IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Generic Construction of Dual-Server Public Key Encryption with Keyword Search on Cloud Computing

RAYLIN TSO[1], KAIBIN HUANG[1], YU-CHI CHEN[2], SK MD MIZANUR RAHMAN[3] AND TSU-YANG WU[4,5]
[1]Department of Computer Science, National Chengchi University, Taipei, Taiwan (e-mail: raylin@cs.nccu.edu.tw, ukstoryle@gmail.com)
[2]Department of Computer Science and Engineering, Yuan Ze University, Taoyuan, Taiwan (e-mail: wycchen@saturn.yzu.edu.tw)
[3]School of Engineering Technology and Applied Science (SETAS), Centennial College, Toronto, Canada (e-mail: Sheikh.Mizanur@gmail.com)
[4]College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China
[5]Fujian Provincial Key Laboratory of Big Data Mining and Applications, Fujian University of Technology, Fuzhou 350118, China

Corresponding author: Tsu-Yang Wu (e-mail: wutsuyang@gmail.com).

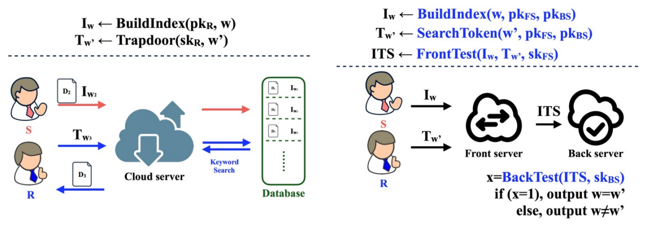
The work of R. Tso were supported by the Ministry of Science and Technology, Taiwan, under the grant MOST 108-2218-E-004-001, 109-2221-E-004-011-MY3 and Tsu-Yang Wu was supported in part by the Natural Science Foundation of Fujian Province, China under Grant no. 2018J01636 and 2018J01638.

ABSTRACT Chen et al. indicated the inner keyword guessing attack coming from the low entropy of keywords, which eliminates the semantic security of most keyword search schemes. Then, they accordingly propose the first dual-server PEKS scheme (abbreviated as DS-PEKS) and its related security models to prevent such attacks. In the DS-PEKS architecture, two non-collusive servers must corporate to execute the keyword search procedure. No individual server has the capability of determining the equivalence between keywords alone, and thus the inner keyword guessing attacks can be avoided. In this paper, we found that the security models are lack of soundness and strength, so our first result is to define new stronger and sounder security models which implies all security aspects of original models. Secondly, we also propose a generic construction of DS-PEKS schemes based on IND-CCA2 secure PKE schemes. Finally, we analyze the newly proposed DS-PEKS scheme, and proof its security with the stronger models based on the IND-CCA2 security in the standard model.

I. INTRODUCTION

NOWADAYS, the cloud services are ubiquitous in real-life.For example, people would like to outsource their photos to Google Drive for reducing cell-phone storage usage. If such data are confidential such as phone numbers, birthdays, credit card numbers or even medical records, it leads a serious security issue for secrecy and privacy. In fact, some financial service providers and on-line banks store extremely sensitive data like bank account numbers and credit card numbers from their consumers. In addition to the issue of authenticity which can be done by authentication schemes [1, 2], the security issue of the outsourced data has also been highly attracted attention along with the spread of cloud applications. A natural solution is directly to encrypt such confidential data before storing but we are not able to do any computation over these encrypted data. For general purposes, fully homomorphic encryption (FHE) firstly introduced by Gentry [3] is a very powerful tool to achieve the goal, but as we knew the computation cost of FHE is quite expansive. For specific purposes (i.e., searching, indexing, clustering, optimizing), we should have some more efficient solutions, and thus cryptosystems with extra specific functionalities [4] have been regarded as an urgent requirement in the modern technology.

Searchable encryption is a cryptographic notion that allows servers to search over encrypted documents without losing their privacy. The first public key encryption with keyword search (abbreviated as PEKS) was proposed by Boneh et al. [5] in 2004. It works in the following scenario (illustrated in Fig. 1(a)): a sender encrypts a document and its searchable keyword using a receiver's public key. Then, the sender stores the document as well as the index (encrypted keyword) on a cloud data server. Once the receiver wants to retrieve her own documents (related to some keywords), she generates a trapdoor using her private key and the desired keyword. On receiving the trapdoor from the receiver, the cloud storage server executes tests between the input trapdoor and all index stored on the server. The corresponding documents will be returned when the keyword hidden in the index is verified equal to the keyword in the trapdoor. For simplicity, we usually focus on the ‘keyword search’ part, and omit the security of the encryption and decryption of documents. It is assumed that the encryption of documents is secure.


$I_w \leftarrow$ BuildIndex(pk_R, w)
$T_{w'} \leftarrow$ Trapdoor(sk_R, w')
D_2 I_{w_2}
S
T_{w_3}
R
D_3
Cloud server
Keyword Search
D_1 I_{w_1}
D_2 I_{w_2}
D_3 I_{w_3}
Database
$I_w \leftarrow$ BuildIndex(w, pk_{FS}, pk_{BS})
$T_{w'} \leftarrow$ SearchToken(w', pk_{FS}, pk_{BS})
ITS $\leftarrow$ FrontTest(I_w, $T_{w'}$, sk_{FS})
I_w
$T_{w'}$
ITS
Front server
Back server
x=BackTest(ITS, sk_{BS})
if (x=1), output w=w'
else, output w≠w'

(a) Syntax of PEKS. (b) Syntax of DS-PEKS.

FIGURE 1: Syntax of PEKS and dual-server PEKS.

PEKS immediately attracted lots of significant interests. Based on the notion of [5], numerous follow-up works have been proposed to achieve different requirements of keyword search. Boneh and Waters [6] proposed a PEKS construction about multi-keyword search, which is able to support conjunctive keyword search, subset keyword search and range queries. The conjunctive keyword search is widely discussed in [7, 8]. However, on considering the security of trapdoors [9], most PEKS schemes rely on a secure channel to secretly transfer the trapdoor. Baek et al. proposed a secure-channel free PEKS [10] solution to deal with this issue. Some PEKS surveys [10–12] aim to compare existing PEKS schemes and show some limitations in this area.

An inherent limitation is the low-entropy of keywords. Byun et al. [13] introduced the off-line keyword guessing attacks which can be referred to as a kind of brute force attacks but is somehow realistic. More precisely, there is an adversary who is able to obtain the hidden keyword of index by testing one by one keyword in the dictionary. Some recent works (i.e., [14][15][16]) tackle the security against off-line keyword guessing attacks and some work for the version of certificateless keyword search [17].

A threat, called Inner Keyword Guessing Attacks [18, 19], is a new security issue about trapdoor privacy. On receiving a trapdoor from a receiver, a malicious server can easily obtain the hidden low-entropy keyword in the trapdoor by keeping making index with different keywords and then executing tests between the index and the trapdoor. This threat is different from the prior attacks as above, and particularly formalize weaknesses from the correlation between index and trapdoor.

A. RELATED WORKS

To deal with the inner keyword guessing attacks, Huang and Li [20] proposed a solution that an extra authentication token of the sender is added into the index. In other words, the index can only be generated by the sender; and the trapdoor has to include the sender’s identity or public key as well. The malicious server cannot produce the sender-specified token so that it cannot repeatedly compute the index with various keywords and test alone. However, in this architecture, the index has to be bind with sender’s identity, just like a digital signature aside by the index.

Ma et al. [26] considered another solution called witness-based searchable encryption. In this scheme, trapdoor is only valid for ciphertext with a witness relationship; therefore, the malicious server cannot adaptively produce ciphertext to test the trapdoor. Inspired by [26], Liu et al. [27] introduced a new notion called designated-ciphertext searchable encryption. Similar with [26], the trapdoor in this scheme is designated to a ciphertext. Unfortunately, this scheme requires interaction between the sender and the receiver, and may not be applicable in many situations where interaction is not possible.

Chen et al. [18, 19] proposed the first dual-server public key encryption with keyword search (DS-PEKS) to deal with the inner keyword guessing attacks. In a high-level view, the keyword equality testing part is split into two parts operated by two servers so that single malicious server can not run the brute force attack. Informally, there are two non-collusive servers, a front server and a back server, in the new architecture. The front server takes as input a pair of index and trapdoor and executes some computations. Then it outputs an ‘internal-testing-stage’ (ITS) to the back server who takes ITS as input and outputs the keyword search result. The framework of DS-PEKS scheme is depicted in Fig. 1(b). The keyword search test is accomplished by the corporation of two servers. More details about DS-PEKS could be found in Section II-B and VI.

Assumption 1: No collusion occurs between the front server and the back server. The security is discussed based on the assumption that at most one server is malicious.

B. CONTRIBUTIONS

DS-PEKS schemes might be a solution which replaces existing PEKS schemes to be secure against inner keyword guessing attacks. In this work, further than the preliminay work [21] and two previous works [18, 19], we have two main technical advances (new security definition and construction) in a nutshell, which are briefly summarized as follows.

1) New security models. We advocate that existing security models in DS-PEKS are sort of insufficient. Then, referring to previous PEKS works such as [5, 7, 12, 22], we refine them and provide sounder and stronger security models: the indistinguishability against chosen keyword attacks for malicious front servers (IND-CKA-FS) and malicious back servers (IND-CKA-BS). In addition, we show that the new security notions imply all aspects of security in the previous works. However, the previous security models of Chen et al. [18, 19] involve five different types of adversaries, but our models do only two. A byproduct of the new secu-

rity notion is to conceptually simplify and strengthen the security analysis.
An additional assumption. The only assumption proposed by Chen et al. is insufficient to guarantee the semantic security of DS-PEKS schemes. We proposed another assumption, the limit of trial times, to compensate the security model.

2) A generic construction. We advocate that the previous works [18, 19] are not as secure as they claimed. One critical flaw of their work is depicted in the Appendix. Besides, we present a new generic construction of DS-PEKS schemes from IND-CCA2 secure PKE schemes. Furthermore, this construction can meet IND-CKA-FS security and IND-CKA-BS security through a series of rigorous security proofs in the standard model.

C. ORGANIZATION

The rest of this paper, some preliminaries, and building blocks, and syntax and original security models of DS-PEKS schemes will be introduced in Section II. The refined security models will be described in Section III. A generic construction of DS-PEKS schemes based on IND-CCA2 secure PKE schemes (with its security proof) is presented in Section IV. The concrete instantiation and the comparison are provided in Section V. Finally, we conclude this work in Section VI.

II. PRELIMINARIES AND BUILDING BLOCKS

We define an operator '$\xleftarrow{\$}$' as 'randomly chosen from' for further usage; λ represents a secure parameter throughout this work; and a symbol ω denotes the keyword space. Then, several preliminaries and building blocks will be introduced for latter uses. The syntax and original security models of DS-PEKS will be introduced in this section as well.

Definition 1 (Hash function): A cryptographic hash function is a deterministic function that maps elements from the domain space to the codomain space. A hash function H is called one-way if the following probability $Adv^{OW}_{\mathcal{A},H}(\lambda)$ is negligible.

$$Adv^{OW}_{\mathcal{A},H}(\lambda) \overset{def}{=} \Pr[m = m' : m \leftarrow \mathcal{A}(H(m))]$$

Definition 2 (The CDH assumption): The computational Deffie-Hellman assumption (CDH) [23] denotes that given three group elements $(g, g^u, g^v) \in G$ where the order of G is big enough in λ, it is computationally difficult to compute g^{uv}. The CDH assumption is sound if probability $Adv^{CDH}_{\mathcal{A},G}(\lambda)$ is negligible.

$$Adv^{CDH}_{\mathcal{A},G}(\lambda) \overset{def}{=} \Pr[g^{uv} \leftarrow \mathcal{A}(g, g^u, g^v)]$$

Definition 3 (The CONF assumption): Given four group elements $(g, g^u, g^{uv}, Z) \in G$ where Z is randomly picked from G or $Z = g^v$ with the same probability (i.e., with the probability $1/2$, Z is a random number of G or, with the probability $1/2$, $Z = g^v$). The CONF assumption [23] denotes when the order of G is large enough in λ such that the CDH assumption holds, it is computationally infeasible to distinguish whether $Z = g^v$ or not. Let 1 denotes $Z = g^v$ and 0 stands for otherwise, the CONF assumption is sound if probability $Adv^{CONF}_{\mathcal{A},G}(\lambda)$ is negligible.

$$Adv^{CONF}_{\mathcal{A},G}(\lambda) \overset{def}{=} \Pr[1/0 \leftarrow \mathcal{A}(g, g^u, g^{uv}, Z)] - 1/2$$

A. IND-CCA2 SECURE PKE SCHEMES

A public key encryption (PKE) is composed of the following four algorithms.

- Setup(1^λ): it generates the algebra environment like cyclic groups or bilinear pairing for latter use.
- KG(pp): it probabilistically generates a pair of public key pk and secret key sk.
- Enc(m, pk): it probabilistically encrypts a message m into a ciphertext c using the public key pk.
- Dec(c, sk): it deterministically decrypts the ciphertext c with secret key sk to obtain the message m.

The PKE scheme is called indistinguishability against adaptive chosen ciphertext attack (IND-CCA2) secure [24][25] if any polynomial-time adversary $\mathcal{A}$ in the experiment $Exp^{IND-CCA2}_{\mathcal{A},PKE}(\lambda)$ (defined in Fig. 3) has at most negligible advanced probability $Adv^{IND-CCA2}_{\mathcal{A},PKE}(\lambda)$ to win. The advanced probability $Adv^{IND-CCA2}_{\mathcal{A},PKE}(\lambda)$ is defined as $Adv^{IND-CCA2}_{\mathcal{A},PKE}(\lambda) \overset{def}{=} \Pr[b = b' : b \leftarrow Exp^{IND-CCA2-b}_{\mathcal{A},PKE}(\lambda)] - 1/2$.

B. SYNTAX OF DS-PEKS

We firstly revisit the framework of DS-PEKS defined in Chen et al.'s works [18][19]. Let Setup, KeyGen, BuildIndex, Trapdoor, FrontTest, BackTest denote a DS-PEKS scheme, it is formalized below. Among these six algorithms, only the BackTest is a deterministic algorithm that outputs 1 for tested match or 0 for mismatch. Other five algorithms are probabilistic algorithms that takes random numbers into computation.

- Setup(1^λ): Take a secure parameter λ as input, this algorithm generates a series of public parameters pp.
- KeyGen(pp): On input pp, this algorithm generates two pairs of public/secret keys (pk_{FS}, sk_{FS}) and (pk_{BS}, sk_{BS}) for front server and back server, respectively.
- BuildIndex(pp, w, pk_{FS}, pk_{BS}): Anyone can build an index corresponding to a keyword w using both public keys of the front server and the back server. The algorithm outputs an index c_w at the end.
- Trapdoor(pp, w_0, pk_{FS}, pk_{BS}): Anyone is able to generate a trapdoor $t_{w'}$ of a keyword w' using two servers' public keys.
- FrontTest(pp, $c_w, t_{w'}, sk_{FS}$): On receiving an index c_w and a trapdoor $t_{w'}$, the front server computes an 'internal testing-state' ITS using its private key sk_{FS}. Then it sends ITS to the back server.

- BackTest(pp, ITS, sk_{BS}): After receiving ITS, the back server executes a deterministic back test using its private key sk_{BS}. It outputs 1 if $w = w'$ or 0 otherwise.

Comparison with tradictional PEKS: Compared to original PEKS schemes with single server (Setup, KeyGen, BuildIndex, Trapdoor, Test) such as [5][7][12][22], two key differences of the newly defined DS-PEKS architecture are observed here.

1) Algorithm Test is replaced by two algorithms: FrontTest and BackTest. The test server who is responsible for the Test operation has been divided into two servers, i.e. the front server FS and the back server BS.
2) Algorithm Trapdoor becomes publicly computable in the DS-PEKS framework, which differs from the setting in the singleserver PEKS setting that the trapdoor, as a search request, can only be requested by the corresponding secret key owner.

Both two modifications above are designed on the purpose against the inner keyword-guessing attack. As mentioned in Section I, most single-server PEKS schemes cannot be semantic secure owing to the inner keyword-guessing attack. Together with the assumption that at least one server in this system is honest and uncompromised; then, in this scenarion, the probability of the inner keyword-guessing attck is expected eliminated to negligible.

C. CHEN ET AL.'S SECURITY NOTIONS

Interestingly, in the original single-server public key encryption with keyword search schemes [5][7][12][22], they discuss the chosen keyword attacks against malicious users. However, in DS-PEKS scheme, the notions of security is discussed against malicious front / back servers (inner adversaries).

When it comes to the discussion about which role does the adversary plays, it could be an outside attacker who has no additional ability; or it could be one of two servers. Obviously, the latter one owns absolutely greater power than the former one; besides, the security against one of two servers implies the security against outside attackers so that the discussion only focuses on the senario that the adversary plays one malicious server out of two servers, which the other server is uncompromised.

The security issues in Chen et al.'s works [18][19] concerns three parts (described in Fig. 2): (i) the privacy of index (SS-CKA), (ii) the privacy of trapdoor (IND-KGA) and (iii) the privacy on considering both index and trapdoor (IND-KGA-II).

1) Privacy of a single index or trapdoor: Informally speaking, both SS-CKA and IND-KGA are similar to the adaptive chosen ciphertext attacks (IND-CCA2) for public key encryption schemes. With the aids of oracles, the adversary $\mathcal{A} \in \{FS, BS\}$ outputs two keywords for the simulator. The simulator randomly selects one of them and encrypts it into a challenge index / trapdoor. After being assigned a challenge, the adversary has to recognize the selected keyword to win the game. They are formalized as experiments shown in Fig. 2; and they ensure neither w_0 nor w_1 has been queried to test oracles before returning b'. The DS-PEKS scheme is called SS-CKA secure or IND-KGA secure if no polynomial-time adversary has capability of earning a non-negligible probability $Adv^{SS-CKA}_{\mathcal{A},DS-PEKS}(\lambda)$ or $Adv^{IND-KGA}_{\mathcal{A},DS-PEKS}(\lambda)$, respectively.
$Adv^{SS-CKA}_{\mathcal{A},DS-PEKS}(\lambda) \overset{def}{=} \Pr[b = b' : b' \leftarrow Exp^{SS-CKA-b}_{\mathcal{A},DS-PEKS}(\lambda)]$
$Adv^{IND-KGA}_{\mathcal{A},DS-PEKS}(\lambda) \overset{def}{=} \Pr[b = b' : b' \leftarrow Exp^{IND-KGA-b}_{\mathcal{A},DS-PEKS}(\lambda)]$
2) Privacy on considering both index and trapdoor: The security model IND-KGA-II concerns the security on input an index and a trapdoor, which is formally described in Fig. 2. Informally speaking, the simulator randomly picks $(b_1, b_2) \overset{\$}{\leftarrow} \{0,1\}$ and computes $c_{w_{b1}}$ and $t_{w_{b2}}$ after the adversary outputs three keywords $(w_0, w_1, w_2), (w_0 \neq w_1 \neq w_2)$. The adversary terminates the game by outputting a guess $\{b_1', b_2'\} \in \{0,1\}^2$ and it wins if $\{b_1, b_2\} = \{b_1', b_2'\}$. It is interesting that $(b_1 = b_1') \wedge (b_2 = b_2')$ is not necessary, the adversary also wins when $(b_1 = b_2') \wedge (b_2 = b_1')$. A DS-PEKS scheme is called IND-KGA-II secure if the following equation holds.
$\Pr\left[\begin{array}{l}\{b_1, b_2\} = \{b_1', b_2'\} : \\ \{b_1', b_2'\} \leftarrow Exp^{IND-KGA-II-b_1,b_2}_{\mathcal{A},DS-PEKS}(\lambda)\end{array}\right] - 3/8 \leq negl(\lambda)$
Worth noting that no oracle access is available in the IND-KGA-II model. On the other hand, it only defines the security against malicious back servers.

III. REFINEMENTS OF SECURITY NOTIONS

In this section, we firstly analyze the security models defined by Chen et al. and indicate some points with respect to the soundness and strength. Then, we introduce our refinements of the security models which are more realistic and can imply previous ones. In particular, there is a byproduct of our refined models (but technically useful) that essentially simply the procedure of the security analysis.

A. ANALYSIS OF EXISTING SECURITY MODELS

In a high-level view, the security models are designed to prevent the front server or the back server to individually distinguish the keyword hidden in the index or trapdoor. For those existing security models, we argue that existing security models are not sound and strong enough, where sound means 'close to the DS-PEKS syntax'.

1) In terms of soundness, we firstly recall the syntax of DS-PEKS schemes that the front server inputs an index and a trapdoor as well as outputs ITS; and the back server inputs ITS as well as outputs 1 for verified match or 0 otherwise. The front server is responsible to generate an ITS, and the back server outputs the

```
$\mathrm{Exp}^{\text{SS-CKA-}b}_{\mathcal{A}\in\{\mathtt{FS},\mathtt{BS}\},\mathtt{DS\text{-}PEKS}}(\lambda)$
    $\mathrm{Set} \leftarrow \emptyset$; $\mathsf{pp} \leftarrow \mathtt{Setup}(1^\lambda)$;
    $(pk_{\mathtt{FS}}, sk_{\mathtt{FS}}, pk_{\mathtt{BS}}, sk_{\mathtt{BS}}) \leftarrow \mathtt{KeyGen}(\mathsf{pp})$;
    $(w_0, w_1) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{T}_1}(\cdot,\cdot)}(\mathsf{pp}, pk_{\mathtt{FS}}, pk_{\mathtt{BS}}, sk_{\mathcal{A}})$;
                                                    $\triangleright\, w_0 \neq w_1$
    $c_{w_b} \leftarrow \mathtt{BuildIndex}(\mathsf{pp}, w_b, pk_{\mathtt{FS}}, pk_{\mathtt{BS}})$;
    $b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{T}_1}(\cdot,\cdot)}(c_{w_b})$;
    return $b'$;

$\mathcal{O}_{\mathrm{T}_1}(c_w, w')$
    $\mathrm{Set} \leftarrow \mathrm{Set} \cup w'$;
    $t_{w'} \leftarrow \mathtt{Trapdoor}(\mathsf{pp}, w', pk_{\mathtt{FS}}, pk_{\mathtt{BS}})$;
    $\mathtt{ITS} \leftarrow \mathtt{FrontTest}(\mathsf{pp}, c_w, t_{w'}, sk_{\mathtt{FS}})$;
    $x \leftarrow \mathtt{BackTest}(\mathsf{pp}, \mathtt{ITS}, sk_{\mathtt{BS}})$;
    return $x$;
```

```
$\mathrm{Exp}^{\text{IND-KGA-}b}_{\mathcal{A}\in\{\mathtt{FS},\mathtt{BS}\},\mathtt{DS\text{-}PEKS}}(\lambda)$
    $\mathrm{Set} \leftarrow \emptyset$; $\mathsf{pp} \leftarrow \mathtt{Setup}(1^\lambda)$;
    $(pk_{\mathtt{FS}}, sk_{\mathtt{FS}}, pk_{\mathtt{BS}}, sk_{\mathtt{BS}}) \leftarrow \mathtt{KeyGen}(\mathsf{pp})$;
    $(w_0, w_1) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{T}_2}(\cdot,\cdot)}(\mathsf{pp}, pk_{\mathtt{FS}}, pk_{\mathtt{BS}}, sk_{\mathcal{A}})$;
                                                    $\triangleright\, w_0 \neq w_1$
    $t_{w_b} \leftarrow \mathtt{Trapdoor}(\mathsf{pp}, w_b, pk_{\mathtt{FS}}, pk_{\mathtt{BS}})$;
    $b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{T}_2}(\cdot,\cdot)}(t_{w_b})$;
    return $b'$;

$\mathcal{O}_{\mathrm{T}_2}(t_{w'}, w)$
    $\mathrm{Set} \leftarrow \mathrm{Set} \cup w$;
    $c_w \leftarrow \mathtt{BuildIndex}(\mathsf{pp}, w, pk_{\mathtt{FS}}, pk_{\mathtt{BS}})$;
    $\mathtt{ITS} \leftarrow \mathtt{FrontTest}(\mathsf{pp}, c_w, t_{w'}, sk_{\mathtt{FS}})$;
    $x \leftarrow \mathtt{BackTest}(\mathsf{pp}, \mathtt{ITS}, sk_{\mathtt{BS}})$;
    return $x$;
```

```
$\mathrm{Exp}^{\text{IND-KGA-II-}b_1,b_2}_{\mathcal{A}=\mathtt{BS},\mathtt{DS\text{-}PEKS}}(\lambda)$
    $\mathsf{pp} \leftarrow \mathtt{Setup}(1^\lambda)$;
    $(pk_{\mathtt{FS}}, sk_{\mathtt{FS}}, pk_{\mathtt{BS}}, sk_{\mathtt{BS}}) \leftarrow \mathtt{KeyGen}(\mathsf{pp})$;
    $(w_0, w_1, w_2) \leftarrow \mathcal{A}(\mathsf{pp}, pk_{\mathtt{FS}}, pk_{\mathtt{BS}}, sk_{\mathtt{BS}})$;
                                                    $\triangleright\, w_0 \neq w_1 \neq w_2$
    $c_{w_{b_1}} \leftarrow \mathtt{BuildIndex}(\mathsf{pp}, w_{b_1}, pk_{\mathtt{FS}}, pk_{\mathtt{BS}})$;
    $t_{w_{b_2}} \leftarrow \mathtt{Trapdoor}(\mathsf{pp}, w_{b_2}, pk_{\mathtt{FS}}, pk_{\mathtt{BS}})$;
    $\mathtt{ITS} \leftarrow \mathtt{FrontTest}(\mathsf{pp}, c_{w_{b_1}}, t_{w_{b_2}}, sk_{\mathtt{FS}})$;
    $(b'_1, b'_2) \leftarrow \mathcal{A}(\mathsf{pp}, \mathtt{ITS})$;
    if $(\{b_1, b_2\} = \{b'_1, b'_2\})$ then return $1$
    else return $0$;
    end if
```

FIGURE 2: Chen et al.'s security models: SS-CKA, IND-KGA and IND-KGA-II.

equivalence, while none of them should be able to distinguish the keywords.

a) For the front server, the SS-CKA security and IND-KGA security denote the 'keyword indistinguishability' on seeing a single index or a single trapdoor, respectively. However, the front server inputs them in the same time. An ideal condition is that the front server generates the corresponding ITS without acquiring any knowledge from the input index and trapdoor.
b) For the back server, despite the fact that the back server might not access the index and trapdoor directly, it might eavesdrop the index and trapdoor from the public network. Furthermore, ITS is a part of input in the BackTest algorithm, which gives the back server more power and opportunity to distinguish the keywords than the front server. In an ideal situation, on input the index, the trapdoor and the ITS, the back server should output the equivalence without learning anything else.
c) For oracle accesses, take $\mathcal{O}_{T_1}(\cdot,\cdot)$ in experiment SS-CKA for example, it takes as input a keyword w' and an index c_w, computes $t_{w'} \leftarrow$ Trapdoor(pp, w', pk_{FS}, pk_{BS}) and outputs the equivalence between w and w'. The oracle plays both the front server party and the back server party (FrontTest(pp, $c_w, t_{w'}, sk_{FS}$) and BackTest(pp, ITS, sk_{BS})). However, the adversary in experiment SS-CKA has its own capability to execute one of FrontTest or BackTest. First, half of the 'two-server help' provided by oracle $\mathcal{O}_{T_1}(\cdot,\cdot)$ is redundant. Second, in case $A = BS$, there might be some helpful information hidden in the ITS, but ITS is absolutely unavailable in this kind of oracle accesses. To summarize, the oracle $\mathcal{O}_{T_1}(\cdot,\cdot)$ in experiment SS-CKA is redundant, and it might weaken the probability of winning experiment SS-CKA. The similar problem occurs with the oracle $\mathcal{O}_{T_2}(\cdot,\cdot)$ in experiment IND-KGA.

2) In terms of strength, the original 'chosen keyword attack' defined in [5] is an adaptive security with oracle accesses, which is similar to the IND-CCA2 security for PKE schemes. However, the IND-KGA-II model in Chen et al.'s works do not provide oracle accesses to adversaries. In other words, the IND-KGA-II security for DS-PEKS schemes is similar to the IND-CPA security for PKE schemes, which is not an adaptive security.

We advocate that the security models of DS-PEKS schemes can be sounder and stronger. In the following section, we propose two conceptually realistic security definitions to conquer all aforementioned issues.

B. CONCEPTUALLY REALISTIC SECURITY DEFINITIONS

Recall the security notions shown in Section II-C. Apparently, the inputs and outputs are different between the front server and the back server. Hence, unlike the previous works that categorize security models based on index and trapdoor, we define the indistinguishability against chosen keyword attacks for front server (IND-CKA-FS), as well as for back server (IND-CKA-BS) in Fig. 3.

Definition 4: The indistinguishability of chosen keyword attack against malicious front servers, IND-CKA-FS.

Informally, IND-CKA-FS can be regarded as the combinational version of SS-CKA and IND-KGA (against malicious front server), whose challenge including the index and trapdoor at the same time. In addition, the oracle access $\mathcal{O}_{BT}(\cdot)$ is answered by the simulator who plays the back server role in the DS-PEKS scheme. We emphasize that the newly defined IND-CKA-FS security not only implies the SS-CKA security and the IND-KGA security but also achieves higher security level than them because of following reasons:

1) Adversary's key pair is chosen and fully controlled by itself, which is unknown to the simulator.
2) The oracle access is sounder in the DS-PEKS architecture without losing its help to the adversary. Compared to previous oracle settings, oracle $\mathcal{O}_{BT}(\cdot)$ in IND-CKA-FS provides the same assistant.

3) For the challenge in IND-CKA-FS, the index and trapdoor are considered together to discuss their security, which is sounder in the DS-PEKS syntax.

Restriction. An important restriction is worthy to be emphasized that the ITS generated from FrontTest($c_{w_b}, \cdot$) or FrontTest($\cdot, t_{w_b}$), the simulator need to recognize, is forbidden to be queried to oracle $\mathcal{O}_{BT}(\cdot)$. In our opinion, the simulator should be able to identify above situation if the DS-PEKS scheme is IND-CKA-FS security.

Assumption 2: The trial time limit. For both servers, a specific index can be requested to test only in a limited times; e.x. 3 trials in 1 minute. It is not in Chen et al.'s works; nevertheless, it is one of the key points to guarantee the semantic security.

Definition 5: The indistinguishability of chosen keyword attack against malicious back servers, IND-CKA-BS.

In principle, the back server in the DS-PEKS syntax is expected to tell the equivalence of keywords between the index side and the trapdoor side (with the aid of front server), rather than being able to distinguish the keywords on both sides. Owing to this principle, the experiment IND-CKA-BS is different from 'common indistinguishability experiments' that distinguish the chosen input from the challenge.

The newly proposed IND-CKA-BS model can be seen as a upgraded IND-KGA-II model which integrates the SS-CKA security and the IND-KGA security (against malicious back server); and IND-CKA-BS implies all of them. Roughly, the adversary has to output a correct b'_1 for index security (SS-CKA), and relatively, a correct b'_2 denotes for trapdoor security (IND-KGA) at the same time. Since the adversary who plays the back server role has capability to verify the equivalence of keywords between the index side and the trapdoor side on input the challenge ITS, recognizing the keyword of index (w_{b_1}) implies knowing the keyword of trapdoor (w_{b_2}), and vice versa. The IND-CKA-BS takes all previous security models into an overall consideration. Compared to previous security models, it is sounder and higher-level security due to the following reasons:

1) Adversary's key pair is chosen and fully controlled by itself, which is unknown to the simulator.
2) The oracle access is sounder in the DS-PEKS architecture without losing its help to the adversary. That is, the oracle access in IND-CKA-BS provides the same assistant as those in SS-CKA and IND-KGA.
3) For the challenge of the IND-CKA-BS model, the index, trapdoor and ITS are jointly considered to discuss their security, which is sounder in the DS-PEKS syntax.

C. REDUCTIONS AMONG SECURITY MODELS

Now we give proofs of our claims in which the refined security models can imply the previous ones shown in Section II-C.

Theorem 1: IND-CKA-FS secure implies SS-CKA and IND-KGA secure.

Proof 1: The reduction is illustrated in Fig. 4. Some different settings which differ from different experiment are explained below.

- In the SS-CKA case, $\mathcal{O}_T = \mathcal{O}_{T_1}$ and $\mathcal{C} = c_{w_{b_1}}$. The IND-CKA-FS secure directly implies the SS-CKA secure against malicious front servers.
- In the IND-KGA case, $\mathcal{O}_T = \mathcal{O}_{T_2}$ and $\mathcal{C} = c_{w_{b_2}}$. The IND-CKA-FS secure directly implies the IND-KGA secure against malicious front servers.

Theorem 2: IND-CKA-BS secure implies SS-CKA, IND-KGA and IND-KGA-II secure.

Proof 2: The reduction illustrated in Fig. 5. Some different settings which differ from different experiment are explained below.

- In the SS-CKA case, $\mathcal{O}_T = \mathcal{O}_{T_1}$, $\mathcal{C} = c_{w_{b_1}}$, $W = (w_0, w_1)$ and $\mathcal{B}' = \mathcal{B}, b'_2$. The simulator executes a BackTest before outputs $\mathcal{B}'$. If BackTest(pp, $ITS*, sk_{BS}$) = 1, $b'_2 \leftarrow \mathcal{B}$; otherwise $b'_2 \leftarrow (1 - \mathcal{B})$. The IND-CKA-BS secure directly implies the SS-CKA secure against malicious back servers.
- In the IND-KGA case, $\mathcal{O}_T = \mathcal{O}_{T_2}$, $\mathcal{C} = c_{w_{b_2}}$, $W = (w_0, w_1)$ and $\mathcal{B}' = \mathcal{B}, b'_1$. The simulator executes a BackTest before outputs $\mathcal{B}'$. If BackTest(pp, $ITS*, sk_{BS}$) = 1, $b'_1 \leftarrow \mathcal{B}$; otherwise $b'_1 \leftarrow (1-\mathcal{B})$. The IND-CKA-BS secure directly implies the IND-KGA secure against malicious back servers.
- In the IND-KGA-II case, $\mathcal{O}_T = \perp$, $C = ITS*$, $W = (w_0, w_1, w_2)$ and $\mathcal{B}' = \mathcal{B}$. The relationship of winning IND-CKA-BS and winning IND-KGA-II is discussed below:
  Assume that there is an adversary A who has non-negligible probability $\in$ to win experiment IND-KGA-II, a simulator can follow operations illustrated in Fig. 5 to win experiment IND-CKA-BS with non-negligible probability.
  The public parameters and key settings are depicted in Fig. 5. On input $W = (w_0, w_1, w_2)$ from the adversary, the simulator randomly picks two of them and delivers to experiment IND-CKA-BS. Without loss of generality, let (w_0, w_1) are selected and delivered. Then, after receiving the challenge $(c_{w_{b_1}}, t_{w_{b_2}}, ITS*)$, the simulator sends $ITS*$ to the adversary and waits for its output $\mathcal{B} = (b'_1, b'_2)$. Finally, the simulator outputs $\mathcal{B}' = \mathcal{B}$ to IND-CKA-BS. Following, it is discussed case by case.
  - If $b'_1 = b'_2$, which occurs in half probability. The adversary has advanced probability $\in$ that $b'_1 = b'_2 = b_1 = b_2$. The simulator inherently wins with advanced probability $\in$.
  - If $b'_1 \neq b'_2$, which also occurs in half probability. Two possible scenarios including (1) $(b'_1 = b_1) \wedge (b'_2 = b_2)$ and (2) $(b'_1 = b_2) \wedge (b'_2 = b_1)$ might happen, but the simulator has no idea to distinguish. No advanced probability is obtained in this scenario.

```
$\mathtt{Exp}^{\text{IND-CKA-FS-}b}_{\mathcal{A}=\mathtt{FS},\mathtt{DS\text{-}PEKS}}(\lambda)$
    $\mathtt{pp} \leftarrow \mathtt{Setup}(1^\lambda)$;
    $(pk_{\mathtt{BS}}, sk_{\mathtt{BS}}) \leftarrow \mathtt{KeyGen}(\mathtt{pp})$;
    $(w_0, w_1, pk_{\mathtt{FS}}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathtt{BT}}(\cdot)}(\mathtt{pp}, pk_{\mathtt{BS}})$;
                                                    $\triangleright\ w_0 \neq w_1$
    $c_{w_b} \leftarrow \mathtt{BuildIndex}(\mathtt{pp}, w_b, pk_{\mathtt{FS}}, pk_{\mathtt{BS}})$;
    $t_{w_b} \leftarrow \mathtt{Trapdoor}(\mathtt{pp}, w_b, pk_{\mathtt{FS}}, pk_{\mathtt{BS}})$;
    $b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathtt{BT}}(\cdot)}(\mathtt{pp}, pk_{\mathtt{BS}}, c_{w_b}, t_{w_b})$;
    return $b'$;

$\mathcal{O}_{\mathtt{BT}}(\mathtt{ITS})$
    return $\mathtt{BackTest}(\mathtt{pp}, \mathtt{ITS}, sk_{\mathtt{BS}})$;
    $\triangleright$ A restriction is defined in Definition 1.
```

```
$\mathtt{Exp}^{\text{IND-CKA-BS-}b_1,b_2}_{\mathcal{A}=\mathtt{BS},\mathtt{DS\text{-}PEKS}}(\lambda)$
    $c_{w_{b_1}} \leftarrow \perp$; $t_{w_{b_2}} \leftarrow \perp$; $\mathtt{pp} \leftarrow \mathtt{Setup}(1^\lambda)$;
    $(pk_{\mathtt{FS}}, sk_{\mathtt{FS}}) \leftarrow \mathtt{KeyGen}(\mathtt{pp})$;
    $(w_0, w_1, pk_{\mathtt{BS}}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathtt{FT}}(\cdot,\cdot)}(\mathtt{pp}, pk_{\mathtt{FS}})$;
                                                    $\triangleright\ w_0 \neq w_1$
    $c_{w_{b_1}} \leftarrow \mathtt{BuildIndex}(\mathtt{pp}, w_{b_1}, pk_{\mathtt{FS}}, pk_{\mathtt{BS}})$;
    $t_{w_{b_2}} \leftarrow \mathtt{Trapdoor}(\mathtt{pp}, w_{b_2}, pk_{\mathtt{FS}}, pk_{\mathtt{BS}})$;
    $\mathtt{ITS}^* \leftarrow \mathtt{FrontTest}(\mathtt{pp}, c_{w_{b_1}}, t_{w_{b_2}}, sk_{\mathtt{FS}})$;
    $(b'_1, b'_2) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathtt{FT}}(\cdot,\cdot)}(c_{w_{b_1}}, t_{w_{b_2}}, \mathtt{ITS}^*)$;
    if $((b_1 = b'_1) \wedge (b_2 = b'_2))$ then return 1
    else return 0;
    end if

$\mathcal{O}_{\mathtt{FT}}(c_w, t_{w'})$
    if $((c_w = c_{w_{b_1}}) \vee (t_{w'} = t_{w_{b_2}}))$ then
        break;
    else
        $\mathtt{ITS} \leftarrow \mathtt{FrontTest}(\mathtt{pp}, c_w, t_{w'}, sk_{\mathtt{FS}})$;
        return $\mathtt{ITS}$;
    end if
```

```
$\mathtt{Exp}^{\text{IND-CCA2-}b}_{\mathcal{A},\mathtt{PKE}}(\lambda)$
    $c_b \leftarrow \perp$; $\mathtt{pp}_{\mathtt{cca}} \leftarrow \mathtt{Setup}_{\mathtt{cca}}(1^\lambda)$;
    $(pk_{\mathtt{cca}}, sk_{\mathtt{cca}}) \leftarrow \mathtt{KG}_{\mathtt{cca}}(\mathtt{pp}_{\mathtt{cca}})$;
    $(m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}_D(\cdot)}(\mathtt{pp}_{\mathtt{cca}}, pk_{\mathtt{cca}})$;
                                                    $\triangleright\ m_0 \neq m_1$
    $c_b \leftarrow \mathtt{Enc}(m_b, pk_{\mathtt{cca}})$;
    $b^* \leftarrow \mathcal{A}^{\mathcal{O}_D(\cdot)}(c_b)$;
    return $b^*$;

$\mathcal{O}_D(c)$
    if $(c = c_b)$ then break;
    else return $\mathtt{Dec}(c, sk_{\mathtt{cca}})$;
    end if
```

FIGURE 3: New IND-CKA-FS notion and IND-CKA-BS notion for DS-PEKS schemes; and IND-CCA2 notion for PKE schemes.

On one hand, an overall advanced probability $\in/2$ will be obtained to break IND-CKA-BS if an adversary who has non-negligible probability $\in$ to break IND-KGA-II exists. The security of IND-KGA-II relies on the security of IND-CKA-BS. On the other hand, it is obvious that if one can break IND-CKA-BS, it can directly break IND-KGA-II with $(b'_1 = b_1) \wedge (b'_2 = b_2)$ where (b_1, b_2) comes from the break of IND-CKA-BS. We can say IND-CKA-BS secure implies IND-KGA-II secure.

IV. GENERIC CONSTRUCTION FROM IND-CCA2 SECURE PUBLIC KEY ENCRYPTION

In this section, we take IND-CCA2 secure PKE schemes like Cramer and Shoup's schemes [24][25] as a building block to construct a DS-PEKS scheme. Let $(Setup_{cca}$, KG_{cca}, Enc, $Dec)$ be an IND-CCA2 secure PKE scheme with public parameters $pp_{cca} \leftarrow Setup_{cca}(1^\lambda)$ which including a multiplicative cyclic group G, a prime order q and a generator g. The keyword space $\mathcal{W}$ is a small subgroup of Z_q^*.

- Setup(1^λ): The algorithm randomly selects $h \xleftarrow{\$} G$ and a one-way hash function $H : G \to \{0,1\}^\lambda$. The discrete logarithm problem of h over base g is unknown. Then, it outputs pp $\leftarrow \{pp_{cca}, h, H\}$.
- KeyGen(pp): The front server picks $x \xleftarrow{\$} Z_q^*$ as private key sk_{FS} and publishes its public key $pk_{FS} = X$ where $X \leftarrow g^x$. The back server generates its key pair by computing $(pk_{BS}, sk_{BS}) \leftarrow KG_{cca}(pp_{cca})$.
- BuildIndex(pp, w, pk_{FS}, pk_{BS}): It picks $r \xleftarrow{\$} Z_q^*$ and outputs $c_w \leftarrow g^r h^w, Enc(X^r, pk_{BS})$.
- Trapdoor(pp, w', pk_{FS}, pk_{BS}): It selects $r' \xleftarrow{\$} G$ and outputs $t'_w \leftarrow g^{r'} h^{-w'}, Enc(X^{r'}, pk_{BS})$.
- FrontTest(pp, $c_w, t_{w'}, sk_{FS}$): The front server firstly parses $(c^{[1]}, c^{[2]}) \leftarrow c_w$ and $(t^{[1]}, t^{[2]}) \leftarrow c_{w'}$. Then, it picks $R \xleftarrow{\$} G$ and outputs ITS $\leftarrow (c^{[2]}, t^{[2]}, H(R), (c^{[1]} \cdot t^{[1]})^x R)$.
- BackTest(pp, ITS, sk_{BS}): The back server parses ITS as $(I^{[1]}, I^{[2]}, I^{[3]}, I^{[4]})$ and outputs 1 if the following equation holds; or 0, otherwise.

$$I^{[3]} = H(I^{[4]}/(Dec(I^{[1]}, sk_{BS}) \cdot Dec(I^{[2]}, sk_{BS}))) \quad (1)$$

Correctness. The perfect correctness is clear when $w = w'$,

$$\begin{aligned} & H(I^{[4]}/(Dec(I^{[1]}, sk_{BS}) \cdot Dec(I^{[2]}, sk_{BS})) \\ = & H(((c^{[1]} \cdot t^{[1]})^x R)/(X^r \cdot X^{r'})) \\ = & H(((g^r h^w \cdot g^{r'} h^{-w'})^x R)/(X^r \cdot X^{r'})) = H(R) = I^{[3]} \end{aligned}$$

A. SECURITY PROOF

We first recall the definition of IND-CCA2, which is shown as the experiment $Exp^{IND-CCA2-b}_{\mathcal{A},PKE}(\lambda)$ in Figure 3. The IND-CCA2 security is guaranteed if no polynomial-time adversary can obtain a non-negligible advanced probability $Adv^{IND-CCA2}_{\mathcal{A},PKE}(\lambda)$ where $Adv^{IND-CCA2}_{\mathcal{A},PKE}(\lambda) \overset{def}{=} \Pr[b* = b : b* \leftarrow Exp^{IND-CCA2-b}_{\mathcal{A},PKE}(\lambda)] - 1/2$.

Theorem 3: The generic DS-PEKS construction is IND-CKA-FS secure based on the IND-CCA2 security of the applied PKE scheme.

Proof 3: In the experiment, $\mathcal{S}$ plays two roles: the first one is a challenger (who takes advantage of $\mathcal{A}$) to challenge the IND-CCA2 experiment for the applied PKE schemes; and the second one is the back server and oracle responser in the IND-CKA-FS experiment. As an challenger, the simulator first receives pp_{cca} and pk_{BS} from the IND-CCA2 experiment. It picks $h \xleftarrow{\$} G$ and H and delivers pp, pk_{BS} to the adversary where pp$\leftarrow \{pp_{cca}, h, H\}$ as the beginning of the IND-CKA-FS experiment. The formal description about

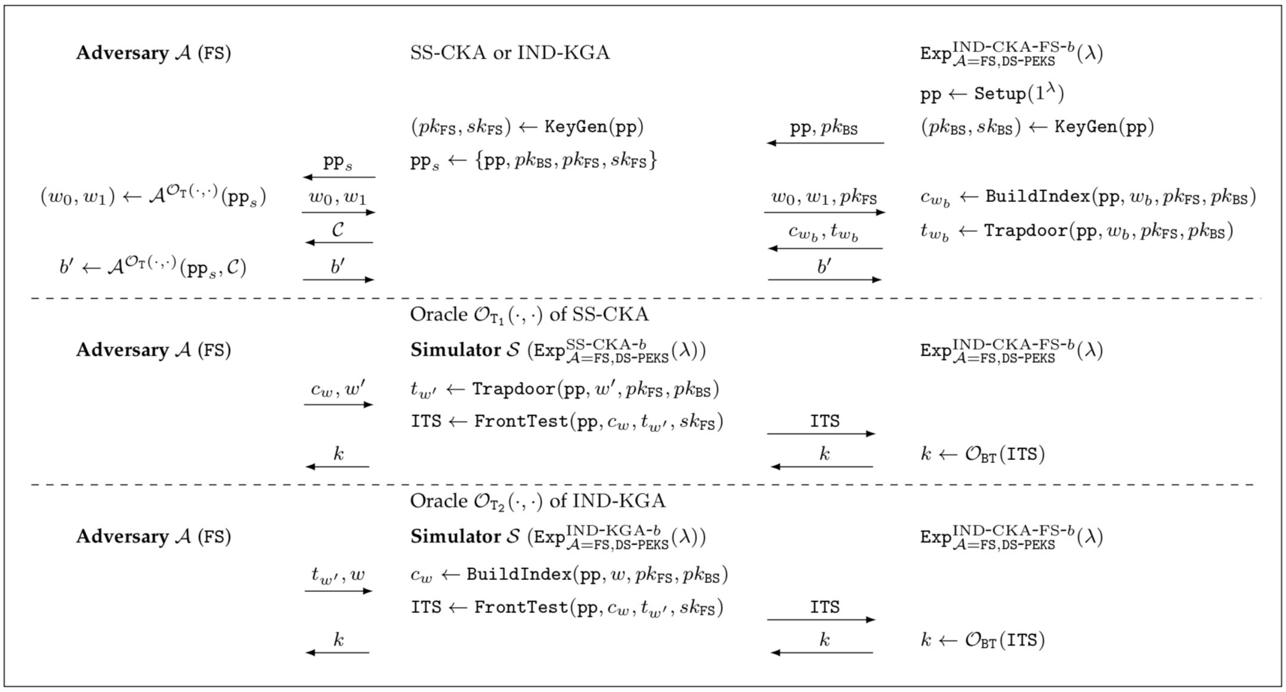

Adversary $\mathcal{A}$ (FS)
SS-CKA or IND-KGA
$\mathsf{Exp}^{\text{IND-CKA-FS-}b}_{\mathcal{A}=\mathtt{FS},\mathtt{DS\text{-}PEKS}}(\lambda)$
$\mathtt{pp} \leftarrow \mathtt{Setup}(1^\lambda)$
$(pk_{\mathtt{FS}}, sk_{\mathtt{FS}}) \leftarrow \mathtt{KeyGen}(\mathtt{pp})$
$\mathtt{pp}, pk_{\mathtt{BS}}$
$(pk_{\mathtt{BS}}, sk_{\mathtt{BS}}) \leftarrow \mathtt{KeyGen}(\mathtt{pp})$
$\mathtt{pp}_s$
$\mathtt{pp}_s \leftarrow \{\mathtt{pp}, pk_{\mathtt{BS}}, pk_{\mathtt{FS}}, sk_{\mathtt{FS}}\}$
$(w_0, w_1) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathtt{T}}(\cdot,\cdot)}(\mathtt{pp}_s)$
w_0, w_1
$w_0, w_1, pk_{\mathtt{FS}}$
$c_{w_b} \leftarrow \mathtt{BuildIndex}(\mathtt{pp}, w_b, pk_{\mathtt{FS}}, pk_{\mathtt{BS}})$
$\mathcal{C}$
c_{w_b}, t_{w_b}
$t_{w_b} \leftarrow \mathtt{Trapdoor}(\mathtt{pp}, w_b, pk_{\mathtt{FS}}, pk_{\mathtt{BS}})$
$b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathtt{T}}(\cdot,\cdot)}(\mathtt{pp}_s, \mathcal{C})$
b'
b'
Oracle $\mathcal{O}_{\mathtt{T}_1}(\cdot,\cdot)$ of SS-CKA
Adversary $\mathcal{A}$ (FS)
Simulator $\mathcal{S}$ $(\mathsf{Exp}^{\text{SS-CKA-}b}_{\mathcal{A}=\mathtt{FS},\mathtt{DS\text{-}PEKS}}(\lambda))$
$\mathsf{Exp}^{\text{IND-CKA-FS-}b}_{\mathcal{A}=\mathtt{FS},\mathtt{DS\text{-}PEKS}}(\lambda)$
c_w, w'
$t_{w'} \leftarrow \mathtt{Trapdoor}(\mathtt{pp}, w', pk_{\mathtt{FS}}, pk_{\mathtt{BS}})$
$\mathtt{ITS} \leftarrow \mathtt{FrontTest}(\mathtt{pp}, c_w, t_{w'}, sk_{\mathtt{FS}})$
ITS
k
k
$k \leftarrow \mathcal{O}_{\mathtt{BT}}(\mathtt{ITS})$
Oracle $\mathcal{O}_{\mathtt{T}_2}(\cdot,\cdot)$ of IND-KGA
Adversary $\mathcal{A}$ (FS)
Simulator $\mathcal{S}$ $(\mathsf{Exp}^{\text{IND-KGA-}b}_{\mathcal{A}=\mathtt{FS},\mathtt{DS\text{-}PEKS}}(\lambda))$
$\mathsf{Exp}^{\text{IND-CKA-FS-}b}_{\mathcal{A}=\mathtt{FS},\mathtt{DS\text{-}PEKS}}(\lambda)$
$t_{w'}, w$
$c_w \leftarrow \mathtt{BuildIndex}(\mathtt{pp}, w, pk_{\mathtt{FS}}, pk_{\mathtt{BS}})$
$\mathtt{ITS} \leftarrow \mathtt{FrontTest}(\mathtt{pp}, c_w, t_{w'}, sk_{\mathtt{FS}})$
ITS
k
k
$k \leftarrow \mathcal{O}_{\mathtt{BT}}(\mathtt{ITS})$

FIGURE 4: IND-CKA-FS secure implies both SS-CKA and IND-KGA secure. The oracle $\mathcal{O}_T$ and the challenge $\mathcal{C}$ are defined in Definition 1.

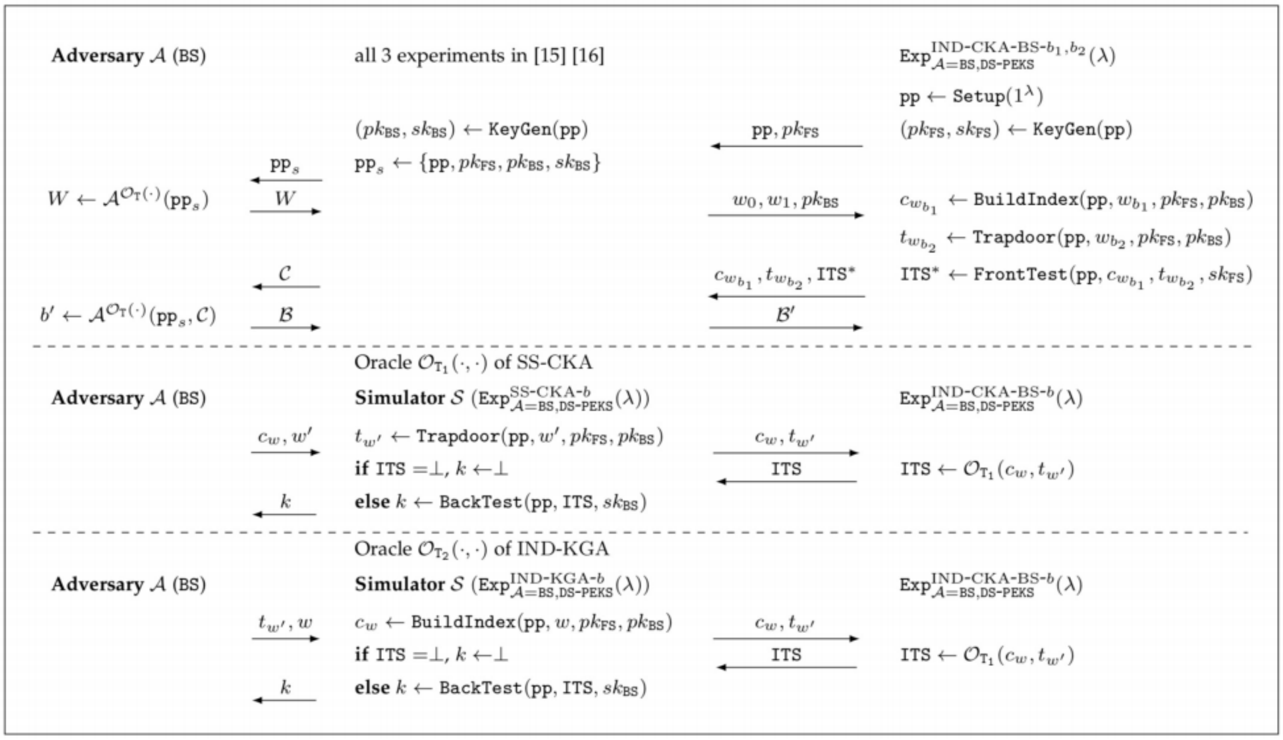

Adversary $\mathcal{A}$ (BS)
all 3 experiments in [15] [16]
$\mathsf{Exp}^{\text{IND-CKA-BS-}b_1,b_2}_{\mathcal{A}=\mathtt{BS},\mathtt{DS\text{-}PEKS}}(\lambda)$
$\mathtt{pp} \leftarrow \mathtt{Setup}(1^\lambda)$
$(pk_{\mathtt{BS}}, sk_{\mathtt{BS}}) \leftarrow \mathtt{KeyGen}(\mathtt{pp})$
$\mathtt{pp}, pk_{\mathtt{FS}}$
$(pk_{\mathtt{FS}}, sk_{\mathtt{FS}}) \leftarrow \mathtt{KeyGen}(\mathtt{pp})$
$\mathtt{pp}_s$
$\mathtt{pp}_s \leftarrow \{\mathtt{pp}, pk_{\mathtt{FS}}, pk_{\mathtt{BS}}, sk_{\mathtt{BS}}\}$
$W \leftarrow \mathcal{A}^{\mathcal{O}_{\mathtt{T}}(\cdot)}(\mathtt{pp}_s)$
W
$w_0, w_1, pk_{\mathtt{BS}}$
$c_{w_{b_1}} \leftarrow \mathtt{BuildIndex}(\mathtt{pp}, w_{b_1}, pk_{\mathtt{FS}}, pk_{\mathtt{BS}})$
$t_{w_{b_2}} \leftarrow \mathtt{Trapdoor}(\mathtt{pp}, w_{b_2}, pk_{\mathtt{FS}}, pk_{\mathtt{BS}})$
$\mathcal{C}$
$c_{w_{b_1}}, t_{w_{b_2}}, \mathtt{ITS}^*$
$\mathtt{ITS}^* \leftarrow \mathtt{FrontTest}(\mathtt{pp}, c_{w_{b_1}}, t_{w_{b_2}}, sk_{\mathtt{FS}})$
$b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathtt{T}}(\cdot)}(\mathtt{pp}_s, \mathcal{C})$
$\mathcal{B}$
$\mathcal{B}'$
Oracle $\mathcal{O}_{\mathtt{T}_1}(\cdot,\cdot)$ of SS-CKA
Adversary $\mathcal{A}$ (BS)
Simulator $\mathcal{S}$ $(\mathsf{Exp}^{\text{SS-CKA-}b}_{\mathcal{A}=\mathtt{BS},\mathtt{DS\text{-}PEKS}}(\lambda))$
$\mathsf{Exp}^{\text{IND-CKA-BS-}b}_{\mathcal{A}=\mathtt{BS},\mathtt{DS\text{-}PEKS}}(\lambda)$
c_w, w'
$t_{w'} \leftarrow \mathtt{Trapdoor}(\mathtt{pp}, w', pk_{\mathtt{FS}}, pk_{\mathtt{BS}})$
$c_w, t_{w'}$
if $\mathtt{ITS} = \perp$, $k \leftarrow \perp$
ITS
$\mathtt{ITS} \leftarrow \mathcal{O}_{\mathtt{T}_1}(c_w, t_{w'})$
k
else $k \leftarrow \mathtt{BackTest}(\mathtt{pp}, \mathtt{ITS}, sk_{\mathtt{BS}})$
Oracle $\mathcal{O}_{\mathtt{T}_2}(\cdot,\cdot)$ of IND-KGA
Adversary $\mathcal{A}$ (BS)
Simulator $\mathcal{S}$ $(\mathsf{Exp}^{\text{IND-KGA-}b}_{\mathcal{A}=\mathtt{BS},\mathtt{DS\text{-}PEKS}}(\lambda))$
$\mathsf{Exp}^{\text{IND-CKA-BS-}b}_{\mathcal{A}=\mathtt{BS},\mathtt{DS\text{-}PEKS}}(\lambda)$
$t_{w'}, w$
$c_w \leftarrow \mathtt{BuildIndex}(\mathtt{pp}, w, pk_{\mathtt{FS}}, pk_{\mathtt{BS}})$
$c_w, t_{w'}$
if $\mathtt{ITS} = \perp$, $k \leftarrow \perp$
ITS
$\mathtt{ITS} \leftarrow \mathcal{O}_{\mathtt{T}_1}(c_w, t_{w'})$
k
else $k \leftarrow \mathtt{BackTest}(\mathtt{pp}, \mathtt{ITS}, sk_{\mathtt{BS}})$

FIGURE 5: IND-CKA-BS secure implies all SS-CKA, IND-KGA and IND-KGA-II secure. The oracle $\mathcal{O}_T$, the challenge $\mathcal{C}$, the input W and the outputs ($\mathcal{B}$ and $\mathcal{B}'$) are defined in Definition 2.

interactions and oracle accesses between the simulator and the adversary are depicted in Fig. 6.

The challenged index $(c^{[1]}, c^{[2]}) \leftarrow c_{w_b}$ and trapdoor $(t^{[1]}, t^{[2]}) \leftarrow t_{w_b}$ are in the similar form. For both possible $w \in \{w_0, w_1\}$, there exists numbers $r, r' \in Z_q^*$ that satisfies $c^{[1]} = g^r h^w$ or $t^{[1]} = g^{r'} h^{-w}$, respectively. In other words, the information of w_d is perfectly hidden by g^r and $g^{r'}$ unless the adversary can distinguish g^r or $g^{r'}$ from ciphertext $c^{[2]}$ or $t^{[2]}$. The indistinguishability between w_0 and w_1 is determined through the distinguishability of g^r (or $g^{r'}$) from the ciphertext because X_r(or $X^{r'}$) might be encrypted as $c^{[2]}$ (or $t^{[2]}$), and $X = g^x$ where x is the private key only known by the adversary. Two scenarios are discussed below depends on the relationship between b and d.

- Case 1: $b = d$ which occurs with half probability. In the true statement, c_{w_b} and t_{w_b} are a legal index and a legal trapdoor. As aforementioned, to distinguish the keyword w_d, it is necessary to distinguish at least one plaintext from $c^{[2]}$ or $t^{[2]}$, which definitely breaks their IND-CCA security. The advanced probability of case 1 is estimated as $2Adv_{\mathcal{A},PKE}^{IND-CCA2}(\lambda)$ since the adversary can easily distinguish w_d if any one of those two ciphertext is distinguishable.
- Case 2: $b \neq d$ which occurs with half probability. Clearly, $\{g^r h^w, Enc(m_{1-d}, pk_{BS})\} \leftarrow c_{w_b}$ is an illegal DS-PEKS index since $m_{1-d} \neq X^r$. It differs in the following two conditions:
  - In case $\mathcal{A}$ terminates after detecting that the given challenge c_{w_b} is invalid, the simulator directly outputs $b = 1 - d$ to break the IND-CCA2 security. We mark this probability as $Adv_{\mathcal{A},PKE}^{IND-CCA2}(\lambda)$ since the adversary distinguishes $Enc(m_{1-d}, pk_{BS})$ from $Enc(X_r, pk_{BS})$ in an IND-CCA2 secure encryption.
  - In case $\mathcal{A}$ didn't recognize the illegal challenge c_{w_b}. The keyword w_d is perfectly hidden in c_{w_b} because no information about the randomness r is available from $c^{[2]}$. On the contrary, t_{w_b} remains a legal trapdoor so that the adversary has advanced probability $Adv_{\mathcal{A},PKE}^{IND-CCA2}(\lambda)$ to distinguish w_d from $t^{[2]}$, which makes the adversary be more willing to output a wrong guess $b' = d \neq b$. To summarize, the adversary is estimated with $-Adv_{\mathcal{A},PKE}^{IND-CCA2}(\lambda)$ advanced probability to outputs $b' = b$ because $d = 1 - b$.

As a result, the probability that the simulator breaks the IND-CCA2 security in case 2 is estimated as $1/2$. In an overall review, the probability that $\mathcal{S}$ output $b^* = b$ which denotes the correct choice of the IND-CCA2 game is calculated as following.

$$1/2 \cdot (1/2 + 2Adv_{\mathcal{A},PKE}^{IND-CCA2}(\lambda)) + 1/2 \cdot 1/2 = 1/2 + Adv_{\mathcal{A},PKE}^{IND-CCA2}(\lambda).$$

In summary, above proof tells that any polynomial-time adversary has at most $Adv_{\mathcal{A},PKE}^{IND-CCA2}(\lambda)$ advanced probability to win the IND-CKA-FS experiment. The only extra cost is twice times of decryption oracle accesses than the IND-CCA2 secure PKE scheme. By the intractability of $Adv_{\mathcal{A},PKE}^{IND-CCA2}(\lambda)$, we can say the proposed generic DS-PEKS construction is IND-CKA-FS secure.

Theorem 4: The generic DS-PEKS construction is IND-CKA-BS secure based on the intractability of the CDH assumption and the CONF assumption as well as the one-wayness of the hash function H.

Proof 4: In the beginning of experiment $Exp_{DS-PEKS,\mathcal{A}}^{IND-CKA-BS}(\lambda)$, the simulator executes PP$\leftarrow$ Setup(1^λ) and $(sk_{FS}, pk_{FS}) \leftarrow$ KeyGen(pp), where $sk_{FS} = x$ and $pk_{FS} = X = g^x$. Then, it delivers ers(pp, pk_{FS} to the adversary in order to start the experiment $Exp_{DS-PEKS,\mathcal{A}}^{IND-CKA-BS}(\lambda)$. On input an oracle query $\mathcal{O}_{FT}(c_w, t_{w'})$, the simulator returns ITS $\leftarrow$ FrontTest(pp, $c_w, t_{w'}, sk_{FS}$). When the adversary outputs keywords (w_0, w_1), the simulator picks $b_1, b_2 \leftarrow \{0, 1\}$, $r, r' \xleftarrow{\$} Z_q^*$, $R \xleftarrow{\$} G$ and computes $c_{w_{b_1}}$ $\leftarrow$BuildIndex(pp, $w_{b_1}, pk_{FS}, pk_{BS}$), $t_{w_{b_2}}$ $\leftarrow$Trapdoor(pp, $w_{b_2}, pk_{FS}, pk_{BS}$). Let $(c^{[1]}, c^{[2]}) \leftarrow c_{w_{b_1}}$ and $(t^{[1]}, t^{[2]}) \leftarrow t_{w_{b_2}}$, the challenge ITS is composed of $(I^{[1]}, I^{[2]}, I^{[3]}, I^{[4]}) \leftarrow ITS*$ where $I^{[1]} \leftarrow c^{[2]}$, $I^{[2]} \leftarrow t^{[2]}$, $I^{[3]} \leftarrow H(R)$ and $I^{[4]} \leftarrow (c^{[1]} \cdot t^{[1]})^x R$. After receiving the challenge $(c_{w_{b_1}}, t_{wb_2}, ITS*)$, the adversary is still allowed to access oracle $\mathcal{O}_{FT}(c_w, t_{w'})$ if $c_w \neq c_{wb_1}$ and $t_{w'} \neq t_{w_{b_2}}$. Finally, the adversary outputs (b_1', b_2') to terminate the experiment. The challenge includes c_{wb_1}, t_{wb_2} and $ITS*$ where

$$
\begin{aligned}
&(c^{[1]}, c^{[2]}) \leftarrow c_{w_{b_1}}, g^r h^{w_{b_1}} \leftarrow c_{[1]}, Enc(X^r, pk_{BS}) \leftarrow c_{[2]}\\
&(t^{[1]}, t^{[2]}) \leftarrow t_{w_{b_2}} g^{r'} h^{-wb_2} \leftarrow t_{[1]}, Enc(X^r, pk_{BS}) \leftarrow t_{[2]}\\
&(I^{[1]}, I^{[2]}, I^{[3]}, I^{[4]}) \leftarrow ITS*, c^{[2]} \leftarrow I^{[2]}, t^{[2]} \leftarrow I^{[2]}\\
&H(R) \leftarrow I^{[3]}, X^{r+r'} h^{x(w_{b_1} - w_{b_2})} R \leftarrow I^{[4]}
\end{aligned}
$$

In fact, $c^{[2]}$ and $t^{[2]}$ can be regarded as plaintext X^r and $X^{r'}$, respectively, because they are encrypted using the adversary's public key. The question can be simplified as:

$$
\begin{aligned}
(b_1', b_2') \leftarrow \mathcal{A}^{\mathcal{O}_{FT}(\cdot,\cdot)}(&g^r h^{w_{b_1}}, X^r, g^{r'} h^{-w_{b_2}}, X^{r'},\\
&H(R), X^{r+r'} h^{x(w_{b_1} - w_{b_2})} R)
\end{aligned}
$$

Claim 1: By the one-wayness of the hash function H and the intractability of the CDH assumption, it is hard to distinguish b_1 and b_2 from $ITS*$.

Proof 5: The randomness R perfectly covers $I^{[4]}$ unless R can be obtained from $I^{[3]} = H(R)$. By the one-wayness of the hash function, the adversary can only gradually cancel each part of $I^{[4]}$ to gain R and verify it through $I^{[3]}$. Obviously, X^r and $X^{r'}$ are known by the adversary; two possible scenarios about value $h^{x(w_{b_1} w_{b_2})}$ are discussed below:

- Case 1: $b_1 = b_2$, which happens in half probability. In this case, neither keyword w_{b_1} nor w_{b_2} exists so that there is no advanced probability to distinguish.

Adversary $\mathcal{A}$ (FS)		**Simulator** $\mathcal{S}$ $(\mathsf{Exp}^{\mathsf{IND\text{-}CKA\text{-}FS}}_{\mathcal{A}=\mathsf{FS},\mathsf{DS\text{-}PEKS}}(\lambda))$		$\mathsf{Exp}^{\mathsf{IND\text{-}CCA2\text{-}}b}_{\mathcal{A},\mathsf{PKE}}(\lambda)$
				$\mathsf{pp}_{\mathsf{cca}} \leftarrow \mathsf{Setup}_{\mathsf{cca}}(1^\lambda)$
		$\{h, H\} \leftarrow \mathsf{Setup}(1^\lambda)$	$\xleftarrow{\mathsf{pp}_{\mathsf{cca}}, pk_{\mathsf{BS}}}$	$(pk_{\mathsf{BS}}, sk_{\mathsf{BS}}) \leftarrow \mathsf{KG}_{\mathsf{cca}}(\mathsf{pp}_{\mathsf{cca}})$
	$\xleftarrow{\mathsf{pp}, pk_{\mathsf{BS}}}$	$\mathsf{pp} \leftarrow \{\mathsf{pp}_{\mathsf{cca}}, h, H\}$		
$(w_0, w_1, pk_{\mathsf{FS}}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{BT}}(\cdot)}(\mathsf{pp}, pk_{\mathsf{BS}})$	$\xrightarrow{w_0, w_1}$	$d \leftarrow_R \{0,1\}, r, r' \leftarrow_R \mathbb{Z}_q^*$		
		$m_d \leftarrow X^r, m_{1\text{-}d} \leftarrow_R \mathbb{G}$	$\xrightarrow{m_0, m_1}$	
		$c_{w_b} \leftarrow (g^r h^{w_d}, c_b)$	$\xleftarrow{c_b}$	$c_b \leftarrow \mathsf{Enc}(m_b, pk_{\mathsf{BS}})$
	$\xleftarrow{c_{w_b}, t_{w_b}}$	$t_{w_b} \leftarrow (g^{r'} h^{\text{-}w_d}, \mathsf{Enc}(X^{r'}, pk_{\mathsf{BS}}))$		
$b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{BT}}(\cdot)}(\mathsf{pp}, pk_{\mathsf{BS}}, c_{w_b}, t_{w_b})$	$\xrightarrow{b'}$		$\xrightarrow{b'}$	

Oracle $\mathcal{O}_{\mathsf{BT}}(\cdot)$ of IND-CKA-FS

Adversary $\mathcal{A}$ (FS)		**Simulator** $\mathcal{S}$ $(\mathsf{Exp}^{\mathsf{IND\text{-}CKA\text{-}FS}}_{\mathcal{A}=\mathsf{FS},\mathsf{DS\text{-}PEKS}}(\lambda))$		$\mathsf{Exp}^{\mathsf{IND\text{-}CCA2\text{-}}b}_{\mathcal{A},\mathsf{PKE}}(\lambda)$
	$\xrightarrow{\mathtt{ITS}}$	$(I^{[1]}, I^{[2]}, I^{[3]}, I^{[4]}) \leftarrow \mathtt{ITS}$	$\xrightarrow{I^{[1]}}$	
			$\xleftarrow{u}$	$u \leftarrow \mathcal{O}_D(I^{[1]})$
			$\xrightarrow{I^{[2]}}$	
		if $((u = \perp) \wedge (v = \perp))$	$\xleftarrow{v}$	$v \leftarrow \mathcal{O}_D(I^{[2]})$
		$\quad k \leftarrow \perp$		
		else if $(I^{[3]} = H(I^{[4]}/(uv)))$		
		$\quad k \leftarrow 1$		
		else		
	$\xleftarrow{k}$	$\quad k \leftarrow 0$		

FIGURE 6: The simulator $\mathcal{S}$ takes advantage of $\mathcal{A}$ (in experiment IND-CKA-FS) to break the IND-CCA2 experiment.

- Case 2: $b_1 \neq b_2$, which also happens in half probability. By the intractability of the CDH assumption, it is computational hard to acquire h^x on input (g, X, h).

In summary, the advanced probability of gaining b_1 and b_2 from $ITS*$ is estimated as $Adv^{OW}_{\mathcal{A},H}(\lambda) + 1/2Adv^{CDH}_{\mathcal{A},G}(\lambda)$.

Claim 2: By the intractability of the CONF assumption, it is hard to distinguish b_1 and b_2 from $c_{w_{b_1}}$ and $t_{w_{b_2}}$, respectively.

Proof 6: By theorem 1, the probability of information leaked from $ITS*$ is negligible so that we continue to discuss the probability of $c_{w_{b_1}}$ and $t_{w_{b_1}}$. For $c_{w_{b_1}}$, for two possible $Z_0 \leftarrow c^{[1]}/h^{w_0}$ and $Z_1 \leftarrow c^{[1]}/h^{w_1}$, it is indistinguishable based on the intractability of the CONF assumption: determine whether $Z_{b_1} = g^r$ or not on input $(g, g^x, g^{xr}, Z_{b_1})$. The advanced probability of distinguish b_1 from $c_{w_{b_1}}$ is estimated as $Adv^{CONF}_{\mathcal{A},G}(\lambda)$. The similar condition above occurs in $t_{w_{b_2}}$ with $Z_0 \leftarrow t^{[1]}/h^{w_0}$ and $Z_1 \leftarrow t^{[1]}/h^{w_1}$ so that the advanced probability on seeing $t_{w_{b_2}}$ is estimated as $Adv^{CONF}_{\mathcal{A},G}(\lambda)$ as well.

By and large, the overall advanced probability of a correct output $((b_1 = b_1') \wedge (b_2 = b_2'))$ is estimated less than $Adv^{OW}_{\mathcal{A},H}(\lambda) + 1/2Adv^{CDH}_{\mathcal{A},G}(\lambda) + +2Adv^{CONF}_{\mathcal{A},G}(\lambda)$. The IND-CKA-BS security obliviously holds owing to the aforementioned negligible advanced probability.

V. CONCRETE INSTANTIATION AND COMPARISON

In this section we first give a concrete instantiation by adopting Cramer and Shoup's IND-CCA2 secure public key encryption scheme. Then, we compare our instantiation with Chen et al.'s DS-PEKS [19].

A. CONCRETE INSTANTIATION

The following we describe how to obtain an instantiation using Cramer and Shoup's encryption scheme [24].

- Setup(1^λ): This algorithm runs as follows.
  - Choose a cyclic group G of prime order q, with two generators g_1, g_2, and select $h \leftarrow G$.
  - Choose two secure hash functions $H_1 : G \rightarrow \{0,1\}^\lambda$, $H_2 : \{0,1\}^* \times G^2 \rightarrow G$.
  - Output a public parameter pp $\leftarrow (H_1, H_2, g_1, g_2, G, h)$.
- KeyGen(pp): This algorithm runs as follows.
  - Choose $x \xleftarrow{\$} Z_q^*$, and compute $X = g^x$.
  - Choose $(s, a, b, a', b') \xleftarrow{\$} Z_q^*$
  - Compute $h = g_1^s, c = g_1^a g_2^b, d = g_1^{a'} g_2^{b'}$.
  - Output front server and back server's key pair $sk_{FS} = x, pk_{FS} = X, pk_{BS} = (h, c, d), sk_{BS} = (s, a, b, a', b')$.
- BuildIndex(pp, w, pk_{FS}, pk_{BS}): This algorithm runs as follows.

- Choose $r \overset{\$}{\leftarrow} Z_q^*$.
- Compute $u_1 = g_1^r, u_2 = g_2^r, e = h^r w, v = (cd^{H_2(u_1,u_2,e)})^r$.
- Output searchable ciphertext $c_w = (g^r h^w, (u_1, u_2, e, v))$

- Trapdoor(pp, w', pk_{FS}, pk_{BS}: This algorithm runs as follows.
  - Choose $r' \leftarrow Z_q^*$
  - Compute $u_1' = g_1^{r'}, u_2' = g_2^{r'}, e' = h^{r'} w, v' = (cd^{H_2(u_1',u_2',e')})^{r'}$
  - Output a trapdoor $t_{w'} = (g^{r'} h^{-w'}, (u_1', u_2', e', v'))$.
- FrontTest(pp, $c_w, t_{w'}, sk_{FS}$): This algorithm runs as follows.
  - Parse $c_w = (c^{[1]} = g^r h^w, c^{[2]} = (u_1, u_2, e, v))$.
  - Parse $t_{w'} = (t^{[1]} = g^{r'} h^{-w'}, t^{[2]} = (u_1', u_2', e', v'))$.
  - Pick $R \overset{\$}{\leftarrow} G$.
  - Output ITS $\leftarrow (c^{[2]}, t^{[2]}, H_1(R), (c^{[1]} \cdot t^{[1]})^x R$.

  BackTest(pp, ITS, sk_{BS}): This algorithm runs as follows.
  - Parse ITS $= (I^{[1]} = c^{[2]} = (u_1, u_2, e, v), I^{[2]} = t^{[2]} = (u_1', u_2', e', v'), I^{[3]} = H_1(R), I^{[4]} = (c^{[1]} \cdot t^{[1]})^x R$.
  - Check whether $v = u_1^{a+\zeta a'} \cdot u_2^{b+\zeta b'}$, for $\zeta = H_1(u_1, u_2, e)$.
  - Compute $\sigma = e/u_1^s$.
  - Check whether $v' = {u_1'}^{a+\zeta' a'} \cdot {u_2'}^{b+\zeta' b'}$, for $\zeta' = H_1(u_1', u_2', e')$.
  - Compute $\sigma' = e'/{u_1'}^s$.
  - If $I[3] = H_1(I^{[4]}/(\sigma \cdot \sigma'))$, then output 1. Otherwise, output 0.

B. COMPARISON

Let $|G|$ denote the bit size of an element in group G. Let T_{Exp}, T_H, and T_{Mul} represent the modular exponentiation over point operation, multiplication over point operation, and hash-to-point operation, respectively. Table 1 shows the compared result of our scheme and Chen et al.'s scheme [19] terms of the computation cost, communication cost, and the security properties. For the computation, we only consider three time-consuming operations, i.e., T_{Exp}, T_H, and T_{Mul}. The result shows that our instantiation has more security properties in terms of SS-CKA and IND-KGA in compared with Chen et al's scheme. Unfortunately, to support more security, our instantiation needs more time to generate ciphertext, trapdoor and testing and the size of ciphertext and trapdoor of our instantiation are large than Chen et al.'s scheme.

Next, to evaluate the efficiency of our scheme and Chen et al's work, we experiments the time-consuming operations on the desktop, where the detail of system environment is listed is Table 2. The data are obtained (Table 3) by using pairing-based cryptography library (PBC)[1] with Type-A pairing

[1]https://crypto.stanford.edu/pbc/

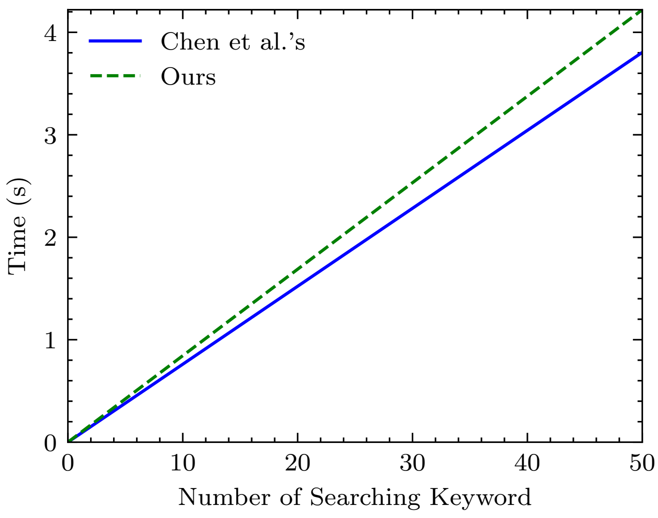


FIGURE 7: Compared the computation cost of ciphertext generation with Chen et al's scheme [19].

where group order and group element are 160-bit and 2048-bit, respectively. The results of computation costs, including detailed ciphertext generation, trapdoor generation, and testing, are shown in Fig. 7, Fig. 8, and Fig 9, respectively. According to the results in Fig. 7 and Fig. 8, although our instantiation requires more operations, the actual operation time varies little. When the number of searching keywords is 50, our instantiation and Chen et al's scheme takes about 4.22035 seconds and 3.80395 seconds respectively to process ciphertext generation and trapdoor generation. Additionally, according to the results in Fig. 9, when the number of searching keywords is 50, our instantiation and Chen et al's scheme takes about 1.46015 seconds and 1.45685 seconds respectively to process testing. Therefore, the results show that with a little more computing time, our instntiation can achieve more robust securities.

VI. CONCLUSION

The inner keyword guessing attacks threat the semantic security of most traditional PEKS schemes. For this issue, Chen et al. proposed the first DS-PEKS scheme as a solution which separates the testing server into two servers to resist the threat. However, their security models can be significantly enhanced which are discussed in Section III; meanwhile, their works are found not as secure as they claimed. Following, we propose new security models and new general constructions for DS-PEKS schemes. For the security notions, the indistinguishability against chosen keyword attacks models, IND-CKA-FS and IND-CKA-BS, are proposed to regulate the security against malicious front servers and back servers, respectively. Moreover, the newly proposed two security models imply all five previous models, which are even sounder and stronger. For the general construction, we take IND-CCA2 secure PKE schemes as building blocks to construct a IND-CKA-FS secure and IND-CKA-BS secure

TABLE 1: Comparisons between Chen et al.s' work [19] and our scheme.

Schemes	Computation			Size (bits)		Security	
	Ciphertext Gen.	Trapdoor Gen.	Testing	Ciphertext	Trapdoor	SS-CKA	IND-KGA
Chen el al.'s	$4T_{Exp} + T_H + 2T_{Mul}$	$4T_{Exp} + T_H + 2T_{Mul}$	$7T_{Exp} + 3T_{Mul}$	$3\|G\|$	$3\|G\|$	×	×
Ours	$6T_{Exp} + T_H + 3T_{Mul}$	$6T_{Exp} + T_H + 3T_{Mul}$	$7T_{Exp} + 6T_{Mul}$	$5\|G\|$	$5\|G\|$	✓	✓

TABLE 2: Experimentation platform information.

Description	Data
CPU	AMD Ryzen 5-2600 3.4GHz
CPU processor number	6
Operation system	Ubuntu 18.04
Linux kernel version	5.3.0-59-generic
Random access memory	16.3GB
Solid state disk	232.9GB

TABLE 3: Notations of operations and their running time.

Notations	Operations	Running time (ms)
T_H	Hash-to-point	59.423115
T_{Mul}	Multiplication over Point	0.022
T_{Exp}	Modular exponentiation over Point	4.153842

DS-PEKS schemes. The formal security proof is discussed in standard model in Section IV-A.

APPENDIX

Flaws of Chen et al.'s security

We briefly recall Chen et al.'s construction here and then introduce its flaw which makes them fail to be proved SS-CKA security and IND-KGA security. Because the newly proposed IND-CKA-FS security implies SS-CKA security and IND-KGA security for malicious front servers, their work is definitely not IND-CKA-FS security. Chen et al.'s works fail to be proved IND-CKA-BS security as well because of the same reasons. In brief, they adopted a variant linear and homomorphic smooth projective hash function (LH-SPHF) to build a DS-PEKS scheme, which is summarized as follows:

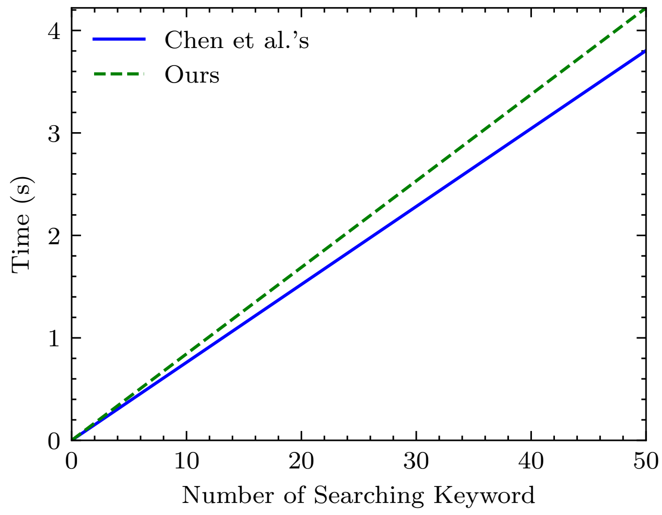


FIGURE 8: Compared the computation cost of trapdoor generation with Chen et al's scheme [19].

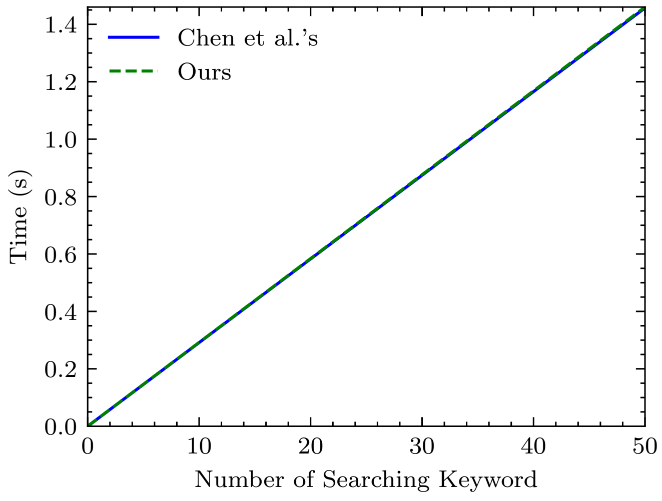


FIGURE 9: Compared the computation cost of testing with Chen et al's scheme [19].

Let H be a cryptographic hash function and $\bigotimes$ is an operator among some group elements; hk denotes the hash key (like a secret key) and hp stands for the hash projective key (like a public key) in the SPHF system. In the SPHF computation, for all randomness W (in some language L) and its witness wt, $ProjHash(hp, W, wt) = Hash(hk, W)$.

The sender picks a randomness W_1 with its witness wt_1 to build an index (W_1, C_1) for his selected keywords w using the public key of the front server (hp_{FS}) and the back server (hp_{BS}), respectively.

$$x_1 \leftarrow ProjHash(hp_{FS}, W_1, wt_1),$$
$$y_1 \leftarrow ProjHash(hp_{BS}, W_1, wt_1),$$
$$C_1 \leftarrow H(w) \bigotimes x_1 \bigotimes y_1$$

The trapdoor is in a similar form with the index. A randomness W_2 is chosen first with its witness wt_2, then, the trapdoor (W_2, C_2) of keyword w' is computed by

$$x_2 \leftarrow ProjHash(hp_{FS}, W_2, wt_2),$$
$$y_2 \leftarrow ProjHash(hp_{BS}, W_2, wt_2),$$
$$C_2 \leftarrow H(w')^{-1} \bigotimes x_2 \bigotimes y_2$$

The front test relies on the linear property and homomorphism of the LH-SPHF.

1) It firstly computes $W \leftarrow W_1 \bigodot W_2$ for some homomorphic operator $\bigodot$ between randomnesses where $Hash(hk, W_1 \bigodot W_2) = Hash(hk, W_1) \bigotimes Hash(hk, W_2)$.

2) Secondly, it computes $C \leftarrow C_1 \bigotimes C_2 \bigotimes (Hash(hk_{FS}, W))^{-1}$.
3) Thirdly, it picks another witness Δw and computes ITS $\leftarrow$ $(W*, C*)$ where $W* \leftarrow \Delta w \circ W$ and $C* \leftarrow \Delta w \bullet C$. The linear operators $\circ$ and $\bullet$ are defined for all randomnesses and their witnesses, where $Hash(hk, \Delta w \circ W) = \Delta w \bullet Hash(hk, W)$.

The back test outputs 1 if $C* = Hash(hk_{BS}, W*)$.

Flaws. We claim that Chen et al.'s schemes are neither SS-CKA security nor IND-KGA security, which implies they are neither IND-CKA-FS security nor IND-CKA-BS security. Take the SS-CKA security against malicious front servers for example, the attack even does not rely on the knowledge of the back server's secret key. In the SS-CKA experiment, after the adversary outputs (w_0, w_1) and receives a challenge index (W_1, C_1)=BuildIndex(w_b, hp_{FS}, hp_{BS}), $b \in \{0, 1\}$, the adversary randomly picks a witness $\hat{w}$ and computes $\hat{W} \leftarrow \hat{w} \circ W_1$ and $\hat{C} \leftarrow \hat{w} \bullet C_1$. Then, b can be easily obtained by sending a query $((\hat{W}, \hat{C}), w_1)$ to oracle $\mathcal{O}_{T_1}$. The same problem also happens with the SS-CKA security against malicious back servers, and the IND-KGA security against malicious front servers and back servers. The linear property of LH-SPHF eliminates the its adaptively secure. The same problem will not occur in our new proposed generic construction because any tampered index or trapdoor will be detected and rejected by the simulator.

REFERENCES

[1] Chien-Ming Chen, King-Hang Wang, Kuo-Hui Yeh, Bin Xiang, Tsu-Yang Wu, Attacks and solutions on a three-party password-based authenticated key exchange protocol for wireless communications, Journal of Ambient Intelligence and Humanized Computing, Vol. 10, Issue 8, pp. 3133-3142, 2019.

[2] Chien-Ming Chen, Bin Xiang, Yining Liu, King-Hang Wang, "A Secure Authentication Protocol for Internet of Vehicles", IEEE ACCESS, vol. 7, Issue 1, pp. 12047-12057, 2019.

[3] C. Gentry, Fully homomorphic encryption using ideal lattices, in: STOC'09, ACM, 2009, pp. 169–178. doi:10.1145/1536414.1536440.

[4] D. Boneh, A. Raghunathan, G. Segev, Function-private subspace-membership encryption and its applications, in: ASIACRYPT'13, Vol. 8269 of LNCS, Springer, 2013, pp. 255–275. doi:10.1007/978-3-642-42033-7.

[5] D. Boneh, G. D. Crescenzo, R. Ostrovsky, G. Persiano, Public key encryption with keyword search, in: EUROCRYPT'04, Vol. 3027 of LNCS, Springer, 2004, pp. 50–522.

[6] D. Boneh, B. Waters, Conjunctive, subset, and range queries on encrypted data, in: TCC'07, Vol. 4392 of LNCS, Springer, 2007, pp. 535– 554.

[7] P. Wang, H. Wang, J. Pieprzyk, Keyword field-free conjunctive keyword searches on encrypted data and extension for dynamic groups, in: CANS'08, Vol. 5339 of LNCS, Springer, 2008, pp. 178–195.

[8] P. Wang, H. Wang, J. Pieprzyk, Common secure index for conjunctive keyword-based retrieval over encrypted data, in: Secure Data Management,SDM'07, Vol. 4721 of LNCS, Springer, 2007, pp. 108–123.

[9] . A. Arriaga, Q. Tang, P. Ryan, Trapdoor privacy in asymmetric searchable encryption schemes, in: AFRICACRYPT'14, Vol. 8469 of LNCS, Springer, 2014, pp. 31–50. doi:10.1007/978-3-319-06734-6.

[10] J. Baek, R. Safavi-Naini, W. Susilo, Public key encryption with keyword search revisited, in: Computational Science and Its Applications - ICCSA'08, Vol. 5072 of LNCS, Springer, 2008, pp. 1249–1259.

[11] C. B"osch, P. H. Hartel, W. Jonker, A. Peter, A survey of provably secure searchable encryption, ACM Comput. Surv. 47 (2) (2014) 18:1–18:51.

[12] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, H. Shi, Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions, J. Cryptology 21 (3) (2008) 350–391.

[13] J. W. Byun, H. S. Rhee, H. Park, D. H. Lee, Off-line keyword guessing attacks on recent keyword search schemes over encrypted data, in: Secure Data Management, SDM'06, Vol. 4165 of LNCS, Springer, 2006, pp. 75–83.

[14] I. R. Jeong, J. O. Kwon, D. Hong, D. H. Lee, Constructing PEKS schemes secure against keyword guessing attacks is possible?, Computer Communications 32 (2) (2009) 394–396.

[15] P. Xu, H. Jin, Q. Wu, W. Wang, Public-key encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack, IEEE Trans. Computers 62 (11) (2013) 2266–2277.

[16] Y.-C. Chen, Speks: Secure server-designation public key encryption with keyword search against keyword guessing attacks, The Computer Journal.

[17] Tsu-Yang Wu, Chien-Ming Chen, King-Hang Wang, Chao Meng, Eric Ke Wang, A Provably Secure Certificateless Public Key Encryption with Keyword Search, Journal of the Chinese Institute of Engineers, vol. 42, no. 1, pp. 20-28, 2019.

[18] R. Chen, Y. Mu, G. Yang, F. Guo, X. Wang, A new general framework for secure public key encryption with keyword search, in: E. Foo, D. Stebila (Eds.), ACISP'15, Vol. 9144 of LNCS, Springer, 2015, pp. 59–76. doi:10.1007/978-3-319-19962-7.

[19] R. Chen, Y. Mu, G. Yang, F. Guo, X. Wang, Dual-server public-key encryption with keyword search for secure cloud storage, IEEE Trans. Information Forensics and Security 11 (4) (2016) 789–798.

[20] Q. Huang, H. Li, An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks, Information Sciences.

[21] K. Huang, R. Tso, Provable secure dual-server public key encryption with keyword search, in: IVSW 2017, 2017, pp. 39–44.

[22] E. Ryu, T. Takagi, Efficient conjunctive keyword-searchable encryption, in: Advanced Information Networking and Applications AINA'07, IEEE Computer Society, 2007, pp. 409–414.

[23] K. Sakurai, H. Shizuya, Relationships among the computational powers of breaking discrete log cryptosystems, in: EUROCRYPT'95, Vol. 921 of LNCS, Springer, 1995, pp. 341–355.

[24] R. Cramer, V. Shoup, A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack, in: CRYPTO'98, Vol. 1462 of LNCS, Springer, 1998, pp. 13–25.

[25] M. Abdalla, F. Benhamouda, D. Pointcheval, Public-key encryption indistinguishable under plaintext-checkable attacks, in: PKC'15, Vol. 9020 of LNCS, Springer, 2015, pp. 332–352. doi:10.1007/978-3-662-46447-2.

[26] S. Ma, Y. Mu, W. Susilo, B. Yang, Witness-based searchable encryption, in: Information Sciences, vol. 453, pp. 364–378, 2018.

[27] Z.-Y. Liu, Y.-F. Tseng, R. Tso, M. Mambo, Designated-ciphertext Searchable Encryption, in Cryptology ePrint Archive, Report 2019/1418, 2019.