinguishable if the DDH assumption holds. We constructed an efficient DDH problem solver $\mathcal{B}_1$ in Figure 10 who is given $(g_0, g_0^\alpha, g_0^\beta, g_1, g_1^\alpha, T)$ and decides if $T \in \mathbb{G}_0$ is random or equals $g_0^{\alpha\beta}$.

We can see that the parameters $(g_0, u_0, h_0, g_1, v_1)$, oracles $\mathsf{K}_K(\cdot)$ and $\mathsf{E}_K(\cdot)$ on queries $w \neq w_b^*$ simulated by $\mathcal{B}_1$ are distributed correctly as in Hyb_0 (and in Hyb_1). When $T = g^{\alpha\beta}$, we have $k_1 = h_0^t T^{\tilde{r}} = h_0^t (g_0^{\alpha\beta\tilde{r}}) = h_0^t u_0^r$, and $k_2 = (g_0^\beta)^{\tilde{r}} = g_0^r$, i.e., a challenge query key distributed as in Hyb_0. We note, for repeated query on w_b^*, the (unknown) randomness $r = \beta\tilde{r}$ is uniformly random over $\mathbb{Z}_p$ where $\tilde{r}$ is freshly chosen. On the other hand, if T is random, the responses are random and independent of each other, which is distributed as required in Hyb_1. Hence,

$$|\Pr[\mathsf{Hyb}_1 \Rightarrow 1] - \Pr[\mathsf{Hyb}_0 \Rightarrow 1]| \leq 2\mathsf{Adv}_{DDH,\mathcal{B}_1}(\lambda) \tag{21}$$

for $\mathcal{B}_1$. Using the same argument to Hyb_1 and Hyb_2, we obtain

$$|\Pr[\mathsf{Hyb}_2 \Rightarrow 1] - \Pr[\mathsf{Hyb}_1 \Rightarrow 1]| \leq 2\mathsf{Adv}_{DDH,\mathcal{B}_1}(\lambda) \tag{22}$$

for the algorithm $\mathcal{B}_2$.

Finally, we can see that all the query-key responses are independent of w_b^* and w_{1-b}^*, hence $\mathcal{A}$ no advantage, i.e.,

$$|\Pr[\mathsf{Hyb}_3 \Rightarrow 1] = 1/2 \tag{23}$$

Combining the inequalities (20) to (23) and the hypotheses PRF is secure and the DBDH assumption holds. □

B Supplemental Proofs for Section 5.3

B.1 Security Proofs for Theorem 8

We give the security proof below. We note that the polynomially many hybrid games make the proof non-tight. We can tighten the reduction using LWE sample generation [GPV08] and noise re-randomisation [KY16], albeit relying on a stronger LWE assumption.

Proof. We assume that the adversary makes Q and Q' queries to $\mathsf{E}_K(\cdot)$ on the challenge keyword w_b^* and w_{1-b}^*, respectively, after receiving the challenge ciphertext σ^*. We proceed with the proof via hybrid games Hyb_0, Hyb_1, $\mathsf{Hyb}_{2,i}$ for $i = \{1, 2, , ..., Q\}$ and $\mathsf{Hyb}_{3,i}$ for $i = \{1, 2, , ..., Q'\}$. A binary value is returned at the end of each game. We denote by $\mathsf{Hyb}_x \Rightarrow 1$ the event that Hyb_x returns 1.

Hyb_0**:** This is identical to $\mathsf{Exp}^{\mathsf{ct-ind}}_{\Sigma,\mathcal{A}}(\lambda)$.

Hyb_1**:** This is identical to Hyb_0 except that a PRF is not used to generate the randomness for $\mathsf{TrapGen}$: At the beginning of Hyb_1, an empty list $\mathcal{L}$ is initialised. Given query w to the oracles K and E, Hyb_1 first check if $(w, r_w, \tilde{r}_w) \in \mathcal{L}$, if yes, returns $(w, r_w, \tilde{r}_w)$; otherwise the game draws $r, \tilde{r} \leftarrow U(\mathcal{S})$, and updates $\mathcal{L} \leftarrow \mathcal{L} \cup \{(w, r_w, \tilde{r}_w)\}$.

Hyb_2**:** This is identical to Hyb_1 except that for all $\mathsf{K}_K(\cdot)$ and $\mathsf{E}_K(\cdot)$ on the challenge keyword w_b^*, $\mathbf{a} \leftarrow U(\mathcal{R}_q^m)$.

Hyb_3**:** This is identical to Hyb_2 except that that for the challenge query on the keyword w_b^* query, the ciphertext components $\tilde{c}$ and $\tilde{c}$ are chosen uniformly random over $\mathcal{R}_q$ and $\mathcal{R}_q^m$, respectively.

$\mathsf{Hyb}_{3,i}$**:** This is identical to $\mathsf{Hyb}_{3,i-1}$ except that for the ith $\mathsf{E}_K(w_b^*)$ query, the ciphertext components $\tilde{c}$ and $\tilde{c}$ are chosen uniformly random over $\mathcal{R}_q^m$ and $\mathcal{R}_q$, respectively. We define $\mathsf{Hyb}_{3,0} = \mathsf{Hyb}_3$.

Hyb_4**:** This is identical to $\mathsf{Hyb}_{3,Q}$ except that for all $\tilde{\mathsf{E}}_K(w_b^*)$ queries, $\tilde{\mathbf{d}} \leftarrow D_{\mathcal{R}^m,\eta}$ and $\tilde{c} \leftarrow \mathbf{b}^\intercal \tilde{\mathbf{d}}$.

$\mathsf{Hyb}_{4,i}$**:** This is identical to $\mathsf{Hyb}_{3,i-1}$ except that for the ith $\mathsf{E}_K(w_{1-b}^*)$ query, the ciphertext components $\tilde{c}$ and $\tilde{c}$ are chosen uniformly random over $\mathcal{R}_q^m$ and $\mathcal{R}_q$, respectively. We define $\mathsf{Hyb}_{4,0} = \mathsf{Hyb}_4$.

Hyb_5**:** This is identical to $\mathsf{Hyb}_{4,Q'}$ except that for all $\mathsf{E}_K(w_{1-b}^*)$ queries, $\tilde{\mathbf{d}} \leftarrow D_{\mathcal{R}^m,\eta}$ and $\tilde{c} \leftarrow \mathbf{b}^\intercal \tilde{\mathbf{d}}$.

By definition, we have

$$\Pr[\mathsf{Hyb}_0 \Rightarrow 1] = \Pr[\mathsf{Exp}^{\mathsf{ct-ind}}_{\Sigma,\mathcal{A}}(\lambda) = 1] \tag{24}$$

Hyb_2 generates true randomness for $\mathsf{TrapGen}$ instead of using PRF. It also uses the list $\mathcal{L}$ to maintain consistency, i.e., the same keyword gives the same vectors $\mathbf{a}, \mathbf{b}$. A routine reduction shows

$$|\Pr[\mathsf{Hyb}_1 \Rightarrow 1] - \Pr[\mathsf{Hyb}_0 \Rightarrow 1]| \leq 2\mathsf{Adv}_{\mathsf{PRF},\mathcal{B}_1}(\lambda) \tag{25}$$

for some algorithm $\mathcal{B}_1$.

Hyb_2 samples $\mathbf{a} \leftarrow U(\mathcal{R}_q^m)$ for the challenge keyword w_b^*. Since w_b^* cannot used to query $\mathsf{K}_K()$, a trapdoor for $\mathbf{a}$ is no longer needed. Using Lemma 3, we have Hyb_2 and Hyb_1 are statistically indistinguishable. Hence,

$$|\Pr[\mathsf{Hyb}_2 \Rightarrow 1] - \Pr[\mathsf{Hyb}_1 \Rightarrow 1]| \leq \mathsf{negl}(\lambda) \tag{26}$$

Hyb_3 differs from Hyb_2 in choosing random ciphertext component $\tilde{c}$ and $\tilde{\mathbf{c}}$. Since $\mathbf{a}$ and u are random, $\tilde{c}$ and $\tilde{\mathbf{c}}$ are LWE samples with the secret s. Hence, a straightforward reduction builds an algorithm $\mathcal{B}_1$ such that

$$|\Pr[\mathsf{Hyb}_3 \Rightarrow 1] - \Pr[\mathsf{Hyb}_2 \Rightarrow 1]| \leq 2\mathsf{Adv}^{\mathsf{lwe}}_{\mathcal{B}_1}(\lambda) \tag{27}$$

for some algorithm $\mathcal{B}_1$.

The difference between $\mathsf{Hyb}_{3,i}$ and $\mathsf{Hyb}_{3,i-1}$ is similar to the difference between Hyb_3 and Hyb_2. Using the same algorithm $\mathcal{B}_1$ we have

$$|\Pr[\mathsf{Hyb}_{3,i} \Rightarrow 1] - \Pr[\mathsf{Hyb}_{3,i-1} \Rightarrow 1]| \leq 2\mathsf{Adv}^{\mathsf{lwe}}_{\mathcal{B}_1}(\lambda) \tag{28}$$

In Hyb_4, $\mathbf{b}$ is statistically close to $\mathcal{R}_q^m$ according to Lemma 3. It produce the joint distribution $(\tilde{\mathbf{d}}, \tilde{c})$ by sampling $\tilde{\mathbf{d}} \leftarrow D_{\mathcal{R}^m,\eta}$ and setting $\tilde{c} \leftarrow \mathbf{b}^\intercal \tilde{\mathbf{d}}$. So, using Lemma 3 again, $\tilde{\mathbf{c}}$ is statistically close to $U(\mathcal{R}_q)$. In Hyb_3, $\tilde{c} \leftarrow U(\mathcal{R}_q)$ and $\tilde{\mathbf{d}}$ is obtained via $\mathsf{SampleD}(\mathbf{b}, \tilde{\mathbf{R}}_{w_b^*}, \tilde{c}, \eta)$. According to Lemma 2, $\tilde{\mathbf{d}}$ in the two hybrid games are distributed statistically close. As a result,

$$|\Pr[\mathsf{Hyb}_4 \Rightarrow 1] - \Pr[\mathsf{Hyb}_{3,Q} \Rightarrow 1]| \leq \mathsf{negl}(\lambda) \tag{29}$$

The difference between $\mathsf{Hyb}_{4,i}$ and $\mathsf{Hyb}_{4,i-1}$ is similar to the difference between $\mathsf{Hyb}_{3,i}$ and $\mathsf{Hyb}_{3,i-1}$. So, we obtain

$$|\Pr[\mathsf{Hyb}_{4,i} \Rightarrow 1] - \Pr[\mathsf{Hyb}_{4,i-1} \Rightarrow 1]| \leq 2\mathsf{Adv}_{\mathcal{B}_1}^{\mathsf{lwe}}(\lambda) \tag{30}$$

The difference between $\mathsf{Hyb}_{4,Q'}$ and Hyb_5 is essentially the same as the difference between $\mathsf{Hyb}_{3,Q}$ and Hyb_4. Hence, using the same argument, we have

$$|\Pr[\mathsf{Hyb}_{4,Q'} \Rightarrow 1] - \Pr[\mathsf{Hyb}_5 \Rightarrow 1]| \leq \mathsf{negl}(\lambda) \tag{31}$$

We note that both the challenge ciphertext and the ciphertexts replied to the encryption query on w_b^* and w_{1-b}^* are randomly chosen and independent of w_b^* and w_{1-b}^*. Hence, we have

$$\Pr[\mathsf{Hyb}_5 \Rightarrow 1] = 1/2 \tag{32}$$

Putting 24 to 32 together and using the fact that Q, Q' are polynomials and the hypothesis that LWE problem is hard, we get $\mathsf{Adv}_{\Sigma,\mathcal{A}}^{\mathsf{ct-ind}}(\lambda)$ is negligible in λ. ☐

B.2 Security Proof of Theorem 9

Proof. Note that the ciphertext and key are constructed essentially in the same way. The ciphertext indistinguishability and the full query-key privacy are defined symmetrically. Thus, a proof similar to the proof of Theorem 8 applies. We omit the details. ☐

C Comments on Existing PAEKS Schemes

C.1 Comment On Liu at al.'s Scheme

Liu et al. [LTT+22] give a generic construction of PAEKS with an instantiation from the LWE problem. There is an inconsistency in the concrete parameters provided. Let n, m, and q be the lattice dimension, the number of LWE samples, and the modulus. The parameters given are $n = 256$, $m = 9753$, and $q = 4096$ plus the discreet Gaussian parameter $\sigma = m \cdot \ell \cdot \omega(\sqrt{\log_2 n})$ with $\ell = 10$, $\omega(\sqrt{\log_2 n}) > 1$. The concrete value of σ provided is 8, which is less than m.

C.2 Comment On Cheng et al.'s Scheme

Cheng et al. propose a PAEKS scheme from pairings under the oracle Diffie-Hellam (DH) assumption [CQFM23]. We comment that the scheme needs the DBDH assumption, which is stronger than the oracle DH assumption.

The scheme use a pairing $\hat{e} : \mathbb{G} \times \mathbb{G} \to \hat{\mathbb{G}}$ where $\mathbb{G}$ and $\hat{\mathbb{G}}$ are cyclic group of prime order p. $g, h \in \mathbb{G}$ and hash functions H, $\hat{H}$ are public parameters. Let S and R be the sender and receiver, respectively. The public/private key pair of S is $(\mathsf{pk}_S, \mathsf{sk}_S) = (g^y, y)$, and the public/private key pair of R is $(\mathsf{pk}_R, \mathsf{sk}_R) = (g^x, x)$. The token is (T_1, T_2, T_3, T_4) where

$$T_1 = H(\hat{H}(K), w, \mathsf{pk}_R, \mathsf{pk}_S)^{s_1} h^{ys_2},\ T_2 = g^{ys_1},\ T_3 = h^{s_2},\ T_4 = g^{s_1}$$

where $K = g^{xy}$, and s_1, $s_2 \leftarrow U(Z_p)$. Given a token, one can compute

$$\begin{aligned}\frac{\hat{e}(T_1, g)}{\hat{e}(T_3, \mathsf{pk}_S)} &= \frac{\hat{e}(H(\hat{H}(K), w, \mathsf{pk}_R, \mathsf{pk}_S)^{s_1} h^{ys_2}, g)}{\hat{e}(h^{s_2}, g^y)} \\ &= \hat{e}(H(\hat{H}(K), w, \mathsf{pk}_R, \mathsf{pk}_S), h^{s_2}) \\ &= \hat{e}(c_{S,R,w}, h^{s_2})\end{aligned}$$

where $c_{S,R,w} \in \mathbb{G}$ is a fixed constant when S, R, and the keyword w is given. In the FTI model, the attacker can ask for multiple, e.g., Q tokens from S, R on w to obtain multiple such values $\{\hat{e}(c_{S,R,w}, h^{s_{2,i}})\}_{i\in\{1,2,\ldots,Q\}}$. To argue the scheme is of FTI, one needs to show the joint distribution of $\{\hat{e}(c_{S,R,w}, h^{s_{2,i}})\}_{i\in\{1,2,\ldots,Q}$ is computationally indistinguishable from the uniform distribution over the Cartesian product of Q groups G, for which the oracle DH assumption is insufficient but the DBDH assumption is.