# Generating Searchable Public-Key Ciphertexts with Hidden Structures for Fast Keyword Search

Peng Xu, Member, IEEE, Qianhong Wu, Member, IEEE, Wei Wang, Member, IEEE,

Willy Susilo, Senior Member, IEEE, Josep Domingo-Ferrer, Fellow, IEEE, Hai Jin, Senior Member, IEEE

Abstract-Existing semantically secure public-key searchable encryption schemes take search time linear with the total number of the ciphertexts. This makes retrieval from large-scale databases prohibitive. To alleviate this problem, this paper proposes Searchable Public-Key Ciphertexts with Hidden Structures (SPCHS) for keyword search as fast as possible without sacrificing semantic security of the encrypted keywords. In SPCHS, all keyword-searchable ciphertexts are structured by hidden relations, and with the search trapdoor corresponding to a keyword, the minimum information of the relations is disclosed to a search algorithm as the guidance to find all matching ciphertexts efficiently. We construct a simple SPCHS scheme from scratch in which the ciphertexts have a hidden star-like structure. We prove our scheme to be semantically secure based on the decisional bilinear Diffie-Hellman assumption in the Random Oracle (RO) model. The search complexity of our scheme is dependent on the actual number of the ciphertexts containing the queried keyword, rather than the number of all ciphertexts. Finally, we present a generic SPCHS construction from anonymous identity-based encryption and collision-free full-identity malleable Identity-Based Key Encapsulation Mechanism (IBKEM) with anonymity. We illustrate two collision-free full-identity malleable IBKEM instances, which are semantically secure and anonymous, respectively, in the RO and standard models. The latter instance enables us to construct an SPCHS scheme with semantic security in the standard model.

*Index Terms*—Public-key searchable encryption, semantic security, identity-based key encapsulation mechanism, identity based encryption

#### I. INTRODUCTION

**P**UBLIC-KEY encryption with keyword search (PEKS), introduced by Boneh *et al.* in [1], has the advantage that anyone who knows the receiver's public key can upload keyword-searchable ciphertexts to a server. The receiver can delegate the keyword search to the server. More specifically,

P. Xu and H. Jin are with Services Computing Technology and System Lab, Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China. E-Mail: {xupeng, hjin}@mail.hust.edu.cn.

Q. Wu is with the School of Electronics and Information Engineering, Beihang Universisity, Beijing, China, and with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China. E-mail: qhwu@xidian.edu.cn.

W. Wang is with Cyber-Physical-Social Systems Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China. E-Mail: viviawangww@gmail.com.

W. Susilo is with Centre for Computer and Information Security Research, School of Computer Science and Software Engineering, University of Wollongong, Australia. E-Mail: wsusilo@uow.edu.au.

J. Domingo-Ferrer is with Universitat Rovira i Virgili, Department of Computer Engineering and Mathematics, UNESCO Chair in Data Privacy, 43007, Tarragona, Catalonia. E-Mail: josep.domingo@urv.cat. each sender separately encrypts a file and its extracted keywords and sends the resulting ciphertexts to a server; when the receiver wants to retrieve the files containing a specific keyword, he delegates a keyword search trapdoor to the server; the server finds the encrypted files containing the queried keyword without knowing the original files or the keyword itself, and returns the corresponding encrypted files to the receiver; finally, the receiver decrypts these encrypted files<sup>1</sup>. The authors of PEKS [1] also presented semantic security against chosen keyword attacks (SS-CKA) in the sense that the server cannot distinguish the ciphertexts of the keywords of its choice before observing the corresponding keyword search trapdoors. It seems an appropriate security notion, especially if the keyword space has no high min-entropy. Existing semantically secure PEKS schemes take search time linear with the total number of all ciphertexts. This makes retrieval from large-scale databases prohibitive. Therefore, more efficient searchable public-key encryption is crucial for practically deploying PEKS schemes.

1

One of the prominent works to accelerate the search over encrypted keywords in the public-key setting is deterministic encryption introduced by Bellare et al. in [2]. An encryption scheme is deterministic if the encryption algorithm is deterministic. Bellare et al. [2] focus on enabling search over encrypted keywords to be as efficient as the search for unencrypted keywords, such that a ciphertext containing a given keyword can be retrieved in time complexity logarithmic in the total number of all ciphertexts. This is reasonable because the encrypted keywords can form a tree-like structure when stored according to their binary values. However, deterministic encryption has two inherent limitations. First, keyword privacy can be guaranteed only for keywords that are a priori hardto-guess by the adversary (i.e., keywords with high minentropy to the adversary); second, certain information of a message leaks inevitably via the ciphertext of the keywords since the encryption is deterministic. Hence, deterministic encryption is only applicable in special scenarios.

# A. Our Motivation and Basic Ideas

We are interested in providing highly efficient search performance without sacrificing semantic security in PEKS.

<sup>&</sup>lt;sup>1</sup>Since the encryption of the original files can be separately processed with an independent public-key encryption scheme as in [1], we only describe the encryption of the keywords (unless otherwise clearly stated in the paper).



Figure 1: Hidden star-like structure formed by keyword searchable ciphertexts. (The dashed arrows denote the hidden relations.  $Enc(W_i)$  denotes the searchable ciphertext of keyword  $W_{i.}$ )

Observe that a keyword space is usually of no high minentropy in many scenarios. Semantic security is crucial to guarantee keyword privacy in such applications. Thus the linear search complexity of existing schemes is the major obstacle to their adoption. Unfortunately, the linear complexity seems to be inevitable because the server has to scan and test each ciphertext, due to the fact that these ciphertexts (corresponding to the same keyword or not) are indistinguishable to the server.

A closer look shows that there is still space to improve search performance in PEKS without sacrificing semantic security if one can organize the ciphertexts with elegantly designed but hidden relations. Intuitively, if the keywordsearchable ciphertexts have a hidden star-like structure, as shown in Figure 1, then search over ciphertexts containing a specific keywords may be accelerated. Specifically, suppose all ciphertexts of the same keyword form a chain by the correlated hidden relations, and also a hidden relation exists from a public *Head* to the first ciphertext of each chain. With a keyword search trapdoor and the Head, the server seeks out the first matching ciphertext via the corresponding relation from the Head. Then another relation can be disclosed via the found ciphertext and guides the searcher to seek out the next matching ciphertext. By carrying on in this way, all matching ciphertexts can be found. Clearly, the search time depends on the actual number of the ciphertexts containing the queried keyword, rather than on the total number of all ciphertexts.

To guarantee appropriate security, the hidden star-like structure should preserve the semantic security of keywords, which indicates that partial relations are disclosed only when the corresponding keyword search trapdoor is known. Each sender should be able to generate the keyword-searchable ciphertexts with the hidden star-like structure by the receiver's public-key; the server having a keyword search trapdoor should be able to disclose partial relations, which is related to all matching ciphertexts. Semantic security is preserved 1) if no keyword search trapdoor is known, all ciphertexts are indistinguishable, and no information is leaked about the structure, and 2) given a keyword search trapdoor, only the corresponding relations can be disclosed, and the matching ciphertexts leak no information about the rest of ciphertexts, except the fact that the rest do not contain the queried keyword.

# B. Our Work

We start by formally defining the concept of Searchable Public-key Ciphertexts with Hidden Structures (SPCHS) and its semantic security. In this new concept, keywordsearchable ciphertexts with their hidden structures can be generated in the public key setting; with a keyword search trapdoor, partial relations can be disclosed to guide the discovery of all matching ciphertexts. Semantic security is defined for both the keywords and the hidden structures. It is worth noting that this new concept and its semantic security are suitable for keyword-searchable ciphertexts with any kind of hidden structures. In contrast, the concept of traditional PEKS does not contain any hidden structure among the PEKS ciphertexts; correspondingly, its semantic security is only defined for the keywords.

Following the SPCHS definition, we construct a simple SPCHS from scratch in the random oracle (RO) model. The scheme generates keyword-searchable ciphertexts with a hidden star-like structure. The search performance mainly depends on the actual number of the ciphertexts containing the queried keyword. For security, the scheme is proven semantically secure based on the Decisional Bilinear Diffie-Hellman (DBDH) assumption [3] in the RO model.

We are also interested in providing a generic SPCHS construction to generate keyword-searchable ciphertexts with a hidden star-like structure. Our generic SPCHS is inspired by several interesting observations on Identity-Based Key Encapsulation Mechanism (IBKEM). In IBKEM, a sender encapsulates a key K to an intended receiver ID. Of course, receiver ID can decapsulate and obtain K, and the sender knows that receiver ID will obtain K. However, a non-intended receiver ID' may also try to decapsulate and obtain K'. We observe that, (1) it is usually the case that K and K' are independent of each other from the view of the receivers, and (2) in some IBKEM the sender may also know K' obtained by receiver ID'. We refer to the former property as *collision freeness* and to the latter as *full-identity* malleability. An IBKEM scheme is said to be collision-free *full-identity malleable* if it possesses both properties.

We build a generic SPCHS construction with Identity-Based Encryption (IBE) and collision-free full-identity malleable IBKEM. The resulting SPCHS can generate keyword-searchable ciphertexts with a hidden star-like structure. Moreover, if both the underlying IBKEM and IBE have semantic security and anonymity (*i.e.* the privacy of receivers' identities), the resulting SPCHS is semantically secure. As there are known IBE schemes [4], [5], [6], [7] in both the RO model and the standard model, an SPCHS construction is reduced to collision-free full-identity malleable IBKEM with anonymity. In 2013, Abdalla *et al.*  proposed several IBKEM schemes to construct Verifiable Random Functions<sup>2</sup> (VRF) [8]. We show that one of these IBKEM schemes is anonymous and collision-free fullidentity malleable in the RO model. In [9], Freire *et al.* utilized the "approximation" of multilinear maps [10] to construct a standard-model version of Boneh-and-Franklin (BF) IBE scheme [11]. We transform this IBE scheme into a collision-free full-identity malleable IBKEM scheme with semantic security and anonymity in the standard model. Hence, this new IBKEM scheme allows us to build SPCHS schemes secure in the standard model with the same search performance as the previous SPCHS construction from scratch in the RO model.

# C. Other Applications of Collision-Free Full-Identity Malleable IBKEM

We note that collision-free full-identity malleable IBKEM is of independent interest. In addition to being a building block for the generic SPCHS construction, it may also find other applications, as outlined in the sequel.

Batch identity-based key distribution. A direct application of collision-free full-identity malleable IBKEM is to achieve batch identity-based key distribution. In such an application, a sender would like to distribute different secret session keys to multiple receivers so that each receiver can only know the session key to himself/herself. With collision-free full-identity malleable IBKEM, a sender just needs to broadcast an IBKEM encapsulation in the identitybased cryptography setting, e.g., encapsulating a session key K to a single user ID. According to the collisionfreeness of IBKEM, each receiver ID' can decapsulate and obtain a different key K' with his/her secret key in the identity based crypto-system. Due to the full-identity malleability, the sender knows the decapsulated keys of all the receivers. In this way, the sender efficiently shares different session keys with different receivers, at the cost of only a single encapsulation and one pass of communication.

Anonymous identity-based broadcast encryption. A slightly more complicated application is anonymous identity-based broadcast encryption with efficient decryption. An analogous application was proposed respectively by Barth et al. [12] and Libert et al. [13] in the traditional public-key setting. With collision-free fullidentity malleable IBKEM, a sender generates an identitybased broadcast ciphertext  $\langle C_1, C_2, (K_1^1 || SE(K_2^1, F_1)), ...,$  $(K_1^N || SE(K_2^N, F_N)))$ , where  $C_1$  and  $C_2$  are two IBKEM encapsulations,  $K_1^i$  is the encapsulated key in  $C_1$  for receiver  $ID_i, K_2^i$  is the encapsulated key in  $C_2$  for receiver  $ID_i$ , and  $SE(K_2^i, F_i)$  is the symmetric-key encryption of file  $F_i$  using the encapsulated key  $K_2^i$ . In this ciphertext, the encapsulated key  $K_1^i$  is not used to encrypt anything. Indeed, it is an index to secretly inform receiver  $ID_i$  on which part of this ciphertext belongs to him. To decrypt the encrypted file  $F_i$ , receiver  $ID_i$  decapsulates and obtains  $K_1^i$ from  $C_1$ , finds out  $K_1^i || SE(K_2^i, F_i)$  by matching  $K_1^i$ , and finally extracts  $F_i$  with the decapsulated key  $K_2^i$  from  $C_2$ . It can be seen that the application will work if the IBKEM is collision-free full-identity malleable. It preserves the anonymity of receivers if the IBKEM is anonymous. Note that trivial anonymous broadcast encryption suffers decryption cost linear with the number of the receivers. In contrast, our anonymous identity-based broadcast encryption enjoys constant decryption cost, plus logarithmic complexity to search the matching index in a set  $(K_1^1, ..., K_1^N)$  organized by a certain partial order, e.g., a dictionary order according to their binary representations.

## D. Related Work

Search on encrypted data has been extensively investigated in recent years. From a cryptographic perspective, the existing works fall into two categories, *i.e.*, symmetric searchable encryption [14] and public-key searchable encryption.

Symmetric searchable encryption is occasionally referred to as symmetric-key encryption with keyword search (SEKS). This primitive was introduced by Song et al. in [15]. Their instantiated scheme takes search time linear with the size of the database. A number of efforts [16], [17], [18], [19], [20] follow this research line and refine Song et al.'s original work. The SEKS scheme due to Curtmola et al. [14] has been proven to be semantically secure against an adaptive adversary. It allows the search to be processed in logarithmic time, although the keyword search trapdoor has length linear with the size of the database. In addition to the above efforts devoted to either provable security or better search performance, attention has recently been paid to achieving versatile SEKS schemes as follows. The works in [14], [21] extend SEKS to a multi-sender scenario. The work in [22] realizes fuzzy keyword search in the SEKS setting. The work in [23] shows practical applications of SEKS and employs it to realize secure and searchable audit logs. Chase et al. [24] proposed to encrypt structured data and a secure method to search these data. To support the dynamic update of the encrypted data, Kamara et al. proposed the dynamic searchable symmetric encryption in [25] and further enhanced its security in [26] at the cost of large index. The very recent work [27] due to Cash et al. simultaneously achieves strong security and high efficiency.

Following the seminal work on PEKS, Abdalla *et al.* [28] fills some gaps w.r.t. consistency for PEKS and deals with the transformations among primitives related to PEKS. Some efforts have also been devoted to make PEKS versatile. The work of this kind includes conjunctive search [29], [30], [31], [32], [33], [34], range search [35], [36], [37], subset search [37], time-scope search [28], [38], similarity search [39], authorized search [49], [50], equality test between heterogeneous ciphertexts [51], and fuzzy keyword search [52]. In addition, Arriaga et al. [53] proposed a PEKS scheme to keep the privacy of keyword search trapdoors.

In the above PEKS schemes, the search complexity takes time linear with the number of all ciphertexts. In [24],

 $<sup>^{2}</sup>$ VRF behaves like a pseudo-random function but one can verify that the output was pseudo-random.

an oblivious generation of keyword search trapdoor is to maintain the privacy of the keyword against a curious trapdoor generation. A chain-like structure is described to speed up the search on encrypted keywords. One may note that the chain in [40] cannot be fully hidden to the server and leaks the frequency of the keywords (see Supplemental Materials A for details). To realize an efficient keyword search, Bellare et al. [2] introduced deterministic publickey encryption (PKE) and formalized a security notion "as strong as possible" (stronger than onewayness but weaker than semantic security). A deterministic searchable encryption scheme allows efficient keyword search as if the keywords were not encrypted. Bellare *et al.* [2] also presented a deterministic PKE scheme (i.e., RSA-DOAEP) and a generic transformation from a randomized PKE to a deterministic PKE in the random oracle model. Subsequently, deterministic PKE schemes secure in the standard model were independently proposed by Bellare et al. [41] and Boldyreva et al. [42]. The former uses general complexity assumptions and the construction is generic, while the latter exploits concrete complexity assumptions and has better efficiency. Brakerski et al. [43] proposed the deterministic PKE schemes with better security, although these schemes are still not semantically secure. So far, deterministic PEKS schemes can guarantee semantic security only if the keyword space has a high min-entropy. Otherwise, an adversary can extract the encrypted keyword by a simple encrypt-and-test attack. Hence, deterministic PEKS schemes are applicable to applications where the keyword space is of a high min-entropy.

# E. Organization of this article

The remaining sections are as follows. Section II defines SPCHS and its semantic security. A simple SPCHS scheme is constructed in Section III. A general construction of SPCHS is given in Section IV. Two collision-free fullidentity malleable IBKEM schemes, respectively in the RO and standard models, are introduced in Section V. Section VI concludes this paper.

#### **II. MODELING SPCHS**

We first explain intuitions behind SPCHS. We describe a hidden structure formed by ciphertexts as ( $\mathbb{C}$ , **Pri**, **Pub**), where  $\mathbb{C}$  denotes the set of all ciphertexts, **Pri** denotes the hidden relations among  $\mathbb{C}$ , and **Pub** denotes the public parts. In case there is more than one hidden structure formed by ciphertexts, the description of multiple hidden structures formed by ciphertexts can be ( $\mathbb{C}$ , (**Pri**\_1, **Pub**\_1), ..., (**Pri**\_N, **Pub**\_N)), where  $N \in \mathbb{N}$ . Moreover, given ( $\mathbb{C}$ , **Pub**\_1, ..., **Pub**\_N) and (**Pri**\_1, ..., **Pri**\_N) except (**Pri**\_i, **Pri**\_j) (where  $i \neq j$ ), one can neither learn anything about (**Pri**\_i, **Pri**\_j) nor decide whether a ciphertext is associated with **Pub**\_i or **Pub**\_j.

In SPCHS, the encryption algorithm has two functionalities. One is to encrypt a keyword, and the other is to generate a hidden relation, which can associate the generated ciphertext to the hidden structure. Let (**Pri**, **Pub**) be the hidden structure. The encryption algorithm must take **Pri** as input, otherwise the hidden relation cannot be generated since **Pub** does not contain anything about the hidden relations. At the end of the encryption procedure, the **Pri** should be updated since a hidden relation is newly generated (but the specific method to update **Pri** relies on the specific instance of SPCHS). In addition, SPCHS needs an algorithm to initialize (**Pri**, **Pub**) by taking the master public key as input, and this algorithm will be run before the first time to generate a ciphertext. With a keyword search trapdoor, the search algorithm of SPCHS can disclose partial relations to guide the discovery of the ciphertexts containing the queried keyword with the hidden structure.

Definition 1 (SPCHS). SPCHS consists of five algorithms:

- SystemSetup(1<sup>k</sup>, W): Take as input a security parameter 1<sup>k</sup> and a keyword space W, and probabilistically output a pair of master public-and-secret keys (PK, SK), where PK includes the keyword space W and the ciphertext space C.
- StructureInitialization(PK): Take as input PK, and probabilistically initialize a hidden structure by outputting its private and public parts (Pri, Pub).
- StructuredEncryption(PK, W, Pri): Take as inputs PK, a keyword  $W \in W$  and a hidden structure's private part Pri, and probabilistically output a keyword-searchable ciphertext C of keyword W with the hidden structure, and update Pri.
- Trapdoor(SK, W): Take as inputs SK and a keyword W ∈ W, and output a keyword search trapdoor T<sub>W</sub> of W.
- StructuredSearch(PK, Pub, C, T<sub>W</sub>): Take as inputs PK, a hidden structure's public part Pub, all keyword-searchable ciphertexts C and a keyword search trapdoor T<sub>W</sub> of keyword W, disclose partial relations to guide finding out the ciphertexts containing keyword W with the hidden structure.

An SPCHS scheme must be consistent in the sense that given any keyword search trapdoor  $T_W$  and any hidden structure's public part **Pub**, algorithm **StructuredSearch**(**PK**, **Pub**,  $\mathbb{C}$ ,  $T_W$ ) finds out all ciphertexts of keyword W with the hidden structure **Pub**.

In the application of SPCHS, a receiver runs algorithm **SystemSetup** to set up SPCHS. Each sender uploads the public part of his hidden structure and keyword-searchable ciphertexts to a server, respectively by algorithms **StructureInitialization** and **StructuredEncryption**. Algorithm **Trapdoor** allows the receiver to delegate a keyword search trapdoor to the server. Then the server runs algorithm **StructuredSearch** for all senders' structures to find out the ciphertexts of the queried keyword.

The above SPCHS definition requires each sender to maintain the private part of his hidden structure for algorithm **StructuredEncryption**. A similar requirement appears in symmetric-key encryption with keyword search (SEKS) in which each sender is required to maintain a secret key shared with the receiver. This implies interactions via authenticated confidential channels before a sender encrypts the keywords to the receiver in SEKS. In contrast, each sender in SPCHS just generates and maintains his/her private values locally, i.e., without requirement of extra secure interactions before encrypting keywords.

In the general case of SPCHS, each sender keeps his/her private values **Pri**. We could let each sender be stateless by storing his/her **Pri** in encrypted form at a server and having each sender download and re-encrypt his/her **Pri** for each update of **Pri**. A similar method also has been suggested by [27].

The semantic security of SPCHS is to resist adaptively chosen keyword and structure attacks (SS-CKSA). In this security notion, a probabilistic polynomial-time (PPT) adversary is allowed to know all structures' public parts, query the trapdoors for adaptively chosen keywords, query the private parts of adaptively chosen structures, and query the ciphertexts of adaptively chosen keywords and structures (including the keywords and structures which the adversary would like to be challenged). The adversary will choose two challenge keyword-structure pairs. The SS-CKSA security means that for a ciphertext of one of two challenge keyword-structure pairs, the adversary cannot determine which challenge keyword or which challenge structure the challenge ciphertext corresponds to, provided that the adversary does not know the two challenge keywords' search trapdoors and the two challenge structures' private parts.

**Definition 2** (SS-CKSA Security). Suppose there are at most  $N \in \mathbb{N}$  hidden structures. An SPCHS scheme is SS-CKSA secure, if any PPT adversary  $\mathcal{A}$  has only a negligible advantage  $Adv_{SPCHS,\mathcal{A}}^{SS-CKSA}$  to win in the following SS-CKSA game:

- Setup Phase: A challenger sets up the SPCHS scheme by running algorithm SystemSetup to generate a pair of master public-and-secret keys (PK, SK), and initializes N hidden structures by running algorithm StructureInitialization N times (let PSet be the set of all public parts of these N hidden structures.); finally the challenger sends PK and PSet to A.
- Query Phase 1: A adaptively issues the following queries multiple times.
  - Trapdoor Query  $Q_{Trap}(W)$ : Taking as input a keyword  $W \in W$ , the challenger outputs the keyword search trapdoor of keyword W;
  - Privacy Query  $Q_{Pri}(\mathbf{Pub})$ : Taking as input a hidden structure's public part  $\mathbf{Pub} \in \mathbf{PSet}$ , the challenger outputs the corresponding private part of this structure;
  - Encryption Query  $Q_{Enc}(W, \mathbf{Pub})$ : Taking as inputs a keyword  $W \in W$  and a hidden structure's public part  $\mathbf{Pub}$ , the challenger outputs an SPCHS ciphertext of keyword W with the hidden structure  $\mathbf{Pub}$ .
- Challenge Phase: A sends two challenge keywordand-structure pairs (W<sup>\*</sup><sub>0</sub>, Pub<sup>\*</sup><sub>0</sub>) ∈ W × PSet and

 $(W_1^*, \mathbf{Pub}_1^*) \in \mathcal{W} \times \mathbf{PSet}$  to the challenger; The challenger randomly chooses  $d \in \{0, 1\}$ , and sends a challenge ciphertext  $C_d^*$  of keyword  $W_d^*$  with the hidden structure  $\mathbf{Pub}_d^*$  to  $\mathcal{A}$ .

- Query Phase 2: This phase is the same as Query Phase 1. Note that in Query Phase 1 and Query Phase 2, A cannot query the corresponding private parts both of Pub<sub>0</sub><sup>\*</sup> and Pub<sub>1</sub><sup>\*</sup> and the keyword search trapdoors both of W<sub>0</sub><sup>\*</sup> and W<sub>1</sub><sup>\*</sup>.
- Guess Phase: A sends a guess d' to the challenger. We say that A wins if d = d'. And let  $Adv_{SPCHS,A}^{SS-CKSA} = Pr[d = d'] - \frac{1}{2}$  be the advantage of A to win in the above game.

A weaker security definition of SPCHS is the selectivekeyword security. We refer to this weaker security notion as SS-sK-CKSA security, and the corresponding attack game as SS-sK-CKSA game. In this attack game, the adversary  $\mathcal{A}$  chooses two challenge keywords before the SPCHS scheme is set up, but the adversary still adaptively chooses two challenge hidden structures at **Challenge Phase**. Let  $Adv_{SPCHS,\mathcal{A}}^{SS-sK-CKSA}$  denote the advantage of adversary  $\mathcal{A}$  to win in this game.

# III. A SIMPLE SPCHS SCHEME FROM SCRATCH

Let  $\gamma \stackrel{\$}{\leftarrow} \Re$  denote an element  $\gamma$  randomly sampled from  $\Re$ . Let  $\mathbb{G}$  and  $\mathbb{G}_1$  denote two multiplicative groups of prime order q. Let g be a generator of  $\mathbb{G}$ . A bilinear map  $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$  [44], [45] is an efficiently computable and non-degenerate function, with the bilinearity property  $\hat{e}(g^a, g^b) = \hat{e}(g, g)^{ab}$ , where  $(a, b) \stackrel{\$}{\leftarrow} \mathbb{Z}_q^*$  and  $\hat{e}(g, g)$  is a generator of  $\mathbb{G}_1$ . Let  $\mathbf{BGen}(1^k)$  be an efficient bilinear map generator that takes as input a security parameter  $1^k$ and probabilistically outputs  $(q, \mathbb{G}, \mathbb{G}_1, g, \hat{e})$ . Let keyword space  $\mathcal{W} = \{0, 1\}^*$ .

A simple SPCHS scheme secure in the random oracle model is constructed as follows.

- SystemSetup(1<sup>k</sup>, W): Take as input 1<sup>k</sup> and the keyword space W, compute (q, G, G<sub>1</sub>, g, ê) = BGen(1<sup>k</sup>), pick s <sup>§</sup>→ Z<sub>q</sub><sup>\*</sup>, set P = g<sup>s</sup>, choose a cryptographic hash function H : W → G, set the ciphertext space C ⊆ G<sub>1</sub>×G×G<sub>1</sub>, and finally output the master public key PK = (q, G, G<sub>1</sub>, g, ê, P, H, W, C), and the master secret key SK = s.
- StructureInitialization(PK): Take as input PK, pick u <sup>\$</sup> Z<sup>\*</sup><sub>q</sub>, and initialize a hidden structure by outputting a pair of private-and-public parts (Pri = (u), Pub = g<sup>u</sup>). Note that Pri here is a variable list formed as (u, {(W, Pt[u, W])|W ∈ W}), which is initialized as (u).
- StructuredEncryption(PK, W, Pri): Take as inputs PK, a keyword W ∈ W, a hidden structure's private part Pri, pick r 
   <sup>§</sup> Z<sup>\*</sup><sub>q</sub> and do the following steps:
  - 1) Search (W, Pt[u, W]) for W in **Pri**;
  - 2) If it is not found, insert  $(W, Pt[u, W] \stackrel{\$}{\leftarrow} \mathbb{G}_1)$  to **Pri**, and output the keyword-searchable ci-

phertext  $C = (\hat{e}(P, H(W))^u, g^r, \hat{e}(P, H(W))^r \cdot Pt[u, W]);$ 

- 3) Otherwise, pick  $R \notin \mathbb{G}_1$ , set  $C = (Pt[u, W], g^r, \hat{e}(P, H(W))^r \cdot R)$ , update Pt[u, W] = R, and output the keyword-searchable ciphertext C;
- Trapdoor(SK, W): Take as inputs SK and a keyword W ∈ W, and output a keyword search trapdoor T<sub>W</sub> = H(W)<sup>s</sup> of keyword W.
- StructuredSearch(PK, Pub, C, T<sub>W</sub>): Take as inputs PK, a hidden structure's public part Pub, all keyword-searchable ciphertexts C (let C[i] denote one ciphertext of C, and this ciphertext can be parsed as (C[i, 1], C[i, 2], C[i, 3]) ∈ G<sub>1</sub>×G×G<sub>1</sub>) and a keyword trapdoor T<sub>W</sub> of keyword W, set C' = φ, and do the following steps:
  - 1) Compute  $Pt' = \hat{e}(\mathbf{Pub}, T_W);$
  - Seek a ciphertext C[i] having C[i, 1] = Pt'; if it exists, add C[i] into C';
  - 3) If no matching ciphertext is found, output  $\mathbb{C}'$ ;
  - 4) Compute  $Pt' = \hat{e}(\mathbb{C}[i, 2], T_W)^{-1} \cdot \mathbb{C}[i, 3]$ , and go to Step 2.

Figure 2 shows a hidden star-like structure, which is generated by the SPCHS instance. When running algorithm **StructuredSearch**(**PK**, **Pub**,  $\mathbb{C}$ ,  $T_{W_i}$ ), it discloses the value  $\hat{e}(P, H(W_i))^u$  by computing  $\hat{e}(\mathbf{Pub}, T_{W_i})$ , and matches  $\hat{e}(P, H(W_i))^u$  with all ciphertexts to find out the ciphertext  $(\hat{e}(P, H(W_i))^u, g^r, \hat{e}(P, H(W_i))^r \cdot Pt[u, W_i])$ . Then the algorithm discloses  $Pt[u, W_i]$  by computing  $\hat{e}(g^r, T_{W_i})^{-1} \cdot \hat{e}(P, H(W_i))^r \cdot Pt[u, W_i]$ , and matches  $Pt[u, W_i]$  with all ciphertexts to find out the ciphertext  $(Pt[u, W_i], g^r, \hat{e}(P, H(W_i))^r \cdot R)$ . By carrying on in this way, the algorithm will find out all ciphertexts of keyword  $W_i$  with the hidden star-like structure, and stop the search if no matching ciphertext is found.

Consistency. Roughly algorithm speaking, StructuredSearch repetitively discloses the value of Pt' and matches the value with all ciphertexts' first parts to find out the matching ciphertexts. Since all disclosed values of Pt' are either collision-free (due to the hash function H) and random (according to algorithm StructuredEncryption), no more than one ciphertext matches in each matching process. The found ciphertexts should contain the queried keyword, since given a keyword search trapdoor, algorithm StructuredSearch only can disclose the values of Pt', which are corresponding to the queried keyword. Formally, we have Theorem 1 on consistency whose proof can be found in Supplemental Materials B.

**Theorem 1.** Suppose the hash function H is collisionfree, except with a negligible probability in the security parameter k. The above SPCHS instance is consistent, also except with a negligible probability in the security parameter k.

Semantic Security. The SS-CKSA security of the above SPCHS scheme relies on the DBDH assumption in

**BGen** $(1^k)$ . The definition of DBDH assumption [3] is as follows.

**Definition 3** (The DBDH Assumption). The DBDH problem in **BGen**(1<sup>k</sup>) =  $(q, \mathbb{G}, \mathbb{G}_1, g, \hat{e})$  is defined as the advantage of any PPT algorithm  $\mathcal{B}$ to distinguish the tuples  $(g^a, g^b, g^c, \hat{e}(g, g)^{abc})$  and  $(g^a, g^b, g^c, \hat{e}(g, g)^y)$ , where  $(a, b, c, y) \stackrel{\$}{\leftarrow} \mathbb{Z}_q^{*4}$ . Let  $Adv_{\mathcal{B}}^{DBDH}(1^k) = Pr[\mathcal{B}(g^a, g^b, g^c, \hat{e}(g, g)^{abc}) = 1] Pr[\mathcal{B}(g^a, g^b, g^c, \hat{e}(g, g)^y) = 1]$  be the advantage of algorithm  $\mathcal{B}$  to solve the DBDH problem. We say that the DBDH assumption holds in **BGen**(1<sup>k</sup>), if the advantage  $Adv_{\mathcal{B}}^{DBDH}(1^k)$  is negligible in the parameter k.

In the security proof, we prove that if there is an adversary who can break the SS-CKSA security of the above SPCHS instance in the RO model, then there is an algorithm which can solve the DBDH problem in **BGen** $(1^k)$ . Formally we have Theorem 2 whose proof can be found in Supplemental Materials C.

**Theorem 2.** Let the hash function H be modeled as the random oracle  $Q_H(\cdot)$ . Suppose there are at most  $N \in \mathbb{N}$  hidden structures, and a PPT adversary A wins in the SS-CKSA game of the above SPCHS instance with advantage  $Adv_{SPCHS,A}^{SS-CKSA}$ , in which A makes at most  $q_t$  queries to oracle  $Q_{Trap}(\cdot)$  and at most  $q_p$  queries to oracle  $Q_{Pri}(\cdot)$ . Then there is a PPT algorithm  $\mathcal{B}$  that solves the DBDH problem in **BGen**(1<sup>k</sup>) with advantage

$$Adv_{\mathcal{B}}^{DBDH}(1^k) \approx \frac{27}{(e \cdot q_t \cdot q_p)^3} \cdot Adv_{SPCHS,\mathcal{A}}^{SS-CKSA},$$

where e is the base of natural logarithms.

Forward and Backward Security. Even in the case that a sender gets his local privacy **Pri** compromised, SPCHS still offers forward security. This means that the existing hidden structure of ciphertexts stays confidential, since the local privacy only contains the relationship of the new generated ciphertexts. To offer backward security with SPCHS, the sender can initialize a new structure by algorithm **StructureInitialization** for the new generated ciphertexts. Because the new structure is independent of the old one, the compromised local privacy will not leak the new generated structure.

Search Complexity. All keyword-searchable ciphertexts can be indexed by their first parts' binary bits. Assume that there are in total *n* ciphertexts from  $n_s$  hidden structures, and the *i*-th hidden structure contains  $n_{w,i}$  ciphertexts of keyword  $W \in W$ . With the *i*-th hidden structure, the search complexity is  $O(n_{w,i} \log n)$ . For all hidden structures, the sum search complexity is  $O((n_s + n_w) \log n)$ , where  $n_w = \sum_{i=1}^{n_s} n_{w,i}$ . Since  $n = \sum_{W \in W} n_w$  and  $n_w = \sum_{i=1}^{n_s} n_{w,i}$ , we have that  $n_s \ll n_w \ll n$ . Thus the above SPCHS instance allows a much more efficient search than existing PEKS schemes, which have O(n) search complexity.

One may note that SPCHS loses its significant advantage in search performance compared with PEKS if  $n_s = n$ holds. However, this special case seldom happens. In practice, a sender will extract several keywords from each of



and it will be updated into R after generating each subsequent ciphertext of keyword  $W_i$ .

Figure 2: Hidden star-like structure generated by the above SPCHS instance

Intel CPU E5300 @ 2.60GHz Hardware OS and compiler Win XP and Microsoft VC++ 6.0 Program library MIRACL version 5.4.1 Parameters of bilinear map Elliptic curve  $y^2 = x^3 + A \cdot x + B \cdot x$  $t^m + t^a + t^b + t^c + 1$ Pentanomial basis Base field:  $2^m$ m = 379Α B  $2^m + 2^{(m+1)/2} + 1$ Group order: q315 а h 301 287 с The default unit is decimal.

Table I: System parameters

his files. So we usually have  $n_s \ll n$  even if each sender only has one file. In addition, most related works on SEKS and PEKS assume that each file has several keywords.

**Experiment.** We coded our SPCHS scheme, and tested the time cost of algorithm **StructuredSearch** to execute its cryptographic operations for different numbers of matching ciphertexts. We also coded the PEKS scheme [1]. Table I shows the system parameters including hardware, software and the chosen elliptic curve. Assume there are in total  $10^4$  searchable ciphertexts. PEKS takes about 53.8 seconds search time per keyword, since it must test all ciphertexts for each search. Figure 3 shows the experimental results of SPCHS. It is clear that the time cost of SPCHS is linear with the number of matching ciphertexts, whereas for PEKS it is linear with the number of total ciphertexts. Hence, SPCHS is much more efficient than PEKS.

# IV. A GENERIC CONSTRUCTION OF SPCHS FROM IBKEM AND IBE

In this section, we formalize collision-free full-identity malleable IBKEM and a generic SPCHS construction from IBKEM and IBE.

# A. Reviewing IBE

Before the generic SPCHS construction, let us review the concept of IBE and its Anonymity and Semantic Security both under adaptive-ID and Chosen Plaintext Attacks (Anon-SS-ID-CPA).

**Definition 4** (IBE [11]). *IBE consists of four algorithms:* 

- Setup<sub>IBE</sub>(1<sup>k</sup>, *ID*<sub>IBE</sub>): Take as inputs a security parameter 1<sup>k</sup> and an identity space *ID*<sub>IBE</sub>, and probabilistically output the master public-and-secret-key pair (**PK**<sub>IBE</sub>, **SK**<sub>IBE</sub>), where **PK**<sub>IBE</sub> includes the message space  $\mathcal{M}_{IBE}$ , the ciphertext space  $C_{IBE}$  and the identity space *ID*<sub>IBE</sub>.
- Extract<sub>*IBE*</sub>(SK<sub>*IBE*</sub>, *ID*): Take as inputs SK<sub>*IBE*</sub> and an identity  $ID \in ID_{IBE}$ , and output a decryption key  $\tilde{S}_{ID}$  of *ID*.
- Enc<sub>*IBE*</sub>(PK<sub>*IBE*</sub>, *ID*, *M*): Take as inputs PK<sub>*IBE*</sub>, an identity  $ID \in ID_{$ *IBE* $}$  and a message *M*, and probabilistically output a ciphertext  $\tilde{C}$ .
- **Dec**<sub>*IBE*</sub> $(\tilde{S}_{ID'}, \tilde{C})$ : Take as inputs the decryption key  $\tilde{S}_{ID'}$  of identity ID' and a ciphertext  $\tilde{C}$ , and output a message or  $\perp$ , if the ciphertext is invalid.

An IBE scheme must be consistent in the sense that for any  $\tilde{C} = \mathbf{Enc}_{\text{\tiny IBE}}(\mathbf{PK}_{\text{\tiny IBE}}, ID, M)$  and  $\tilde{S}_{ID'} =$ 



Figure 3: Time cost of SPCHS

**Extract**<sub>*IBE*</sub>(**SK**<sub>*IBE*</sub>, *ID'*), **Dec**<sub>*IBE*</sub>( $\tilde{S}_{ID'}, \tilde{C}$ ) = M holds if ID' = ID, except with a negligible probability in the security parameter k.

In the Anon-SS-ID-CPA security notion of IBE, a PPT adversary is allowed to query the decryption keys for adaptively chosen identities, and adaptively choose two challenge identity-and-message pairs. The Anon-SS-ID-CPA security of IBE means that for a challenge ciphertext, the adversary cannot determine which challenge identity and which challenge message it corresponds to, provided that the adversary does not know the two challenge identities' decryption keys. The Anon-SS-ID-CPA security of an IBE scheme is as follows.

**Definition 5** (Anon-SS-ID-CPA security of IBE [46]). An *IBE scheme is Anon-SS-ID-CPA secure if any PPT adversary*  $\mathcal{B}$  *has only a negligible advantage*  $Adv_{IBE,\mathcal{B}}^{Anon-SS-ID-CPA}$  *to win in the following Anon-SS-ID-CPA game:* 

- Setup Phase: A challenger sets up the IBE scheme by running algorithm Setup<sub>IBE</sub> to generate the master public-and-secret-keys pair (PK<sub>IBE</sub>, SK<sub>IBE</sub>), and sends PK<sub>IBE</sub> to β.
- Query Phase 1: Adversary B adaptively issues the following query multiple times.
  - Decryption Key Query  $Q_{DK}^{IBE}(ID)$ : Taking as input an identity  $ID \in \mathcal{ID}_{IBE}$ , the challenger outputs the decryption key of identity ID.
- Challenge Phase: Adversary B sends two challenge identity-and-message pairs (ID<sub>0</sub><sup>\*</sup>, M<sub>0</sub><sup>\*</sup>) and (ID<sub>1</sub><sup>\*</sup>, M<sub>1</sub><sup>\*</sup>) to the challenger; the challenger picks d <sup>€</sup> {0,1}, and sends the challenge IBE ciphertext C<sub>d</sub><sup>\*</sup> = Enc<sub>IBE</sub>(PK<sub>IBE</sub>, ID<sub>d</sub><sup>\*</sup>, M<sub>d</sub><sup>\*</sup>) to B.
  Query Phase 2: This phase is the same as Query
- Query Phase 2: This phase is the same as Query Phase 1. Note that in Query Phase 1 and Query Phase 2, β cannot query the decryption key corresponding to the challenge identity ID<sub>0</sub><sup>6</sup> or ID<sub>1</sub><sup>\*</sup>.
- Guess Phase: Adversary  $\mathcal{B}$  sends a guess  $\tilde{d}'$  to the challenger. We say that  $\mathcal{B}$  wins if  $\tilde{d}' = \tilde{d}$ . Let  $Adv_{IBE,\mathcal{B}}^{Anon-SS-ID-CPA} = Pr[\tilde{d}' = \tilde{d}] \frac{1}{2}$  be the advantage of  $\mathcal{B}$  to win in the above game.

#### B. The Collision-Free Full-Identity Malleable IBKEM

Our generic construction also relies on a notion of collision-free full-identity malleable IBKEM. The following IBKEM definition is derived from [47]. A difference only appears in algorithm  $\mathbf{Encaps}_{IBKEM}$ . In order to highlight that the generator of an IBKEM encapsulation knows the chosen random value used in algorithm  $\mathbf{Encaps}_{IBKEM}$ , we take the random value as an input of the algorithm.

# Definition 6 (IBKEM). IBKEM consists of four algorithms:

• Setup<sub>IBKEM</sub>  $(1^k, \mathcal{ID}_{IBKEM})$ : Take as inputs a security parameter  $1^k$  and an identity space  $\mathcal{ID}_{IBKEM}$ , and probabilistically output the master public-and-secretkeys pair (**PK**<sub>IBKEM</sub>, **SK**<sub>IBKEM</sub>), where **PK**<sub>IBKEM</sub> includes the identity space  $\mathcal{ID}_{IBKEM}$ , the encapsulated key space  $\mathcal{K}_{IBKEM}$  and the encapsulation space  $C_{IBKEM}$ .

- Extract<sub>IBKEM</sub> (SK<sub>IBKEM</sub>, ID): Take as inputs SK<sub>IBKEM</sub> and an identity  $ID \in ID_{IBKEM}$ , and output a decryption key  $\hat{S}_{ID}$  of ID.
- Encaps<sub>IBKEM</sub>(PK<sub>IBKEM</sub>, ID, r): Take as inputs PK<sub>IBKEM</sub>, an identity  $ID \in \mathcal{ID}_{IBKEM}$  and a random value r, and deterministically output a key-and-encapsulation pair  $(\hat{K}, \hat{C})$  of ID.
- **Decaps**<sub>*IBKEM*</sub>( $\hat{S}_{ID'}, \hat{C}$ ): Take as inputs the decryption key  $\hat{S}_{ID'}$  of identity ID' and an encapsulation  $\hat{C}$ , and output an encapsulated key or  $\bot$ , if the encapsulation is invalid.

An IBKEM scheme must be consistent in the sense that for any  $(\hat{K}, \hat{C}) = \mathbf{Encaps}_{IBKEM}(\mathbf{PK}_{IBKEM}, ID, r)$ ,  $\mathbf{Decaps}_{IBKEM}(\hat{S}_{ID'}, \hat{C}) = \hat{K}$  holds if ID' = ID, except with a negligible probability in the security parameter k.

The collision-free full-identity malleable IBKEM implies the following characteristics: all identities' decryption keys can decapsulate the same encapsulation; all decapsulated keys are collision-free; the generator of the encapsulation can also compute these decapsulated keys; the decapsulated keys of different encapsulations are also collision-free.

**Definition 7** (Collision-Free Full-Identity Malleable IBKEM). *IBKEM is collision-free full-identity malleable*, if there is an efficient function **FIM** that for any  $(\hat{K}, \hat{C}) =$ **Encaps**<sub>IBKEM</sub>(**PK**<sub>IBKEM</sub>, *ID*, *r*), the function **FIM** satisfies the following features:

- (Full-Identity Malleability) For any identity  $ID' \in \mathcal{ID}_{IBKEM}$ , the equation  $\mathbf{FIM}(ID', r) =$   $\mathbf{Decaps}_{IBKEM}(\hat{S}_{ID'}, \hat{C})$  always holds, where  $\hat{S}_{ID'} = \mathbf{Extract}_{IBKEM}(\mathbf{SK}_{IBKEM}, ID');$
- (Collision-Freeness) For any identity  $ID' \in \mathcal{ID}_{\text{IBKEM}}$ and any random value r', if  $ID \neq ID' \bigvee r \neq r'$ , then  $\mathbf{FIM}(ID, r) \neq \mathbf{FIM}(ID', r')$  holds, except with a negligible probability in the security parameter k.

A collision-free full-identity malleable IBKEM scheme may preserve semantic security and anonymity. We incorporate the semantic security and anonymity into Anon-SS-ID-CPA secure IBKEM. But this security is different from the traditional version [47] of the Anon-SS-ID-CPA security due to the full-identity malleability of IBKEM. The difference will be introduced after defining that security. In that security, a PPT adversary is allowed to query the decryption keys for adaptively chosen identities, and adaptively choose two challenge identities. The Anon-SS-ID-CPA security of IBKEM means that for a challenge key-and-encapsulation pair, the adversary cannot determine the correctness of this pair and the challenge identity of this pair, given that the adversary does not know the two challenging identities' decryption keys. The Anon-SS-ID-CPA security of a collision-free full-identity malleable IBKEM scheme is as follows.

**Definition 8** (Anon-SS-ID-CPA security of IBKEM). An *IBKEM scheme is Anon-SS-ID-CPA secure if any PPT adversary B has only a negligible advantage*  $Adv_{IBKEM,B}^{Anon-SS-ID-CPA}$  to win in the following Anon-SS-ID-CPA game:

- Setup Phase: A challenger sets up the IBKEM scheme by running algorithm  $\mathbf{Setup}_{\text{IBKEM}}$  to generate the master public-and-secret-keys pair ( $\mathbf{PK}_{\text{IBKEM}}, \mathbf{SK}_{\text{IBKEM}}$ ), and sends  $\mathbf{PK}_{\text{IBKEM}}$  to  $\mathcal{B}$ .
- Query Phase 1: B adaptively issues the following query multiple times.
  - Decryption Key Query  $\mathcal{Q}_{DK}^{IBKEM}(ID)$ : Taking as input an identity  $ID \in \mathcal{ID}_{IBKEM}$ , the challenger outputs the decryption key of identity ID.
- Challenge Phase:  $\mathcal{B}$  sends two challenge identities  $ID_0^*$  and  $ID_1^*$  to the challenger; the challenger picks  $\hat{d} \stackrel{\$}{\leftarrow} \{0,1\}$ , computes  $(\hat{K}_0^*, \hat{C}_0^*) =$ **Encaps**<sub>IBKEM</sub> $(PK_{IBKEM}, ID_0^*, r_0)$  and  $(\hat{K}_1^*, \hat{C}_1^*) =$ **Encaps**<sub>IBKEM</sub> $(PK_{IBKEM}, ID_1^*, r_1)$ , and sends the challenge key-and-encapsulation pair  $(\hat{K}_{\hat{d}}^*, \hat{C}_0^*)$  to  $\mathcal{B}$ , where  $r_0$  and  $r_1$  are randomly chosen.
- Query Phase 2: This phase is the same as Query Phase 1. Note that in Query Phase 1 and Query Phase 2, β cannot query the decryption keys both of the challenge identities ID<sub>0</sub><sup>6</sup> and ID<sub>1</sub><sup>\*</sup>.
- Guess Phase:  $\mathcal{B}$  sends a guess  $\hat{d}'$  to the challenger. We say that  $\mathcal{B}$  wins if  $\hat{d}' = \hat{d}$ . Let  $Adv_{IBKEM,\mathcal{B}}^{Anon-SS-ID-CPA} = Pr[\hat{d}' = \hat{d}] - \frac{1}{2}$  be the advantage of  $\mathcal{B}$  to win in the above game.

In the above definition, the anonymity of the encapsulated keys is defined by the indistinguishability of  $\hat{K}_0^*$  and  $\hat{K}_1^*$ . But we do not define the anonymity of the IBKEM encapsulations (*i.e.* the challenge key-and-encapsulation pair consists of  $\hat{C}_0^*$  instead of  $\hat{C}_d^*$ ), since the full-identity malleability of IBKEM implies that any IBKEM encapsulation is valid for all identities.

A weaker security definition of IBKEM is the selectiveidentity security, referred to as the Anon-SS-sID-CPA security. The corresponding attack game is called the Anon-SS-sID-CPA game in which the adversary must commit to the two challenge identities before the system is set up.

# C. The Proposed Generic SPCHS Construction

Let keyword space  $\mathcal{W} \subset \mathcal{ID}_{\text{IBKEM}} = \mathcal{ID}_{\text{IBE}}$ . Our generic SPCHS construction from the collision-free full-identity malleable IBKEM and IBE is as follows.

- SystemSetup $(1^k, W)$ : Take as inputs a security parameter  $1^k$  and the keyword space W, run  $(\mathbf{PK}_{\text{IBKEM}}, \mathbf{SK}_{\text{IBKEM}}) = \mathbf{Setup}_{\text{IBKEM}}(1^k, \mathcal{ID}_{\text{IBKEM}})$  and  $(\mathbf{PK}_{\text{IBE}}, \mathbf{SK}_{\text{IBE}}) = \mathbf{Setup}_{\text{IBE}}(1^k, \mathcal{ID}_{\text{IBE}})$ , and output a pair of master public-and-secret keys  $(\mathbf{PK} = (\mathbf{PK}_{\text{IBKEM}}, \mathbf{PK}_{\text{IBE}}), \mathbf{SK} = (\mathbf{SK}_{\text{IBKEM}}, \mathbf{SK}_{\text{IBE}}))$ . Let the SPCHS ciphertext space  $\mathcal{C} = \mathcal{K}_{\text{IBKEM}} \times \mathcal{C}_{\text{IBE}}$ , and  $\mathcal{K}_{\text{IBKEM}} = \mathcal{M}_{\text{IBE}}$ .
- StructureInitialization(PK): Take as input PK, arbitrarily pick a keyword  $W \in W$  and a random value u, generate an IBKEM encapsulated key and its encapsulation  $(\hat{K}, \hat{C}) = \mathbf{Encaps}_{\text{IBKEM}}(\mathbf{PK}_{\text{IBKEM}}, W, u)$ , and initialize a hidden structure by outputting a pair of private-and-public parts ( $\mathbf{Pri} = (u), \mathbf{Pub} = \hat{C}$ ). Note that  $\mathbf{Pri}$  here is a variable list formed as

 $(u, \{(W, Pt[u, W]) | W \in W\})$ , which is initialized as (u).

(In the above, an IBKEM encapsulation and its related random value are respectively taken as the public-andprivate parts of a hidden structure. To generate these two parts , an arbitrary keyword have to be chosen to run algorithm  $\mathbf{Encaps}_{\text{IBKEM}}$ .)

- StructuredEncryption(PK, W, Pri): Take as inputs PK, a keyword  $W \in W$ , a hidden structure's private part Pri, and do the following steps:
  - 1) Search (W, Pt[u, W]) for W in **Pri**;
  - 2) If it is not found, insert  $(W, Pt[u, W] \leftarrow \mathcal{M}_{\text{IBE}})$  to **Pri**, and output the keyword-searchable ciphertext  $C = (\mathbf{FIM}(W, u), \mathbf{Enc}_{\text{IBE}}(\mathbf{PK}_{\text{IBE}}, W, Pt[u, W]);$
  - 3) Otherwise, pick  $R \stackrel{\$}{\leftarrow} \mathcal{M}_{\text{\tiny IBE}}$ , set  $C = (Pt[u, W], \mathbf{Enc}_{\text{\tiny IBE}}(\mathbf{PK}_{\text{\tiny IBE}}, W, R))$ , update Pt[u, W] = R, and output the keyword-searchable ciphertext C;
- Trapdoor(SK, W): Take as inputs SK and a keyword  $W \in W$ , run  $\hat{S}_W = \mathbf{Extract}_{\text{IBKEM}}(\mathbf{SK}_{\text{IBKEM}}, W)$ and  $\tilde{S}_W = \mathbf{Extract}_{\text{IBE}}(\mathbf{SK}_{\text{IBE}}, W)$ , and output a keyword search trapdoor  $T_W = (\hat{S}_W, \tilde{S}_W)$  of keyword W.
- StructuredSearch(PK, Pub,  $\mathbb{C}, T_W$ ): Take as inputs PK, a hidden structure's public part Pub, all keyword-searchable ciphertexts  $\mathbb{C}$  (let  $\mathbb{C}[i]$  denote one ciphertext of  $\mathbb{C}$ , and this ciphertext can be parsed as  $(\mathbb{C}[i, 1], \mathbb{C}[i, 2]) \in \mathcal{C} = \mathcal{K}_{\text{IBKEM}} \times \mathcal{C}_{\text{IBE}}$ ) and a keyword trapdoor  $T_W = (\hat{S}_W, \tilde{S}_W)$  of keyword W, set  $\mathbb{C}' = \phi$ , and do the following steps:
  - 1) Compute  $Pt' = \mathbf{Decaps}_{\mathrm{IBKEM}}(\hat{S}_W, \mathbf{Pub});$
  - Seek a ciphertext C[i] having C[i, 1] = Pt'; if it exists, add C[i] into C';
  - 3) If no matching ciphertext is found, output  $\mathbb{C}'$ ;
  - 4) Compute  $Pt' = \mathbf{Dec}_{\text{\tiny IBE}}(\hat{S}_{ID'}, \mathbb{C}[i, 2])$ , go to step 2;

Figure 4 shows a hidden star-like structure generated by the generic SPCHS construction. When running algorithm **StructuredSearch**(**PK**, **Pub**,  $\mathbb{C}$ ,  $T_{W_i}$ ), the full-identity malleability of IBKEM allows the algorithm to disclose the value **FIM**( $W_i$ , u) by computing **FIM**( $W_i$ , u) = **Decaps**<sub>IBKEM</sub>( $\hat{S}_{W_i}$ , **Pub**) and find out the ciphertext (**FIM**( $W_i$ , u), **Enc**<sub>IBE</sub>(**PK**<sub>IBE</sub>,  $W_i$ ,  $Pt[u, W_i]$ )). Then the consistency of IBE allows the algorithm to disclose  $Pt[u, W_i]$  by decrypting **Enc**<sub>IBE</sub>(**PK**<sub>IBE</sub>,  $W_i$ ,  $Pt[u, W_i]$ ) and find out the ciphertext ( $Pt[u, W_i]$ , **Enc**<sub>IBE</sub>(**PK**<sub>IBE</sub>,  $W_i$ , R)). By carrying on in this way, the consistency of IBE allows the algorithm to find out the rest of ciphertexts of keyword  $W_i$  with the hidden star-like structure, and stop the search if no more ciphertexts are found.

**Consistency.** When running the above algorithm **StructuredSearch**(**PK**, **Pub**,  $\mathbb{C}$ ,  $T_W$ ), the consistency and full-identity malleability of IBKEM assures that **FIM**(W, u) = **Decaps**<sub>IBKEM</sub>( $\hat{S}_W$ , **Pub**) holds. The collision-freeness of IBKEM assures that only one ciphertext containing keyword W has the value **FIM**(W, u) as



Figure 4: Hidden star-like structure generated by the generic SPCHS construction

its first part. Therefore the algorithm can find out the first ciphertext of keyword W with the hidden structure **Pub**. Then the consistency of IBE allows the algorithm **StructuredSearch** to find out the rest of ciphertexts containing keyword W with the hidden structure **Pub**. Formally we have Theorem 3. The proof can be found in Supplemental Materials D.

# **Theorem 3.** The above generic SPCHS scheme is consistent if its underlying collision-free full-identity malleable IBKEM and IBE schemes are both consistent.

**Semantic Security.** The SS-sK-CKSA security of the above generic SPCHS construction relies on the Anon-SS-sID-CPA security of the underlying IBKEM and the Anon-SS-ID-CPA security of the underlying IBE. In the security proof, we prove that if there is an adversary who can break the SS-sK-CKSA security of the above generic SPCHS construction, then there is another adversary who can break the Anon-SS-ID-CPA security of the underlying IBKEM or the Anon-SS-ID-CPA security of the underlying IBKEM or the Anon-SS-ID-CPA security of the underlying IBKEM or the Anon-SS-ID-CPA security of the underlying IBE. Theorem 4 formally states the semantic security of our generic SPCHS construction. The proof can be found in Supplemental Materials E.

**Theorem 4.** Suppose there are at most  $N \in \mathbb{N}$  hidden structures, and a PPT adversary  $\mathcal{A}$  wins in the SS-sK-CKSA game with advantage  $Adv_{SPCHS,\mathcal{A}}^{SS-sK-CKSA}$ . Then there is a PPT adversary  $\mathcal{B}$ , who utilizes the capability of  $\mathcal{A}$  to win in the Anon-SS-sID-CPA game of the underlying IBKEM or the Anon-SS-ID-CPA game of the underlying IBE with advantage  $\frac{1}{4N} \cdot Adv_{SPCHS,\mathcal{A}}^{SS-sK-CKSA}$ .

# V. Two Collision-Free Full-Identity Malleable IBKEM Instances

The Instance in the RO Model. Abdalla *et al.* proposed several VRF-suitable IBKEM instances in [8]. An IBKEM instance is VRF-suitable if it provides *unique decapsulation*. This means that given any encapsulation, all the decryption keys corresponding to the same identity

decapsulate out the same encapsulated key, and the key is pseudo-random. Here, the decryption key extraction is probabilistic and for the same identity, different decryption key may be extracted in different runs of the key extraction algorithm. It is clear that our proposed collision-free fullidentity malleability not only implies *unique decapsulation*, but also implies that the generator of an encapsulation knows what keys will be decapsulated by the decryption keys of all identities. In Supplemental Materials F, we prove that the VRF-suitable IBKEM instance proposed in Appendix A.2 of [8] is collision-free full-identity malleable. Even though this IBKEM scheme has the traditional Anon-SS-ID-CPA security, we further prove that this IBKEM scheme is Anon-SS-ID-CPA secure based on the DBDH assumption in the RO model according to Definition 8.

The Instance in the Standard Model. In [9], Freire et al. utilized the "approximation" of multilinear maps [10] to construct a programmable hash function in the multilinear setting (MPHF). Then Freire et al. utilized this hash function to replace the traditional hash functions of the BF IBE scheme in [11] and reconstructed this IBE scheme in the multilinear setting. They finally constructed a new IBE scheme with semantic security in the standard model. We find that this new IBE scheme can be easily transformed into a collision-free full-identity malleable IBKEM scheme with Anon-SS-ID-CPA security in the standard model. To simplify the description of this IBKEM scheme, we do not consider the "approximation" of multilinear maps. This means that we will leave out the functions that are the encoding of a group element, the re-randomization of an encoding and the extraction of an encoding. Some related definitions are reviewed as follows.

**Definition 9** (Multilinear Maps [9]). An  $\ell$ -group system in the multilinear setting consists of  $\ell$  cyclic groups  $\mathbb{G}_1, \dots, \mathbb{G}_\ell$  of prime order p, along with bilinear maps  $\hat{e}_{i,j} : \mathbb{G}_i \times \mathbb{G}_j \to \mathbb{G}_{i+j}$  for all  $i, j \ge 1$  with  $i+j \le \ell$ . Let  $g_i$  be a generator of  $\mathbb{G}_i$ . The map  $\hat{e}_{i,j}$  satisfies  $\hat{e}_{i,j}(g_i^a, g_j^b) = g_{i+j}^{ab}$  (for all  $a, b \in \mathbb{Z}_p$ ). When i, j are clear, we will simply

write  $\hat{e}$  instead of  $\hat{e}_{i,j}$ . It will also be convenient to abbreviate  $\hat{e}(h_1, \dots, h_j) = \hat{e}(h_1, \hat{e}(h_2, \dots, \hat{e}(h_{j-1}, h_j) \dots))$  for  $h_j \in \mathbb{G}_{i_j}$  and  $i = (i_1 + i_2 + \dots + i_j) \leq \ell$ . By induction, it is easy to see that this map is *j*-linear. Additionally, We define  $\hat{e}(g) = g$ . Finally, it can also be useful to define the group  $\mathbb{G}_0 = \mathbb{Z}^+_{|\mathbb{G}_1|}$  of exponents to which this pairing family naturally extends. In the following, we will assume an  $\ell$ group system  $\mathbf{MPG}_{\ell} = \{\{\mathbb{G}_i\}_{i\in[1,\ell]}, p, \{\hat{e}_{i,j}\}_{i,j\geq 1, i+j\leq \ell}\}$ generated by a multilinear maps parameter generator  $\mathbf{MG}_{\ell}$ on input a security parameter  $1^k$ .

**Definition 10** (The  $\ell$ -MDDH Assumption [9]). Given  $(g, g^{x_1}, \dots, g^{x_{\ell+1}})$  (for  $g \stackrel{\$}{\leftarrow} \mathbb{G}_1$  and uniform exponents  $x_i$ ), the  $\ell$ -MDDH assumption is that the element  $\hat{e}(g^{x_1}, \dots, g^{x_\ell})^{x_{\ell+1}} \in \mathbb{G}_\ell$  is computationally indistinguishable from a uniform  $\mathbb{G}_\ell$ -element.

**Definition 11** (Group hash function [9]). A group hash function **H** into  $\mathbb{G}$  consists of two polynomial-time algorithms: the probabilistic algorithm  $\mathbf{HGen}(1^k)$  outputs a key hk, and  $\mathbf{HEval}(hk, X)$  (for a key hk and  $X \in$  $\{0, 1\}^k$ ) deterministically outputs an image  $\mathbf{H}_{hk}(X) \in \mathbb{G}$ .

**Definition 12** (MPHF [9]). Assume an  $\ell'$ -group system  $\mathbf{MPG}_{\ell'}$  as generated by  $\mathbf{MG}_{\ell'}(1^k)$ . Let  $\mathbf{H}$  be a group hash function into  $\mathbb{G}_{\ell}(\ell \leq \ell')$ , and let  $m, n \in \mathbb{N}$ . We say that  $\mathbf{H}$  is an (m,n)-programmable hash function in the multilinear setting ((m,n)-MPHF) if there are PPT algorithms **TGen** and **TEval** as follows.

- **TGen** $(1^k, c_1, \dots, c_l, h)$  (for  $c_i, h \in \mathbb{G}_1$  and  $h \neq 1$ ) outputs a key hk and a trapdoor td. We require that for all  $c_i, h$ , that distribution of hk is statistically close to the output of **HGen**.
- **TEval**(td, X) (for a trapdoor td and  $X \in \{0, 1\}^k$ ) deterministically outputs  $a_X \in \mathbb{Z}_p^*$  and  $B_X \in \mathbb{G}_{\ell-1}$ with  $\mathbf{H}_{hk}(X) = \hat{e}(c_1, \cdots, c_\ell)^{a_X} \cdot \hat{e}(B_X, h)$ . We require that there is a polynomial p(k) such that for all hk and  $X_1, \cdots, X_m, Z_1, \cdots, Z_n \in \{0, 1\}^k$  with  $\{X_i\}_i \cap \{Z_j\}_j = \emptyset, P_{hk, \{X_i\}, \{Z_j\}} = Pr[(a_{X_1} = \cdots = a_{X_m} = 0) \land (a_{Z_1}, \cdots, a_{X_n} \neq 0)] \ge 1/p(k)$ , where the probability is over possible trapdoors tdoutput by **TGen** along with the given hk. Furthermore, we require that  $P_{hk, \{X_i\}, \{Z_j\}}$  is close to statistically independent of hk. (Formally,  $|P_{hk, \{X_i\}, \{Z_j\}} - P_{hk', \{X_i\}, \{Z_j\}}| \le v(k)$  for all hk and hk' in the range of **TGen**, all  $\{X_i\}, \{Z_j\}$ , and negligible v(k).)

We say that **H** is a (poly, n)-MPHF if it is a (q(k), n)-MPHF for every polynomial q(k). Note that **TEval** algorithm of an MPHF into  $\mathbb{G}_1$  yields  $B_X \in \mathbb{G}_0$ , i.e., exponents  $B_X$ .

Let identity space  $\mathcal{ID}_{\text{IBKEM}} = \{0,1\}^k$ . The IBKEM instance in the standard model is as follows.

• Setup<sub>IBKEM</sub> $(1^k, \mathcal{ID}_{IBKEM})$ : Take as input a security parameter  $1^k$  and the identity space  $\mathcal{ID}_{IBKEM}$ , generate an  $(\ell + 1)$ -group system  $\mathbf{MPG}_{\ell+1} =$  $\{\{\mathbb{G}_i\}_{i\in[1,\ell+1]}, p, \{\hat{e}_{i,j}\}_{i,j\geq 1, i+j\leq \ell+1}\}$   $\leftarrow$  $\mathbf{MG}_{\ell+1}(1^k)$ , generate a (poly, 2)-MPHF **H** into  $\mathbb{G}_\ell$  and  $hk \leftarrow$  **HGen** $(1^k)$ , choose  $h \stackrel{\$}{\leftarrow} \mathbb{G}_1$  and  $x \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ , set the encapsulated key space  $\mathcal{K}_{\text{IBKEM}} = \mathbb{G}_{\ell+1}$ , set the encapsulation space  $\mathcal{C}_{\text{IBKEM}} = \mathbb{G}_1$ , and output the master public key  $\mathbf{PK}_{\text{IBKEM}} = (\mathbf{MPG}_{\ell+1}, hk, \mathbf{H}, h, h^x, \mathcal{ID}_{\text{IBKEM}}, \mathcal{K}_{\text{IBKEM}}, \mathcal{C}_{\text{IBKEM}})$  and the master secret key  $\mathbf{SK}_{\text{IBKEM}} = (hk, x)$ .

- Extract<sub>IBKEM</sub> (SK<sub>IBKEM</sub>, *ID*): Take as inputs SK<sub>IBKEM</sub> and an identity  $ID \in \mathcal{ID}_{IBKEM}$ , and output a decryption key  $\hat{S}_{ID} = \mathbf{H}_{hk}(ID)^x$  of *ID*.
- Encaps<sub>IBKEM</sub> (**PK**<sub>IBKEM</sub>, *ID*, *r*): Take as inputs **PK**<sub>IBKEM</sub>, an identity  $ID \in \mathcal{ID}_{IBKEM}$  and a random value  $r \in \mathbb{Z}_p^*$ , and output a key-and-encapsulation pair  $(\hat{K}, \hat{C})$ , where  $\hat{K} = \hat{e}(\mathbf{H}_{hk}(ID), h^x)^r \in \mathbb{G}_{\ell+1}$ and  $\hat{C} = h^r$ .
- **Decaps**<sub>IBKEM</sub>  $(\hat{S}_{ID'}, \hat{C})$ : Take as inputs the decryption key  $\hat{S}_{ID'}$  of identity ID' and an encapsulation  $\hat{C}$ , and output the encapsulated key  $\hat{K} = \hat{e}(\hat{C}, \hat{S}_{ID'}) \in \mathbb{G}_{\ell+1}$ if  $\hat{C} \in \mathbb{G}_1$  or output  $\perp$  otherwise.

**Consistency.** According to Definitions 9 and 11, it is very easy to verify the consistency of the above IBKEM scheme.

**Collision-Free Full-Identity Malleability.** Let the function **FIM** $(ID, r) = \hat{e}(h^x, \mathbf{H}_{hk}(ID))^r \in \mathbb{G}_{\ell+1}$  for any identity  $ID \in \mathcal{ID}_{\text{IBKEM}}$  and any random value  $r \in \mathbb{Z}_p^*$ . Given any  $(\hat{K}, \hat{C}) \leftarrow \mathbf{Encaps}_{\text{IBKEM}}(\mathbf{PK}_{\text{IBKEM}}, ID, r)$ , we clearly have that: (1) for any identity ID', equation **FIM** $(ID', r) = \mathbf{Decaps}_{\text{IBKEM}}(\hat{S}_{ID'}, \hat{C})$  holds; (2) for any identity ID'and any random value r', if  $ID' \neq ID \bigvee r' \neq r$  holds, equation **FIM** $(ID, r) \neq \mathbf{FIM}(ID', r')$  holds except with a negligible probability. So the above IBKEM scheme is collision-free full-identity malleable.

**Anon-SS-ID-CPA Security.** In [9], Freire *et al.* utilized a (poly, 1)-MPHF to construct a standard-model version of the BF IBE scheme with the SS-ID-CPA security. On the contrary, we use a (poly, 2)-MPHF in constructing the above IBKEM scheme, since this kind of MPHF is more useful in proving the Anon-SS-ID-CPA security. Theorem 5 formally states the Anon-SS-ID-CPA security of the above IBKEM scheme. The proof can be found in Supplemental Materials G.

**Theorem 5.** Assume the above IBKEM scheme is implemented in an  $(\ell + 1)$ -group system, and with a (poly, 2)-MPHF **H** into  $\mathbb{G}_{\ell}$ . Then, under the  $(\ell+1)$ -MDDH assumption, this IBKEM scheme is Anon-SS-ID-CPA secure.

According to Theorem 4 and 5, the generic SPCHS construction implies a SPCHS instance with SS-sK-CKSA security in the standard model. Indeed, this SPCHS instance can be provably SS-CKSA secure.

# VI. CONCLUSION AND FUTURE WORK

This paper investigated as-fast-as-possible search in PEKS with semantic security. We proposed the concept of SPCHS as a variant of PEKS. The new concept allows keyword-searchable ciphertexts to be generated with a hidden structure. Given a keyword search trapdoor, the search algorithm of SPCHS can disclose part of this hidden structure for guidance on finding out the ciphertexts of the queried keyword. Semantic security of SPCHS captures the privacy of the keywords and the invisibility of the hidden structures. We proposed an SPCHS scheme from scratch with semantic security in the RO model. The scheme generates keyword-searchable ciphertexts with a hidden star-like structure. It has search complexity mainly linear with the exact number of the ciphertexts containing the queried keyword. It outperforms existing PEKS schemes with semantic security, whose search complexity is linear with the number of all ciphertexts. We identified several interesting properties, i.e., collision-freeness and full-identity malleability in some IBKEM instances, and formalized these properties to build a generic SPCHS construction. We illustrated two collision-free full-identity malleable IBKEM instances, which are respectively secure in the RO and standard models.

SPCHS seems a promising tool to solve some challenging problems in public-key searchable encryption. One application may be to achieve retrieval completeness verification which, to the best of our knowledge, has not been achieved in existing PEKS schemes. Specifically, by forming a hidden ring-like structure, i.e., letting the last hidden pointer always point to the head, one can obtain PEKS allowing to check the completeness of the retrieved ciphertexts by checking whether the pointers of the returned ciphertexts form a ring.

Another application may be to realize public key encryption with content search, a similar functionality realized by symmetric searchable encryption. Such kind of contentsearchable encryption is useful in practice, e.g., to filter the encrypted spams. Specially, by forming a hidden treelike structure between the sequentially encrypted words in one file, one can obtain public-key searchable encryption allowing content search (e.g., to find whether there are specific contents in an encrypted file). The search complexity is linear with the size of the queried content.

#### **ACKNOWLEDGMENTS**

The authors would like to thank the reviewers for their valuable suggestions that helped to improve the paper greatly. The first author is partly supported by the National Natural Science Foundation of China under grant no. 61472156 and the National Program on Key Basic Research Project (973 Program) under grant no. 2014CB340600. The second author is supported by by the Chinese National Key Basic Research Program (973 program) through project 2012CB315905, the Natural Science Foundation of China through projects 61370190, 61173154, 61472429, 61402029, 61272501, 61202465, 61321064 and 61003214, the Beijing Natural Science Foundation through project 4132056, the Fundamental Research Funds for the Central Universities, and the Research Funds (No. 14XNLF02) of Renmin University of China and the Open Research Fund of Beijing Key Laboratory of Trusted Computing. The fifth author is partly support by the European Commission (H2020 project "CLARUS"), the Government of Catalonia (grant 2014 SGR 537 and ICREA Acadèmia Award

2013) and the Spanish Government (TIN2011-27076-C03-01 "CO-PRIVACY"). The views in this paper do not necessarily reflect the views of UNESCO.

#### REFERENCES

- Boneh D., Crescenzo G. D., Ostrovsky R., Persiano G.: Public Key Encryption with Keyword Search. In: Cachin C., Camenisch J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506-522. Springer, Heidelberg (2004)
- [2] Bellare M., Boldyreva A., O'Neill A.: Deterministic and Efficiently Searchable Encryption. In: Menezes A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 535-552. Springer, Heidelberg (2007)
- [3] Boneh D., Boyen X.: Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In: Cachin C., Camenisch J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223-238. Springer, Heidelberg (2004)
- [4] Boyen X., Waters B. R.: Anonymous Hierarchical Identity-Based Encryption (Without Random Oracles). In: Dwork C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 290-307. Springer, Heidelberg (2006)
- [5] Gentry C.: Practical Identity-Based Encyption Without Random Oracles. In: Vaudenay S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp.445-464. Springer, Heidelberg (2006)
- [6] Ateniese G., Gasti P.: Universally Anonymous IBE Based on the Quadratic Residuosity Assumption. In: Fischlin M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 32-47. Springer, Heidelberg (2009)
- [7] Ducas L.: Anonymity from Asymmetry: New Constructions for Anonymous HIBE. In: Pieprzyk J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 148-164. Springer, Heidelberg (2010)
- [8] Abdalla M., Catalano D., Fiore D.: Verifiable Random Functions: Relations to Identity-Based Key Encapsulation and New Constructions. Journal of Cryptology, 27(3), pp. 544-593 (2013)
- [9] Freire E.S.V., Hofheinz D., Paterson K.G., Striecks C.: Programmable Hash Functions in the Multilinear Setting. In: Canetti R., Garay J.A. (eds.) Advances in Cryptology - CRYPTO 2013. LNCS, vol. 8042, pp. 513-530. Springer, Heidelberg (2013)
- [10] Garg S., Gentry C., Halevi S.: Candidate Multilinear Maps from Ideal Lattices. In: Johansson T., Nguyen P. (eds.) Advances in Cryptology - EUROCRYPT 2013. LNCS, vol. 7881, pp. 1-17. Springer, Heidelberg (2013)
- [11] Boneh D., Franklin M.: Identity-Based Encryption from the Weil Pairing. In: Kilian J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213-239. Springer, Heidelberg (2001)
- [12] Barth A., Boneh D., Waters B.: Privacy in Encrypted Content Distribution Using Private Broadcast Encryption. In: Di Crescenzo G., Rubin A.(eds.) FC 2006. LNCS, vol. 4107, pp. 52-64. Springer, Heidelberg (2006)
- [13] Libert B., Paterson K. G., Quaglia E. A.: Anonymous Broadcast Encryption: Adaptive Security and Efficient Constructions in the Standard Model. In: Fischlin M., Buchmann J., Manulis M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 206-224. Springer, Heidelberg (2012)
- [14] Curtmola R., Garay J., Kamara S., Ostrovsky R.: Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions. In: ACM CCS 2006, pp. 79-88. ACM (2006)
- [15] Song D. X., Wagner D., Perrig A.: Practical techniques for searches on encrypted data. In: IEEE S&P 2000, pp. 44-55. IEEE (2000)
- [16] Goh E.-J.: Secure Indexes. Cryptography ePrint Archive, Report 2003/216 (2003)
- [17] Bellovin S. M., Cheswick W.R.: Privacy-Enhanced Searches Using Encrypted Bloom Filters. Cryptography ePrint Archive, Report 2004/022 (2004)
- [18] Agrawal R., Kiernan J., Srikant R., Xu Y.: Order Preserving Encryption for Numeric Data. In: Proceedings of the 2004 ACM SIGMOD international conference on Management of data, pp. 563-574. ACM (2004)
- [19] Chang Y.-C., Mitzenmacher M.: Privacy Preserving Keyword Searches on Remote Encrypted Data. In: Ioannidis J., Keromytis A. and Yung M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 442-455. Springer, Heidelberg (2005)
- [20] Boldyreva A., Chenette N., Lee Y., O'Neill A. : Order-Preserving Symmetric Encryption. In: Joux A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 224-241. Springer, Heidelberg (2009)
- [21] Bao F., Deng R. H., Ding X., Yang Y.: Private Query on Encrypted Data in Multi-User Settings. In: Chen L., Mu Y., Susilo W. (eds.) ISPEC 2008. LNCS, vol. 4991, pp. 71-85. Springer, Heidelberg (2008)

- [22] Li J., Wang Q., Wang C., Cao N., Ren K., Lou W.: Fuzzy Keyword Search over Encrypted Data in Cloud Computing. In: IEEE INFO-COM 2010, pp. 1-5. (2010)
- [23] Waters B. R., Balfanz D., Durfee G., Smetters D. K.: Building an Encrypted and Searchable Audit Log. In: NDSS 2004 (2004)
- [24] Chase M., Kamara S.: Structured Encryption and Controlled Disclosure. In: M. Abe (ed.) Advances in Cryptology - ASIACRYPT 2010. LNCS, vol. 6477, pp. 577-594. Springer, Heidelberg (2010)
- [25] Kamara S., Papamanthou C., Roeder T.: Dynamic searchable symmetric encryption. In ACM Conference on Computer and Communications Security, pp. 965976 (2012)
- [26] Kamara S., Papamanthou C.: Parallel and Dynamic Searchable Symmetric Encryption. In: Sadeghi A.-R. (ed.) FC 2013. LNCS, vol.7859, pp. 258-274. Springer, Heidelberg (2013)
- [27] Cash D., Jaeger J., Jarecki S., Jutla C., Krawczyk H., Ros M.-C., Steiner M.: Dynamic Searchable Encryption in Very Large Databases: Data Structures and Implementation. In: NDSS 2014.
- [28] Abdalla M., Bellare M., Catalano D., Kiltz E., Kohno T., Lange T., Malone-Lee J., Neven G., Paillier P., Shi H.: Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions. In: Shoup V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 205-222. Springer, Heidelberg (2005)
- [29] Park D. J., Kim K., Lee P. J.: Public Key Encryption with Conjunctive Field Keyword Search. In: Lim C. H. and Yung M. (eds.) WISA 2004. LNCS, vol. 3325, pp. 73-86. Springer, Heidelberg (2004)
- [30] Golle P., Staddon J., Waters B. R.: Secure Conjunctive Keyword Search over Encrypted Data. In: Jakobsson M., Yung M., Zhou J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 31-45. Springer, Heidelberg (2004)
- [31] Ballard L., Kamara S., Monrose F.: Achieving Efficient Conjunctive Keyword Searches over Encrypted Data. In: Qing S. et al. (eds.) ICICS 2005. LNCS, vol. 3783, pp. 414-426. Springer, Heidelberg (2005)
- [32] Hwang Y. H., Lee P. J.: Public Key Encryption with Conjunctive Keyword Search and Its Extension to a Multi-user System. In: Takagi T., Okamoto T., Okamoto E. and Okamoto T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 2-22. Springer, Heidelberg (2007)
- [33] Ryu E.K., Takagi T.: Efficient Conjunctive Keyword-Searchable Encryption. In: 21st International Conference on Advanced Information Networking and Applications Workshops, pp. 409-414. IEEE (2007)
- [34] Baek J., Safavi-Naini R., Susilo W.: Public Key Encryption with Keyword Search Revisited. In: Gervasi O. (ed.) ICCSA 2008. LNCS, vol. 5072, pp. 1249-1259. Springer, Heidelberg (2008)
- [35] Bethencourt J., Chan T.-H. H., Perrig A., Shi E., Song D.: Anonymous Multi-Attribute Encryption with Range Query and Conditional Decryption. Technical Report CMU-CS-06-135 (2006)
- [36] Shi E., Bethencourt J., Chan T.-H. H., Song D., Perrig A.: Multi-Dimensional Range Query over Encrypted Data. In: IEEE S&P 2007, pp. 350-364. IEEE (2007)
- [37] Boneh D., Waters B. R.: Conjunctive, Subset, and Range Queries on Encrypted Data. In: Vadhan S. P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535-554. Springer, Heidelberg (2007)
- [38] Davis D., Monrose F., Reiter M. K.: Time-Scoped Searching of Encrypted Audit Logs. In: Lopez J., Qing S., Okamoto E. (eds.) ICICS 2004. LNCS, vol. 3269, pp. 532-545. Springer, Heidelberg (2004)
- [39] Cheung D. W., Mamoulis N., Wong W. K., Yiu S. M., Zhang Y.: Anonymous Fuzzy Identity-based Encryption for Similarity Search. In: Cheong O., Chwa K.-Y and Park K. (eds.) ISAAC 2010. LNCS, vol. 6505, pp. 61-72. Springer, Heidelberg (2010)
- [40] Camenisch J., Kohlweiss M., Rial A., Sheedy C.: Blind and Anonymous Identity-Based Encryption and Authorised Private Searches on Public Key Encrypted Data. In: Jarecki S. and Tsudik G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 196-214. Springer, Heidelberg (2009)
- [41] Bellare M., Fischlin M., O'Neill A., Ristenpart T.: Deterministic Encryption: Definitional Equivalences and Constructions without Random Oracles. In: Wagner D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 360-378. Springer, Heidelberg (2008)
- [42] Boldyreva A., Fehr S., O'Neill A. : On Notions of Security for Deterministic Encryption, and Efficient Constructions without Random Oracles. In: Wagner D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 335-359. Springer, Heidelberg (2008)
- [43] Brakerski Z., Segev G.: Better Security for Deterministic Public-Key Encryption: The Auxiliary-Input Setting. In: Rogaway P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 543-560. Springer, Heidelberg (2011)
- [44] Menezes A. J., Okamoto T., Vanstone S. A.: Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field. IEEE Transactions on Information Theory, 39(5), pp. 1639-1646 (1993)

- [45] Frey G., Muller M., Ruck H.-G.: The Tate Pairing and the Discrete Logarithm Applied to Elliptic Curve Cryptosystems. IEEE Transactions on Information Theory, vol. 45, no. 5, pp. 1717-1719 (1999)
- [46] Abdalla M., Bellare M., Neven G.: Robust encryption. In: Micciancio G. (ed.) TCC 2010. LNCS, vol. 5978, pp. 480497. Springer, Heidelberg (2010)
- [47] Izabachène M., Pointcheval D.: New Anonymity Notions for Identity-Based Encryption. In: Ostrovsky R., De Prisco R. and Visconti I. (eds.) SCN 2008. LNCE, vol. 5229, pp. 375-391. Springer, Heidelberg (2008)
- [48] Waters B.: Efficient Identity-Based Encryption Without Random Oracles. In: Cramer R. (ed.), Advances in Cryptology - EUROCRYPT 2005. LNCS, vol. 3494, pp. 1-17. Springer, Heidelberg (2005)
- [49] Tang Q., Chen X.: Towards asymmetric searchable encryption with message recovery and flexible search authorization. ASIACCS 2013, pp. 253-264 (2013)
- [50] Ibraimi L., Nikova S., Hartel P. H., Jonker W.: Public-Key Encryption with Delegated Search. In: Lopez J. and Tsudik G. (eds.) ACNS 2011. LNCS, vol. 6715, pp. 532-549. Springer, Heidelberg (2011)
- [51] Yang G., Tan C. H., Huang Q., Wong D. S.: Probabilistic Public Key Encryption with Equality Test. In: Pieprzyk J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 119-131. Springer, Heidelberg (2010)
- [52] Xu P., Jin H., Wu Q., Wang W.: Public-Key Encryption with Fuzzy Keyword Search: A Provably Secure Scheme under Keyword Guessing Attack. IEEE Transactions on Computers, 62(11), pp. 2266-2277 (2013)
- [53] Arriaga A., Tang Q., Ryan P.: Trapdoor Privacy in Asymmetric Searchable Encryption Schemes. In: Pointcheval D. and Vergnaud D. (eds.) AFRICACRYPT 2014. LNCS, vol. 8469, pp. 31-50. Springer, Heidelberg (2014)



**Peng Xu** received the B.A. degree in computer science from Wuhan university of science and technique, Wuhan, China, in 2003, the Master and Ph.D. degree in computer science from Huazhong university of science and technology, Wuhan, China, respectively in 2006 and 2010. Since 2010, he works as a post-doctor at Huazhong university of science and technology, Wuhan, China. He was PI in three grants respectively from National Natural Science Foundation of China (No. 61472156 and No. 61100222) and

China Postdoctoral Science Foundation (No. 20100480900), and a key member in several projects supported by 973 (No. 2014CB340600). He has authored over 20 research papers. He is a member of ACM and IEEE.



**Qianhong Wu** received his Ph.D. in Cryptography from Xidian University in 2004. Since then, he has been with Wollongong University (Australia) as an associate research fellow, with Wuhan University (China) as an associate professor, with Universitat Rovira i Virgili (Catalonia) as a research director and now with Beihang University (China) as a full professor. His research interests include cryptography, information security and privacy, and *ad hoc* network security. He has been a holder/co-holder of 7

China/Australia/Spain funded projects. He has authored 7 patents and over 100 publications. He has served in the program committee of several international conferences in information security and privacy. He is a member of IACR, ACM and IEEE.



Wei Wang received the B.S. and Ph.D. degrees in Electronic and Communication Engineering from Huazhong University of Science and Technology, Wuhan, China, in 2006 and 2011, respectively. Currently she is a researcher with Cyber-Physical-Social Systems Lab, Huazhong University of Science and Technology, Wuhan, China. She was a Post-doctoral researcher with Peking University, Beijing, China from April 2012 to July 2014. Her research interests include cloud security, network coding and multimedia

transmission. She has published over 10 papers in international journals and conferences.



Josep Domingo-Ferrer is a Distinguished Professor of Computer Science and an ICREA-Acadèmia Researcher at Universitat Rovira i Virgili, Tarragona, Catalonia, where he holds the UNESCO Chair in Data Privacy. His research interests are in data privacy and data security. He received his M. Sc. and Ph. D. degrees in Computer Science from the Autonomous University of Barcelona in 1988 and 1991, respectively. He also holds an M. Sc. in Mathematics. He has won several research and technology transfer awards,

including twice the ICREA Academia Prize (2008 and 2013) and the "Narcís Monturiol" Medal to the Scientific Merit, both awarded by the Government of Catalonia, and a Google Faculty Research Award (2014). He has authored 5 patents and over 350 publications. He has been the co-ordinator of projects funded by the European Union and the Spanish government. He has been the PI of US-funded research contracts and currently of a Templeton World Charity Foundation grant. He has held visiting appointments at Princeton, Leuven and Rome. He is a co-Editor-in-Chief of *Transactions on Data Privacy*. He is an IEEE Fellow and an Elected Member of Academia Europaea.



Hai Jin received his PhD in computer engineering from HUST in 1994. In 1996, he was awarded a German Academic Exchange Service fellowship to visit the Technical University of Chemnitz in Germany. He worked at The University of Hong Kong between 1998 and 2000, and as a visiting scholar at the University of Southern California between 1999 and 2000. He was awarded Excellent Youth Award from the National Science Foundation of China in 2001. He is the chief scientist of National 973 Basic

Research Program Project of Virtualization Technology of Computing System. He has co-authored 15 books and published over 400 research papers. He is a senior member of the IEEE and a member of the ACM.

## A. Analysis on The Work [40]

A sender generates the searchable ciphertexts of any keyword  $W_i \in W$  by the following steps: 1) The first time to encrypt keyword  $W_i$ , he uploads

 $PEKS(Pub, W_i, K_i^1 || P_i^1), P_i^1 || E(K_i^1, P_i^2 || K_i^2 || \mathcal{P}_i^1), P_i^2$ 

to the server, and asks the server to store  $E(K_i^1, P_i^2 || K_i^2 || P_i^1)$  in position  $P_i^1$  and store a flag in position  $P_i^2$ .

Note: algorithm  $PEKS(Pub, W_i^1, K_i^1 || P_i^1) = IBE(Pub, W_i^1, K_i^1 || P_i^1 || C_2) || C_2$  takes public parameter Pub, identity  $W_i^1$  and plaintext  $K_i^1 || P_i^1 || C_2$  as inputs and generates an IBE ciphertext, and finally outputs the IBE ciphertext and  $C_2$ , where the symmetric key  $K_i^1$  and  $C_2$  are randomly chosen. Algorithm  $E(K_i^1, P_i^2 || K_i^2 || P_i^1)$  denotes using the symmetric key  $K_i^1$  to encrypt  $P_i^2 || K_i^2 || P_i^1$ , where the symmetric key  $K_i^2$  is randomly chosen, and  $\mathcal{P}_i^1$  denotes the parameters for private information retrieval (they will be used to retrieve the corresponding data when the keyword  $W_i$  is queried).

- 2) The second time to encrypt keyword  $W_i$ , he uploads  $P_i^2 ||E(K_i^2, P_i^3||K_i^3||\mathcal{P}_i^2), P_i^3$  to the server, and asks the server to store  $E(K_i^2, P_i^3||K_i^3||\mathcal{P}_i^2)$  in position  $P_i^2$  and store the flag in position  $P_i^3$ .
- 3) The subsequent encryptions of keyword  $W_i$  are similar to Step 2.

Figure 5: Procedure to generate keyword searchable ciphertexts in [40].

In Fig. 5, we first review how to generate keyword-searchable ciphertexts according to [40] such that the ciphertexts of the same keyword form a chain. Then we analyze why the chain of any keyword is visible in the view of the server, and give a straightforward method to make the chain invisible. But this method seems to be impractical.

According to the first step in Fig. 5, the server trivially knows the relation between ciphertexts  $PEKS(Pub, W_i, K_i^1 || P_i^1)$  and  $E(K_i^1, P_i^2 || K_i^2 || P_i^1)$ , and knows that if a subsequent ciphertext is stored in the position  $P_2$ , this subsequent ciphertext is related to  $E(K_i^1, P_i^2 || K_i^2 || P_i^1)$ . So in the second step, the server knows the relation between ciphertexts  $E(K_i^1, P_i^2 || K_i^2 || P_i^1)$  and  $E(K_i^2, P_i^3 || K_i^3 || P_i^2)$ , and knows that if another subsequent ciphertext is stored in the position  $P_3$ , this subsequent ciphertext is related to  $E(K_i^2, P_i^3 || K_i^3 || P_i^2)$ . By the same method, the server will know the chain of keyword  $W_i$  even without the keyword search trapdoor of keyword  $W_i$ . Furthermore, the length of the chain leaks the frequency of keyword  $W_i$ .

- A sender generates the searchable ciphertexts of any keyword  $W_i \in W$  by the following steps:
  - 1) At the setup phase, he uploads  $\{PEKS(Pub, W_i, K_i^1 || P_i^1) | i \in [1, |W|]\}$  to the server, where |W| denotes the size of keyword space W.
  - 2) The first time to encrypt keyword  $W_i$ , he uploads  $P_i^1 || E(K_i^1, P_i^2) || K_i^2 || \mathcal{P}_i^1)$  to the server, and asks the server to store  $E(K_i^1, P_i^2) || K_i^2 || \mathcal{P}_i^1)$  in position  $P_i^1$ .
  - 3) The second time to encrypt keyword  $W_i$ , he uploads  $P_i^2 ||E(K_i^2, P_i^3||K_i^3||\mathcal{P}_i^2)$  to the server, and asks the server to store  $E(K_i^2, P_i^3||K_i^3||\mathcal{P}_i^2)$  in position  $P_i^2$ .
  - 4) The subsequent encryptions of keyword  $W_i$  are similar to Step 3.

Figure 6: New procedure to generate keyword-searchable ciphertexts for [40].

In order to keep the privacy of the chain, a straightforward method is to generate the PEKS ciphertexts for all keywords at the setup phase and delete the flag. The specific procedure is given in Fig. 6. This method hides the relation between the PEKS ciphertext and the symmetric-key ciphertext of any keyword, and the relation between two symmetric-key ciphertexts of any keyword also is hidden. But it seems that this method is impractical from a performance viewpoint, since each sender must generate the PEKS ciphertexts for all keywords at the setup phase and remember lots of private information which are encrypted by these PEKS ciphertexts.

# B. Proof of Theorem 1

*Proof:* Without loss of generality, it is sufficient to prove that given the keyword-searchable trapdoor  $T_{W_i} = H(W_i)^s$  of keyword  $W_i$  and the hidden structure's public part  $\mathbf{Pub} = g^u$ , algorithm **StructuredSearch**(**PK**, **Pub**,  $\mathbb{C}, T_{W_i}$ ) only finds out all ciphertexts of keyword  $W_i$  with the hidden structure **Pub**. Note that  $P = g^s$ .

Algorithm StructuredSearch(PK, Pub,  $\mathbb{C}, T_{W_i}$ ) computes  $Pt' = \hat{e}(\mathbf{Pub}, T_{W_i})$  in its first step. Since  $\hat{e}(\mathbf{Pub}, T_{W_i}) = \hat{e}(P, H(W_i))^u$ , algorithm StructuredSearch(PK, Pub,  $\mathbb{C}, T_{W_i}$ ) finds out the ciphertext  $(\hat{e}(P, H(W_i))^u, g^r, \hat{e}(P, H(W_i))^r \cdot Pt[u, W_i])$  by matching Pt' with all ciphertexts' first part in its second step. Moreover,

due to the collision-freeness of hash function H, only keyword  $W_i$  has  $Pt' = \hat{e}(P, H(W_i))^u$ , except with a negligible probability in the security parameter k. So only the ciphertext  $(\hat{e}(P, H(W_i))^u, q^r, \hat{e}(P, H(W_i))^r \cdot Pt[u, W_i])$  is found with overwhelming probability in this step.

algorithm **StructuredSearch**(**PK**, **Pub**,  $\mathbb{C}$ ,  $T_{W_i}$ ) discloses  $Pt[u, W_i]$  from the Then ciphertext  $(\hat{e}(P, H(W_i))^u, g^r, \hat{e}(P, H(W_i))^r \cdot Pt[u, W_i])$  by computing  $Pt' = Pt[u, W_i] = \hat{e}(g^r, T_{W_i})^{-1} \cdot \hat{e}(P, H(W_i))^r \cdot Pt[u, W_i].$ 

Recall that in algorithm StructuredEncryption,  $Pt[u, W_i]$  was randomly chosen in  $\mathbb{G}_1$  and taken as the first part of only one ciphertext of keyword  $W_i$  with the hidden structure **Pub**. So when algorithm **StructuredSearch**(**PK**, **Pub**,  $\mathbb{C}$ ,  $T_{W_i}$ ) goes back to its second step, only the ciphertext  $(Pt[u, W_i], q^r, \hat{e}(P, H(W_i))^r)$ R) is found with overwhelming probability.

By carrying on in this way, algorithm StructuredSearch(PK, Pub,  $\mathbb{C}, T_{W_i}$ ) only finds out all ciphertexts of keyword  $W_i$  with the hidden structure **Pub**, except with a negligible probability in the security parameter k. And the algorithm will stop, since the random value R contained in the last found ciphertext does not match any other ciphertext's first part.

# C. Proof of Theorem 2

*Proof:* To prove this theorem, we will construct a PPT algorithm  $\mathcal{B}$  that plays the SS-CKSA game with adversary  $\mathcal{A}$  and utilizes the capability of  $\mathcal{A}$  to solve the DBDH problem in **BGen** $(1^k)$  with advantage approximately  $\frac{27}{(e \cdot q_t \cdot q_p)^3}$ .  $Adv_{SPCHS,\mathcal{A}}^{\text{SS-CKSA}}$ . Let  $Coin \leftarrow \{0,1\}$  denote the operation that picks  $Coin \in \{0,1\}$  according to the probability  $Pr[Coin = Coin \leftarrow Co$ 1] =  $\sigma$  (the specified value of  $\sigma$  will be decided latter). The constructed algorithm  $\mathcal{B}$  in the SS-CKSA game is as follows.

- Setup Phase: Algorithm  $\mathcal{B}$  takes as inputs  $(q, \mathbb{G}, \mathbb{G}_1, g, \hat{e}, g^a, g^b, g^c, Z)$  (where Z equals either  $\hat{e}(q, g)^{abc}$  or  $\hat{e}(q, g)^y$ ) and the keyword space W, and performs the following steps:
  - 1) Initialize the three lists  $\mathbf{Pt} = \emptyset \subseteq \mathcal{W} \times \mathbb{G} \times \mathbb{G}_1$ ,  $\mathbf{SList} = \emptyset \subseteq \mathbb{G} \times \mathbb{Z}_a^* \times \{0,1\}$  and  $\mathbf{HList} = \emptyset \subseteq \mathbb{G}$  $\mathcal{W} \times \mathbb{G} \times \mathbb{Z}_a^* \times \{0, 1\};$
  - 2) Set the ciphertext space  $C = \mathbb{G}_1 \times \mathbb{G} \times \mathbb{G}_1$  and  $\mathbf{PK} = (q, \mathbb{G}, \mathbb{G}_1, g, \hat{e}, P = g^a, W, C)$ ;
  - 3) Initialize N hidden structures by repeating the following steps for  $i \in [1, N]$ :

    - a) Pick u<sub>i</sub> ← Z<sub>q</sub><sup>\*</sup> and Coin<sub>i</sub> ← {0,1};
      b) If Coin<sub>i</sub> = 1, compute Pub<sub>i</sub> = g<sup>b·u<sub>i</sub></sup>;
    - c) Otherwise, compute  $\mathbf{Pub}_i = q^{u_i}$ ;
  - 4) Set  $\mathbf{PSet} = {\{\mathbf{Pub}_i | i \in [1, N]\}}$  and  $\mathbf{SList} = {\{(\mathbf{Pub}_i, u_i, Coin_i) | i \in [1, N]\}};$
  - 5) Send **PK** and **PSet** to adversary A.
- Query Phase 1: Adversary  $\mathcal{A}$  adaptively issues the following queries multiple times.
  - Hash Query  $\mathcal{Q}_H(W)$ : Taking as input a keyword  $W \in \mathcal{W}$ , algorithm  $\mathcal{B}$  does the following steps:
    - 1) Pick  $x \stackrel{\$}{\leftarrow} \mathbb{Z}_q^*$  and  $Coin \stackrel{\sigma}{\leftarrow} \{0, 1\}$ ;
    - 2) If Coin = 0, add  $(W, g^x, x, Coin)$  into **HList** and output  $g^x$ ;
    - 3) Otherwise, add  $(W, g^{c \cdot x}, x, Coin)$  into **HList** and output  $g^{c \cdot x}$ ;
  - Trapdoor Query  $\mathcal{Q}_{Trap}(W)$ : Taking as input a keyword  $W \in \mathcal{W}$ , algorithm  $\mathcal{B}$  does the following steps:
    - 1) If  $(W, *, *, *) \notin \mathbf{HList}$ , query  $\mathcal{Q}_H(W)$ ;
    - 2) According to W, retrieve (W, X, x, Coin) from **HList**;
    - 3) If Coin = 0, output  $q^{a \cdot x}$ ; otherwise abort and output  $\perp$ ;
  - Privacy Query  $Q_{Pri}(\mathbf{Pub})$ : Taking as input a structure's public part  $\mathbf{Pub} \in \mathbf{PSet}$ , algorithm  $\mathcal{B}$  does the following steps:
    - 1) According to **Pub**, retrieve (**Pub**, *u*, *Coin*) from **SList**;
    - 2) If Coin = 0, output u; otherwise abort and output  $\bot$ ;
  - Encryption Query  $\mathcal{Q}_{Enc}(W, \mathbf{Pub})$ : Taking as inputs a keyword  $W \in \mathcal{W}$  and a structure's public part  $\mathbf{Pub}$ , algorithm  $\mathcal{B}$  does the following steps:
    - 1) If  $(W, *, *, *) \notin \mathbf{HList}$ , query  $\mathcal{Q}_H(W)$ ;
    - 2) According to W and Pub, retrieve (W, X, x, Coin) and (Pub, u, Coin') respectively from HList and SList;
    - 3) Pick  $r \stackrel{\$}{\leftarrow} \mathbb{Z}_q^*$ , and search  $(W, \mathbf{Pub}, Pt[u, W])$  for W and Pub in Pt;
    - 4) If W is not found, insert  $(W, \mathbf{Pub}, Pt[u, W] \stackrel{\$}{\leftarrow} \mathbb{G}_1)$  to **Pt** and do the following steps:
      - a) If  $Coin = 1 \bigwedge Coin' = 1$ , output  $C = (Z^{x \cdot u}, g^r, \hat{e}(g^a, X)^r \cdot Pt[u, W]);$
      - b) If  $Coin = 0 \bigwedge Coin' = 1$ , output  $C = (\hat{e}(g^a, g^{b \cdot u})^x, g^r, \hat{e}(g^a, X)^r \cdot Pt[u, W]);$
      - c) If Coin' = 0, output  $C = (\hat{e}(g^a, X)^u, g^r, \hat{e}(g^a, X)^r \cdot Pt[u, W]);$
    - 5) Otherwise, pick  $R \stackrel{\$}{\leftarrow} \mathbb{G}_1$ , set  $C = (Pt[u, W], q^r, \hat{e}(q^a, X)^r \cdot R)$ , update Pt[u, W] = R and output C;

- Challenge Phase: Adversary  $\mathcal{A}$  sends two challenge keyword-structure pairs  $(W_0^*, \mathbf{Pub}_0^*) \in \mathcal{W} \times \mathbf{PSet}$  and  $(W_1^*, \mathbf{Pub}_1^*) \in \mathcal{W} \times \mathbf{PSet}$  to algorithm  $\mathcal{B}$ ;  $\mathcal{B}$  picks  $d \stackrel{\$}{\leftarrow} \{0, 1\}$ , and does the following steps:
  - 1) According to  $\mathbf{Pub}_0^*$  and  $\mathbf{Pub}_1^*$ , retrieve  $(\mathbf{Pub}_0^*, u_0^*, Coin_0^*)$  and  $(\mathbf{Pub}_1^*, u_1^*, Coin_1^*)$  from SList; and if  $Coin_0^* = 0 \bigvee Coin_1^* = 0$ , then abort and output  $\perp$ ;
  - 2) If  $(W_d^*, *, *, *) \notin \mathbf{HList}$ , query  $\mathcal{Q}_H(W_d^*)$ ;
  - 3) According to  $W_d^*$ , retrieve  $(W_d^*, X_d^*, x_d^*, Coin)$  from **HList**; and if Coin = 0, then abort and output  $\perp$ ;
  - 4) Search  $(W_d^*, \mathbf{Pub}_d^*, Pt[u_d^*, W_d^*])$  for  $W_d^*$  and  $\mathbf{Pub}_d^*$  in  $\mathbf{Pt}$ ;
  - 5) If it is not found, insert  $(W_d^*, \mathbf{Pub}_d^*, Pt[u_d^*, W_d^*] \stackrel{\$}{\leftarrow} \mathbb{G}_1)$  to  $\mathbf{Pt}$ , and send  $C_d^* = (Z^{x_d^* \cdot u_d^*}, g^b, Z^{x_d^*} \cdot Pt[u_d^*, W_d^*])$  to adversary  $\mathcal{A}$ ;
  - 6) Otherwise, pick  $R \stackrel{\$}{\leftarrow} \mathbb{G}_1$ , set  $C_d^* = (Pt[u_d^*, W_d^*], g^b, Z^{x_d^*} \cdot R)$ , update  $Pt[u_d^*, W_d^*] = R$ , and send  $C_d^*$  to adversary  $\mathcal{A}$ ;
- Query Phase 2: This phase is the same as Query Phase 1. Note that in Query Phase 1 and Query Phase 2, adversary  $\mathcal{A}$  cannot query the corresponding private parts both of  $\mathbf{Pub}_0^*$  and  $\mathbf{Pub}_1^*$  and the keyword search trapdoors both of  $W_0^*$  and  $W_1^*$ .
- Guess Phase: Adversary A sends a guess d' to algorithm B. If d = d', B output 1; otherwise, output 0.

Let  $\overline{Abort}$  denote the event that algorithm  $\mathcal{B}$  does not abort in the above game. Next, we will compute the probabilities  $Pr[\overline{Abort}]$ ,  $Pr[\mathcal{B} = 1|Z = \hat{e}(g,g)^{abc}]$  and  $Pr[\mathcal{B} = 1|Z = \hat{e}(g,g)^y]$ , and the advantage  $Adv_{\mathcal{B}}^{DBDH}(1^k)$ .

According to the above game, the probability of the event  $\overline{Abort}$  only relies on the probability  $\sigma$  and the number of times that adversary  $\mathcal{A}$  queries oracles  $\mathcal{Q}_{Trap}(\cdot)$  and  $\mathcal{Q}_{Pri}(\cdot)$ . We have that  $Pr[\overline{Abort}] = (1 - \sigma)^{q_t \cdot q_p} \cdot \sigma^3$ . Let  $\sigma = \frac{3}{3+q_t \cdot q_p}$ . We have that  $Pr[\overline{Abort}] \approx \frac{27}{(e \cdot q_t \cdot q_p)^3}$ , where e is the base of natural logarithms.

When  $Z = \hat{e}(g,g)^{abc}$  and the event  $\overline{Abort}$  holds, it is easy to find that algorithm  $\mathcal{B}$  simulates a real SS-CKSA game in adversary  $\mathcal{A}$ 's mind. So we have

$$Pr[d = d' | \overline{Abort} \bigwedge Z = \hat{e}(g, g)^{abc}] = (Adv_{SPCHS, \mathcal{A}}^{SS-CKSA} + \frac{1}{2}).$$

When  $Z = \hat{e}(g, g)^y$  and the event *Abort* holds, algorithm  $\mathcal{B}$  generates a challenge ciphertext, which is independent of the challenge keywords  $W_0^*$  and  $W_1^*$ . So we have

$$Pr[d = d' | \overline{Abort} \bigwedge Z = \hat{e}(g, g)^y] = \frac{1}{2}.$$

Now, we can compute the advantage  $Adv_{\mathcal{B}}^{DBDH}(1^k)$  as follows:

$$\begin{split} \operatorname{Adv}_{\mathcal{B}}^{DBDH}(1^k) &= \Pr[\mathcal{B} = 1 | Z = \hat{e}(g,g)^{abc}] - \Pr[\mathcal{B} = 1 | Z = \hat{e}(g,g)^y] \\ &= \Pr[d = d' \bigwedge \overline{\operatorname{Abort}} | Z = \hat{e}(g,g)^{abc}] - \Pr[d = d' \bigwedge \overline{\operatorname{Abort}} | Z = \hat{e}(g,g)^y] \\ &= \Pr[d = d' | \overline{\operatorname{Abort}} \bigwedge Z = \hat{e}(g,g)^{abc}] \cdot \Pr[\overline{\operatorname{Abort}} | Z = \hat{e}(g,g)^{abc}] \\ &- \Pr[d = d' | \overline{\operatorname{Abort}} \bigwedge Z = \hat{e}(g,g)^y] \cdot \Pr[\overline{\operatorname{Abort}} | Z = \hat{e}(g,g)^y] \\ &\approx (\operatorname{Adv}_{SPCHS,\mathcal{A}}^{SS-CKSA} + \frac{1}{2}) \cdot \frac{27}{(e \cdot q_t \cdot q_p)^3} - \frac{1}{2} \cdot \frac{27}{(e \cdot q_t \cdot q_p)^3} \\ &\approx \frac{27}{(e \cdot q_t \cdot q_p)^3} \cdot \operatorname{Adv}_{SPCHS,\mathcal{A}}^{SS-CKSA} \end{split}$$

In addition, it is clear that algorithm  $\mathcal{B}$  is a PPT algorithm, if adversary  $\mathcal{A}$  is a PPT adversary. In conclusion, if a PPT adversary  $\mathcal{A}$  wins in the SS-CKSA game of the above SPCHS instance with advantage  $Adv_{SPCHS,\mathcal{A}}^{SS-CKSA}$ , in which  $\mathcal{A}$ makes at most  $q_t$  queries to oracle  $\mathcal{Q}_{Trap}(\cdot)$  and at most  $q_p$  queries to oracle  $\mathcal{Q}_{Pri}(\cdot)$ , then there is a PPT algorithm  $\mathcal{B}$ that solves the DBDH problem in **BGen**(1<sup>k</sup>) with advantage approximately

$$Adv_{\mathcal{B}}^{DBDH}(1^k) \approx \frac{27}{(e \cdot q_t \cdot q_p)^3} \cdot Adv_{SPCHS,\mathcal{A}}^{\text{SS-CKSA}}$$

where e is the base of natural logarithms.

#### D. Proof of Theorem 3

ŀ

**Proof:** Without loss of generality, it is sufficient to prove that given the keyword-searchable trapdoor  $T_{W_i} = (\hat{S}_{W_i}, \tilde{S}_{W_i})$  of keyword  $W_i$  and the hidden structure's public part  $\mathbf{Pub} = \hat{C}$ , algorithm **StructuredSearch**( $\mathbf{PK}, \mathbf{Pub}, \mathbb{C}, T_{W_i}$ ) only finds out all ciphertexts of keyword  $W_i$  with the hidden structure  $\mathbf{Pub}$ , where  $\hat{S}_{W_i} = \mathbf{Extract}_{\mathrm{IBKEM}}(\mathbf{SK}_{\mathrm{IBKEM}}, W_i)$ ,  $\tilde{S}_{W_i} = \mathbf{Extract}_{\mathrm{IBE}}(\mathbf{SK}_{\mathrm{IBE}}, W_i)$ ,  $\hat{C}$  is from  $(\hat{K}, \hat{C}) =$ **Encaps**<sub>IBKEM</sub>( $\mathbf{PK}_{\mathrm{IBKEM}}, W, u$ ), keyword W is arbitrarily chosen in  $\mathcal{W}$ , and u is a random value. Algorithm StructuredSearch(PK, Pub,  $\mathbb{C}$ ,  $T_{W_i}$ ) computes  $Pt' = \text{Decaps}_{\text{IBKEM}}(\hat{S}_{W_i}, \text{Pub})$  in its first step. According to the full-identity malleability of IBKEM in Definition 7, we have  $\text{FIM}(W_i, u) =$  $\text{Decaps}_{\text{IBKEM}}(\hat{S}_{W_i}, \text{Pub})$ . So algorithm StructuredSearch(PK, Pub,  $\mathbb{C}$ ,  $T_{W_i}$ ) finds out the ciphertext (FIM( $W_i, u$ ),  $\text{Enc}_{\text{IBE}}(\text{PK}_{\text{IBE}}, W_i, Pt[u, W_i])$ ) by matching Pt' with all ciphertexts' first part in its second step. Moreover, due to the collision-freeness of IBKEM in Definition 7, there is no keyword  $W_j \equiv W_i$ ) to meet  $\text{FIM}(W_i, u) = \text{FIM}(W_j, u)$ , and no hidden structure  $\text{Pub}' \equiv equiv equiv equiv equiv equiv (<math>W_i, u$ ) =  $\text{FIM}(W_i, u')$ , where Pub' is generated by algorithm StructureInitialization(PK) with the random value u'. So only the ciphertext ( $\text{FIM}(W_i, u), \text{Enc}_{\text{IBE}}(\text{PK}_{\text{IBE}}, W_i, Pt[u, W_i])$ ) is found in this step, except with a negligible probability in the security parameter k. Then, according to the consistency of IBE, algorithm StructuredSearch(PK, Pub,  $\mathbb{C}, T_{W_i}$ ) can decrypt  $Pt[u, W_i]$  by algorithm  $\text{Dec}_{\text{IBE}}(\tilde{S}_{W_i}, \text{Enc}_{\text{IBE}}(\text{PK}_{\text{IBE}}, W_i, Pt[u, W_i])$ ).

Recall that in algorithm **StructuredEncryption**,  $Pt[u, W_i]$  was randomly chosen in  $\mathbb{G}_1$  and taken as the first part of only one ciphertext of keyword  $W_i$ . So when **StructuredSearch**(**PK**, **Pub**,  $\mathbb{C}$ ,  $T_{W_i}$ ) goes back to its second step, only the ciphertext ( $Pt[u, W_i]$ , **Enc**<sub>IBE</sub>(**PK**<sub>IBE</sub>,  $W_i, R$ )) is found, except with a negligible probability in the security parameter k.

By carrying on in the same way, algorithm **StructuredSearch**( $\mathbf{PK}, \mathbf{Pub}, \mathbb{C}, T_{W_i}$ ) only finds out all ciphertexts of keyword  $W_i$  with the hidden structure  $\mathbf{Pub}$ , except with a negligible probability in the security parameter k. And the algorithm will stop, since the random value R contained in the last found ciphertext of keyword  $W_i$  fails to match any other ciphertext's first part.

# E. Proof of Theorem 4

*Proof:* Let  $\mathcal{G}_1$  and  $\mathcal{G}_2$  be the challengers respectively in the Anon-SS-sID-CPA game of the underlying IBKEM scheme and the Anon-SS-ID-CPA game of the underlying IBE scheme. A constructed adversary  $\mathcal{B}$  in the SS-sK-CKSA game of the generic SPCHS construction is as follows.

- Setup Phase: In this phase,
  - 1)  $\mathcal{A}$  sends two challenge keywords  $(W_0^*, W_1^*)$  to  $\mathcal{B}$ .
  - 2)  $\mathcal{B}$  arbitrarily picks  $I_1^* \leftarrow (\mathcal{ID}_{\text{IBKEM}} \mathcal{W})$ , and sends two challenge identities  $(W_0^*, I_1^*)$  to  $\mathcal{G}_1$ . (The  $I_1^*$  is existing, since we have  $\mathcal{W} \subset \mathcal{ID}_{\text{IBKEM}}$ .)
  - 3)  $\mathcal{G}_1$  generates (**PK**<sub>IBKEM</sub>, **SK**<sub>IBKEM</sub>) by algorithm **Setup**<sub>IBKEM</sub> and sends **PK**<sub>IBKEM</sub> to  $\mathcal{B}$ .
  - 4)  $\mathcal{B}$  queries  $\mathcal{G}_1$  for the challenge key-and-encapsulation pair.
  - 5)  $\mathcal{G}_1$  picks  $\hat{d} \stackrel{\$}{\leftarrow} \{0,1\}$ , generates  $(\hat{K}_0^*, \hat{C}_0^*) = \mathbf{Encaps}_{\mathsf{IBKEM}}(\mathbf{PK}_{\mathsf{IBKEM}}, W_0^*, r_0)$  and  $(\hat{K}_1^*, \hat{C}_1^*) = \mathbf{Encaps}_{\mathsf{IBKEM}}(\mathbf{PK}_{\mathsf{IBKEM}}, I_1^*, r_1)$ , and sends  $(\hat{K}_d^*, \hat{C}_0^*)$  to  $\mathcal{B}$ , where  $r_0$  and  $r_1$  are randomly chosen.
  - 6)  $\mathcal{B}$  adds  $\hat{C}_0^*$  into the set  $\mathbf{PSet} \subseteq \mathcal{C}_{\text{IBKEM}}$ .
  - 7)  $\mathcal{G}_2$  generates ( $\mathbf{PK}_{\text{IBE}}, \mathbf{SK}_{\text{IBE}}$ ) by algorithm  $\mathbf{Setup}_{\text{IBE}}$ , and sends  $\mathbf{PK}_{\text{IBE}}$  to  $\mathcal{B}$ .
  - 8)  $\mathcal{B}$  initializes the two lists  $\mathbf{SList} = \emptyset \subseteq \mathcal{C}_{\text{IBKEM}} \times \{0, 1\}^*$  and  $\mathbf{Pt} = \emptyset \subseteq \mathcal{W} \times \mathcal{C}_{\text{IBKEM}} \times \mathcal{M}_{\text{IBE}}$ , and initializes N-1 hidden structures by repeating the following steps for  $i \in [1, N-1]$ :
    - a) Pick a random value  $u_i$  and an arbitrary keyword  $W_i \in \mathcal{W}$ ;
    - b) Generate  $(\hat{K}_i, \hat{C}_i) = \text{Encaps}_{\text{IBKEM}}(\mathbf{PK}_{\text{IBKEM}}, W_i, u_i)$ , add  $\mathbf{Pub}_i = \hat{C}_i$  into the set  $\mathbf{PSet}$ , and add  $(\mathbf{Pub}_i, u_i)$  into  $\mathbf{SList}$ ;
  - 9)  $\mathcal{B}$  finally sends **PK** and **PSet** to  $\mathcal{A}$ .
- Query Phase 1: In this phase, adversary A adaptively issues the following queries multiple times.
  - Trapdoor Query  $Q_{Trap}(W)$ : Taking as input a keyword  $W \in W$ ,  $\mathcal{B}$  forwards the query W both to the decryption key oracles  $\hat{S}_W = Q_{DK}^{IBKEM}(W)$  and  $\tilde{S}_W = Q_{DK}^{IBE}(W)$ , and sends  $T_W = (\hat{S}_W, \tilde{S}_W)$  to  $\mathcal{A}$ .

(In this query,  $\mathcal{A}$  cannot query the keyword search trapdoor corresponding to the challenge keyword  $W_0^*$  or  $W_1^*$ . In addition, one may find that  $\mathcal{B}$  cannot respond the query  $\mathcal{Q}_{Trap}(I_1^*)$ . However, this is not a problem, since we let  $I_1^* \in (\mathcal{ID}_{\text{IBKEM}} - \mathcal{W})$ . So  $\mathcal{A}$  never issues that query.)

- Privacy Query  $Q_{Pri}(\mathbf{Pub})$ : Taking as input a structure's public part  $\mathbf{Pub} \in \mathbf{PSet}$ ,  $\mathcal{B}$  aborts and outputs  $\perp$  if  $\mathbf{Pub} = \hat{C}_0^*$ ; otherwise,  $\mathcal{B}$  retrieves ( $\mathbf{Pub}, u$ ) from SList according to  $\mathbf{Pub}$  and outputs u.
- Encryption Query  $Q_{Enc}(W, \mathbf{Pub})$ : Taking as inputs a keyword  $W \in W$  and a structure's public part  $\mathbf{Pub}$ ,  $\mathcal{B}$  does the following steps:
  - 1) If  $\mathbf{Pub} = \hat{C}_0^* \bigwedge W \neq W_0^*$ , then
    - a) Search (W, Pub, Pt[u\*, W]) for W and Pub in Pt;
       (Note that u\* is not a really known value. It is just a symbol to denote the random value used to generate Pub = Ĉ\_0\*.)
    - b) If it is not found, query  $\hat{S}_W = \mathcal{Q}_{DK}^{IBKEM}(W)$ , insert  $(W, \mathbf{Pub}, Pt[u^*, W] \stackrel{\$}{\leftarrow} \mathcal{M}_{\scriptscriptstyle \mathrm{IBE}})$  to  $\mathbf{Pt}$  and output  $C = (\mathbf{Decaps}_{\scriptscriptstyle \mathrm{IBKEM}}(\hat{S}_W, \mathbf{Pub}), \mathbf{Enc}_{\scriptscriptstyle \mathrm{IBE}}(\mathbf{PK}_{\scriptscriptstyle \mathrm{IBE}}, W, Pt[u^*, W]));$

(Note that when  $W = W_1^*$ ,  $\mathcal{B}$  still can query  $\hat{S}_W = \mathcal{Q}_{DK}^{IBKEM}(W)$ , since  $W_1^*$  is not a challenge IBKEM identity in the above **Setup Phase**.)

- c) Otherwise, pick  $R \stackrel{\$}{\leftarrow} \mathcal{M}_{\text{IBE}}$ , set  $C = (Pt[u^*, W], \mathbf{Enc}_{\text{IBE}}(\mathbf{PK}_{\text{IBE}}, W, R))$ , update  $Pt[u^*, W] = R$  and output C;
- 2) If  $\mathbf{Pub} = \hat{C}_0^* \wedge W = W_0^*$ , then
  - a) Search  $(W, \mathbf{Pub}, Pt[u^*, W])$  for W and Pub in Pt;
  - b) If it is not found, insert  $(W, \operatorname{Pub}, Pt[u^*, W] \leftarrow \mathcal{M}_{\text{\tiny IBE}})$  to Pt, and output  $C = (\hat{K}^*_{\hat{d}}, \operatorname{Enc}_{\text{\tiny IBE}}(\operatorname{PK}_{\text{\tiny IBE}}, W, Pt[u^*, W]));$ (Note that if  $\hat{d} = 0$ , the output ciphertext C is correct, since the full-identity malleability of the IBKEM scheme allows  $\operatorname{FIM}(\hat{C}^*_0, W^*_0, u^*) = \hat{K}^*_{\hat{d}}$ . Otherwise, the output ciphertext C is incorrect. If  $\mathcal{A}$  can find this incorrectness, it implies that  $\hat{d} = 1$  holds. Accordingly,  $\mathcal{B}$  has advantage to win in the Anon-SS-sID-CPA game of the IBKEM scheme.)
  - c) Otherwise, pick  $R \stackrel{\$}{\leftarrow} \mathcal{M}_{\text{IBE}}$ , set  $C = (Pt[u^*, W], \mathbf{Enc}_{\text{IBE}}(\mathbf{PK}_{\text{IBE}}, W, R))$ , update  $Pt[u^*, W] = R$  and output C;
- 3) If  $\mathbf{Pub} \neq \hat{C}_0^*$ , then
  - a) According to **Pub**, retrieve  $(\mathbf{Pub}, u)$  from **SList**;
  - b) Search  $(W, \mathbf{Pub}, Pt[u, W])$  for W and **Pub** in **Pt**;
  - c) If it is not found, insert  $(W, \mathbf{Pub}, Pt[u, W] \stackrel{\$}{\leftarrow} \mathcal{M}_{\text{\tiny IBE}})$  to  $\mathbf{Pt}$  and output  $C = (\mathbf{FIM}(W, u), \mathbf{Enc}_{\text{\tiny IBE}}(\mathbf{Pk}_{\text{\tiny IBE}}, W, Pt[u, W]));$
  - d) Otherwise, pick  $R \stackrel{\$}{\leftarrow} \mathcal{M}_{\text{\tiny IBE}}$ , set  $C = (Pt[u, W], \mathbf{Enc}_{\text{\tiny IBE}}(\mathbf{PK}_{\text{\tiny IBE}}, W, R))$ , update Pt[u, W] = R and output C;
- Challenge Phase: In this phase,
  - 1)  $\mathcal{A}$  sends two challenge structures  $(\mathbf{Pub}_0^*, \mathbf{Pub}_1^*) \in \mathbf{PSet} \times \mathbf{PSet}$  to  $\mathcal{B}$ ;
  - 2)  $\mathcal{B}$  does the following steps:
    - a) If  $\mathbf{Pub}_0^* \neq \hat{C}_0^*$ , then abort and output  $\perp$ ;
    - b) Send two challenge IBE identity-and-message pairs  $(W_0^*, M_0^*)$  and  $(I_1^*, M_1^*)$  to  $\mathcal{G}_1$ , where  $M_0^* \stackrel{\$}{\leftarrow} \mathcal{M}_{\text{\tiny IBE}}$ and  $M_1^* \stackrel{\$}{\leftarrow} \mathcal{M}_{\text{\tiny IBE}}$ ;
  - 3)  $\mathcal{G}_2$  picks  $\tilde{d} \stackrel{\$}{\leftarrow} \{0,1\}$ , and sends the challenge IBE ciphertext  $\tilde{C}^*_{\tilde{d}} = \mathbf{Enc}_{\text{\tiny IBE}}(\mathbf{PK}_{\text{\tiny IBE}}, W^*_0, M^*_0)$  to  $\mathcal{B}$  if  $\tilde{d} = 0$ , otherwise sends  $\tilde{C}_{\tilde{d}} = \mathbf{Enc}_{\text{\tiny IBE}}(\mathbf{PK}_{\text{\tiny IBE}}, I^*_1, M^*_1)$  to  $\mathcal{B}$ .
  - 4)  $\mathcal{B}$  does the following steps:
    - a) Search  $(W_0^*, \mathbf{Pub}_0^*, Pt[u^*, W_0^*])$  for  $W_0^*$  and  $\mathbf{Pub}_0^*$  in Pt;
    - b) If it is not found, insert (W<sub>0</sub><sup>\*</sup>, Pub<sub>0</sub><sup>\*</sup>, Pt[u<sup>\*</sup>, W<sub>0</sub><sup>\*</sup>] = M<sub>0</sub><sup>\*</sup>) to Pt, output the challenge ciphertext C<sup>\*</sup> to A and stop this phase, where C<sup>\*</sup> = (K<sub>d</sub><sup>\*</sup>, C<sub>d</sub>);
      (Note that if d = 0 and d = 0, the C<sup>\*</sup> is a correct one. Otherwise, it is an incorrect one. If A confirms

(Note that if d = 0 and d = 0, the C<sup>\*</sup> is a correct one. Otherwise, it is an incorrect one. If A confirms the incorrectness of  $C^*$ , it implies that  $\hat{d} = 1$  or  $\tilde{d} = 1$  holds. Accordingly, B has advantage to win in the Anon-SS-sID-CPA game of the IBKEM or the Anon-SS-ID-CPA game of the IBE scheme.)

c) Otherwise, set the challenge ciphertext  $C^* = (Pt[u^*, W_0^*], \tilde{C}_{\tilde{d}})$ , update  $Pt[u^*, W_0^*] = M_0^*$ , send  $C^*$  to  $\mathcal{A}$  and stop this phase.

(Note that if  $\tilde{d} = 0$ , the  $C^*$  is a correct one. Otherwise, it is an incorrect one. If  $\mathcal{A}$  confirms the incorrectness of  $C^*$ , it implies that  $\tilde{d} = 1$  holds. Accordingly,  $\mathcal{B}$  has advantage to win in the Anon-SS-ID-CPA game of the IBE scheme.)

- Query Phase 2: This phase is the same as Query Phase 1. Note that in Query Phase 1 and Query Phase 2, adversary  $\mathcal{A}$  cannot query the private part corresponding to the structure  $\mathbf{Pub}_0^*$  or  $\mathbf{Pub}_1^*$  and the keyword search trapdoor corresponding to the challenge keyword  $W_0^*$  or  $W_1^*$ .
- Guess Phase: Adversary  $\mathcal{A}$  sends a guess d' to adversary  $\mathcal{B}$ .  $\mathcal{B}$  takes d' as his guess at both  $\hat{d}$  and  $\hat{d}$ , and forwards d' to challengers  $\mathcal{G}_1$  and  $\mathcal{G}_2$ .

Let  $\overline{Abort}$  denote the event that adversary  $\mathcal{B}$  does not abort in the above game. Suppose adversary  $\mathcal{A}$  totally queries  $\mathcal{Q}_{Pri}$  for  $q_p$  times. Then we have  $Pr[\overline{Abort}] = \frac{N-q_p}{N} \cdot \frac{1}{N-q_p} = \frac{1}{N}$ . Note that  $q_p \leq (N-2)$  always holds, since adversary  $\mathcal{A}$  cannot query  $\mathcal{Q}_{Pri}$  for the challenge structures ( $\mathbf{Pub}_0^*, \mathbf{Pub}_1^*$ ). Let  $Win_{IBKEM,\mathcal{B}}^{\text{Anon-SS-sID-CPA}}$  denote the event that  $\mathcal{B}$  wins in the Anon-SS-sID-CPA game of the underlying IBKEM scheme

Let  $Win_{IBKEM,\mathcal{B}}^{\text{Anon-SS-BD-CPA}}$  denote the event that  $\mathcal{B}$  wins in the Anon-SS-sID-CPA game of the underlying IBKEM scheme under the condition that  $\mathcal{B}$  does not abort. Let  $Win_{IBE,\mathcal{B}}^{\text{Anon-SS-ID-CPA}}$  denote the event that  $\mathcal{B}$  wins in the Anon-SS-ID-CPA game of the underlying IBE scheme under the condition that  $\mathcal{B}$  does not abort. Let  $Adv_{\mathcal{B}}$  be the advantage of  $\mathcal{B}$  to have  $Win_{IBKEM,\mathcal{B}}^{\text{Anon-SS-sID-CPA}}$  or  $Win_{IBE,\mathcal{B}}^{\text{Anon-SS-ID-CPA}}$  holds. Since  $\mathcal{B}$  has the probability no less than  $\frac{3}{4}$  to have  $Win_{IBKEM,\mathcal{B}}^{\text{Anon-SS-ID-CPA}}$  or  $Win_{IBE,\mathcal{B}}^{\text{Anon-SS-ID-CPA}}$  holds under the condition that  $\mathcal{B}$  does not abort, we clearly have

$$\begin{aligned} Adv_{\mathcal{B}} &= (Pr[Win_{IBKEM,\mathcal{B}}^{\text{Anon-SS-sID-CPA}} \bigvee Win_{IBE,\mathcal{B}}^{\text{Anon-SS-ID-CPA}} | \overline{Abort}] - \frac{3}{4}) \cdot Pr[\overline{Abort}] \\ &= (Pr[Win_{IBKEM,\mathcal{B}}^{\text{Anon-SS-sID-CPA}} | \overline{Abort}] + Pr[Win_{IBE,\mathcal{B}}^{\text{Anon-SS-ID-CPA}} | \overline{Abort}] \\ &- Pr[Win_{IBKEM,\mathcal{B}}^{\text{Anon-SS-sID-CPA}} \bigwedge Win_{IBE,\mathcal{B}}^{\text{Anon-SS-ID-CPA}} | \overline{Abort}] - \frac{3}{4}) \cdot Pr[\overline{Abort}] \end{aligned}$$

Let  $\overline{Belong}$  denote the event that  $(W_0^*, \mathbf{Pub}_0^*, Pt[u^*, W_0^*]) \notin \mathbf{Pt}$  holds in the above **Challenge Phase**. On the contrary, let Belong denote the event that  $(W_0^*, \mathbf{Pub}_0^*, Pt[u^*, W_0^*]) \in \mathbf{Pt}$  holds in the above **Challenge Phase**. We compute the probability  $Pr[Win_{IBKEM,\mathcal{B}}^{\text{Anon-SS-sID-CPA}} | \overline{Abort}] + Pr[Win_{IBE,\mathcal{B}}^{\text{Anon-SS-ID-CPA}} | \overline{Abort}]$  as follows.

$$\begin{split} & Pr[Win_{BEE,S}^{SubCPk} | \overline{Abort} | + Pr[Win_{BE,S}^{SubCPk} | \overline{Abort} ] \\ &= Pr[d' = d|\overline{Abort} \bigwedge \overline{Belong} ] \cdot Pr[\overline{Belong}] + Pr[d' = d|\overline{Abort} \bigwedge Belong] \cdot Pr[Belong] \\ &+ Pr[d' = d|\overline{Abort} \bigwedge \overline{Belong} \bigwedge d = 0 \land \overline{d} = 0] \cdot Pr[\overline{d} = 0 \land \overline{d} = 0] \\ &+ Pr[d' = d|\overline{Abort} \land \overline{Belong} \land \overline{d} = 1 \land \overline{d} = 0] \cdot Pr[\overline{d} = 0 \land \overline{d} = 0] \\ &+ Pr[d' = d|\overline{Abort} \land \overline{Belong} \land \overline{d} = 0 \land \overline{d} = 1] \cdot Pr[\overline{d} = 0 \land \overline{d} = 1] \\ &+ Pr[d' = d|\overline{Abort} \land \overline{Belong} \land \overline{d} = 0 \land \overline{d} = 1] \cdot Pr[\overline{d} = 0 \land \overline{d} = 1] \\ &+ Pr[d' = d|\overline{Abort} \land \overline{Belong} \land \overline{d} = 1 \land \overline{d} = 1] \cdot Pr[\overline{d} = 0 \land \overline{d} = 1] \\ &+ Pr[d' = d|\overline{Abort} \land \overline{Belong} \land \overline{d} = 1 \land \overline{d} = 1] \cdot Pr[\overline{d} = 0 \land \overline{d} = 1] \\ &+ Pr[d' = d|\overline{Abort} \land \overline{Belong} \land \overline{d} = 0 \land \overline{d} = 0] \cdot Pr[\overline{d} = 0 \land \overline{d} = 0] \\ &+ Pr[d' = d|\overline{Abort} \land \overline{Belong} \land \overline{d} = 0 \land \overline{d} = 0] \cdot Pr[\overline{d} = 0 \land \overline{d} = 0] \\ &+ Pr[d' = d|\overline{Abort} \land \overline{Belong} \land \overline{d} = 1 \land \overline{d} = 0] \cdot Pr[\overline{d} = 0 \land \overline{d} = 1] \\ &+ Pr[d' = d|\overline{Abort} \land \overline{Belong} \land \overline{d} = 1 \land \overline{d} = 0] \cdot Pr[\overline{d} = 0 \land \overline{d} = 1] \\ &+ Pr[d' = d|\overline{Abort} \land \overline{Belong} \land \overline{d} = 1 \land \overline{d} = 1] \cdot Pr[\overline{d} = 0 \land \overline{d} = 1] \\ &+ Pr[d' = d|\overline{Abort} \land \overline{Belong} \land \overline{d} = 1 \land \overline{d} = 1] \cdot Pr[\overline{d} = 0 \land \overline{d} = 0] \\ &+ Pr[d' = d|\overline{Abort} \land \overline{Belong} \land \overline{d} = 1 \land \overline{d} = 0] \cdot Pr[\overline{d} = 0 \land \overline{d} = 0] \\ &+ Pr[d' = d|\overline{Abort} \land \overline{Belong} \land \overline{d} = 1 \land \overline{d} = 1] \cdot Pr[\overline{d} = 0 \land \overline{d} = 0] \\ &+ Pr[d' = d|\overline{Abort} \land \overline{Belong} \land \overline{d} = 1 \land \overline{d} = 1] \cdot Pr[\overline{d} = 1 \land \overline{d} = 1] \\ &+ Pr[d' = d|\overline{Abort} \land \overline{Belong} \land \overline{d} = 1 \land \overline{d} = 1] \cdot Pr[\overline{d} = 0 \land \overline{d} = 1] \\ &+ Pr[d' = d|\overline{Abort} \land \overline{Belong} \land \overline{d} = 1 \land \overline{d} = 1] \cdot Pr[\overline{d} = 0 \land \overline{d} = 0] \\ &+ Pr[d' = d|\overline{Abort} \land \overline{Belong} \land \overline{d} = 1 \land \overline{d} = 0] \cdot Pr[\overline{d} = 0 \land \overline{d} = 1] \\ &+ Pr[d' = d|\overline{Abort} \land \overline{Belong} \land \overline{d} = 1 \land \overline{d} = 1] \cdot Pr[\overline{d} = 0] \\ &+ Pr[d' = d|\overline{Abort} \land \overline{Belong} \land \overline{d} = 1 \land \overline{d} = 1] \cdot Pr[\overline{d} = 0] \\ &+ Pr[d' = d|\overline{Abort} \land \overline{Belong} \land \overline{d} = 1 \land \overline{d} = 0] \\ &+ Pr[d' = d|\overline{Abort} \land \overline{Belong} \land \overline{d} = 1 \land \overline{d} = 1] \cdot Pr[\overline{d} = 0] \\ &+ Pr[d' = d|\overline{Abort} \land \overline{Belong} \land \overline{d} = 1 \land \overline{d} = 1]$$

We compute the probability  $Pr[Win_{IBKEM,\mathcal{B}}^{\text{Anon-SS-sID-CPA}} \land Win_{IBE,\mathcal{B}}^{\text{Anon-SS-ID-CPA}} | \overline{Abort}]$  as follows.

$$\begin{split} ⪻[Win_{IBKEM,\mathcal{B}}^{\text{Anon-SS-sID-CPA}} \bigwedge Win_{IBE,\mathcal{B}}^{\text{Anon-SS-ID-CPA}} | \overline{Abort}] \\ &= Pr[d' = \hat{d} = \tilde{d} | \overline{Abort} \bigwedge \overline{Belong}] \cdot Pr[\overline{Belong}] + Pr[d' = \hat{d} = \tilde{d} | \overline{Abort} \bigwedge Belong] \cdot Pr[Belong] \\ &= (Pr[d' = \hat{d} = \tilde{d} | \overline{Abort} \bigwedge \overline{Belong} \bigwedge \hat{d} = 0 \bigwedge \tilde{d} = 0] \cdot Pr[\hat{d} = 0 \bigwedge \tilde{d} = 0] \\ &+ Pr[d' = \hat{d} = \tilde{d} | \overline{Abort} \bigwedge \overline{Belong} \bigwedge \hat{d} = 1 \bigwedge \tilde{d} = 1] \cdot Pr[\hat{d} = 1 \bigwedge \tilde{d} = 1]) \cdot Pr[\overline{Belong}] \\ &+ (Pr[d' = \hat{d} = \tilde{d} | \overline{Abort} \bigwedge Belong \bigwedge \hat{d} = 0 \bigwedge \tilde{d} = 0] \\ &+ Pr[d' = \hat{d} = \tilde{d} | \overline{Abort} \bigwedge Belong \bigwedge \hat{d} = 1 \bigwedge \tilde{d} = 1] \cdot Pr[\hat{d} = 0 \bigwedge \tilde{d} = 0] \\ &+ Pr[d' = \hat{d} = \tilde{d} | \overline{Abort} \bigwedge Belong \bigwedge \hat{d} = 1 \bigwedge \tilde{d} = 1] \cdot Pr[\hat{d} = 0 \bigwedge \tilde{d} = 0] \\ &+ Pr[d' = \hat{d} = \tilde{d} | \overline{Abort} \bigwedge Belong \bigwedge \hat{d} = 1 \bigwedge \tilde{d} = 1] \cdot Pr[\hat{d} = 1 \bigwedge \tilde{d} = 1]) \cdot Pr[Belong] \\ &= (Adv_{SPCHS,\mathcal{A}}^{SS-sK-CKSA} + \frac{1}{2} + Pr[d' = \hat{d} = \tilde{d} | \overline{Abort} \bigwedge Belong \bigwedge \hat{d} = 1 \bigwedge \tilde{d} = 1]) \cdot \frac{1}{4} \cdot Pr[\overline{Belong}] \\ &+ (Adv_{SPCHS,\mathcal{A}}^{SS-sK-CKSA} + \frac{1}{2} + Pr[d' = \hat{d} = \tilde{d} | \overline{Abort} \bigwedge Belong \bigwedge \hat{d} = 1 \bigwedge \tilde{d} = 1]) \cdot \frac{1}{4} \cdot Pr[Belong] \\ &= (Adv_{SPCHS,\mathcal{A}}^{SS-sK-CKSA} + \frac{1}{2}) \cdot \frac{1}{4} + Pr[d' = \hat{d} = \tilde{d} | \overline{Abort} \bigwedge \hat{d} = 1 \bigwedge \tilde{d} = 1] \cdot \frac{1}{4} = \frac{1}{4} \cdot Adv_{SPCHS,\mathcal{A}}^{SS-sK-CKSA} + \frac{1}{4} \end{split}$$

According to the above computations, we have

$$\begin{split} Adv_{\mathcal{B}} &= (Pr[Win_{IBKEM,\mathcal{B}}^{\text{Anon-SS-sID-CPA}} \bigvee Win_{IBE,\mathcal{B}}^{\text{Anon-SS-ID-CPA}} | \overline{Abort}] - \frac{3}{4}) \cdot Pr[\overline{Abort}] \\ &= (Pr[Win_{IBKEM,\mathcal{B}}^{\text{Anon-SS-sID-CPA}} | \overline{Abort}] + Pr[Win_{IBE,\mathcal{B}}^{\text{Anon-SS-ID-CPA}} | \overline{Abort}] \\ &- Pr[Win_{IBKEM,\mathcal{B}}^{\text{Anon-SS-sID-CPA}} \bigwedge Win_{IBE,\mathcal{B}}^{\text{Anon-SS-ID-CPA}} | \overline{Abort}] - \frac{3}{4}) \cdot Pr[\overline{Abort}] \\ &= \frac{1}{4N} \cdot Adv_{SPCHS,\mathcal{A}}^{\text{SS-sK-CKSA}} \end{split}$$

In addition, it is clear that adversary  $\mathcal{B}$  is a PPT adversary, if  $\mathcal{A}$  is a PPT adversary. In conclusion, we have that if a PPT adversary  $\mathcal{A}$  wins in the SS-sK-CKSA game of the generic SPCHS construction with advantage  $Adv_{SPCHS,\mathcal{A}}^{SS-sK-CKSA}$ , then the above PPT adversary  $\mathcal{B}$  can utilize the capability of adversary  $\mathcal{A}$  to win in the Anon-SS-sID-CPA game of the underlying IBKEM scheme or the Anon-SS-ID-CPA game of the underlying IBE scheme with advantage  $\frac{1}{4N} \cdot Adv_{SPCHS,\mathcal{A}}^{SS-sK-CKSA}$ .

#### F. A Collision-free Full-identity Malleable IBKEM Instance in the RO Model

We first review the VRF-suitable IBKEM instance proposed in Appendix A.2 of [8]. Then we prove its collision-free full-identity malleability and the Anon-SS-ID-CPA security in the RO model. Let identity space  $\mathcal{ID}_{IBKEM} = \{0, 1\}^*$ . This IBKEM instance is as follows.

- Setup<sub>IBKEM</sub> $(1^k, \mathcal{ID}_{IBKEM})$ : Take as input a security parameter  $1^k$  and the identity space  $\mathcal{ID}_{IBKEM}$ , compute  $(q, \mathbb{G}, \mathbb{G}_1, g, \hat{e}) \stackrel{\$}{\leftarrow} \mathbf{BGen}(1^k)$ , pick  $s \stackrel{\$}{\leftarrow} \mathbb{Z}_q^*$ , set  $P \leftarrow g^s$ , choose a cryptographic hash function  $H : \{0, 1\}^* \to \mathbb{G}$ , set the encapsulated key space  $\mathcal{K}_{IBKEM} = \mathbb{G}_1$ , set the encapsulation space  $\mathcal{C}_{IBKEM} = \mathbb{G}$ , and output the master public key  $\mathbf{PK}_{IBKEM} = (q, \mathbb{G}, \mathbb{G}_1, g, \hat{e}, P, H, \mathcal{ID}_{IBKEM}, \mathcal{K}_{IBKEM}, \mathcal{C}_{IBKEM})$  and the master secret key  $\mathbf{SK}_{IBKEM} = s$ .
- Extract<sub>IBKEM</sub> (SK<sub>IBKEM</sub>, *ID*): Take as inputs SK<sub>IBKEM</sub> and an identity  $ID \in ID_{IBKEM}$ , and output a decryption key  $\hat{S}_{ID} = H(ID)^s$  of *ID*.
- Encaps<sub>IBKEM</sub> (**PK**<sub>IBKEM</sub>, *ID*, *r*): Take as inputs **PK**<sub>IBKEM</sub>, an identity  $ID \in \mathcal{ID}_{IBKEM}$  and a random value *r*, and output a key-and-encapsulation pair  $(\hat{K}, \hat{C})$ , where  $\hat{K} = \hat{e}(P, H(ID))^r$  and  $\hat{C} = g^r$ .
- **Decaps**<sub>IBKEM</sub>  $(\hat{S}_{ID'}, \hat{C})$ : Take as inputs the decryption key  $\hat{S}_{ID'}$  of identity ID' and an encapsulation  $\hat{C}$ , and output the encapsulated key  $\hat{K} = \hat{e}(\hat{C}, \hat{S}_{ID'})$  if  $\hat{C} \in \mathbb{G}$  or output  $\perp$  otherwise.

**Collision-Free Full-Identity Malleability.** Let the function  $\mathbf{FIM}(ID, r) = \hat{e}(P, H(ID))^r$  for any identity  $ID \in \mathcal{ID}_{\text{IBKEM}}$  and any random value  $r \in \mathbb{Z}_q^*$ . Clearly, the function **FIM** is efficient. Moreover, it is easy to find that the function **FIM** has collision-freeness and full-identity malleability by the following reasons.

For any  $(\hat{K}, \hat{C}) = \mathbf{Encaps}_{\text{IBKEM}}(\mathbf{PK}_{\text{IBKEM}}, ID, r)$  and any identity  $ID' \in \mathcal{ID}_{\text{IBKEM}}$ , it is clear that  $\mathbf{FIM}(ID', r) = \hat{e}(P, H(ID'))^r = \mathbf{Decaps}_{\text{IBKEM}}(\hat{S}_{ID'}, \hat{C})$  holds. So the function **FIM** has full-identity malleability. In addition, for any identity  $ID' \in \mathcal{ID}_{\text{IBKEM}}$ , if  $ID \neq ID'$ , we clearly have  $\mathbf{FIM}(ID, r) \neq \mathbf{FIM}(ID', r)$  due to the collision freeness of the hash function H; for any random value  $r' \in \mathbb{Z}_q^*$ , if  $r \neq r'$ , we clearly have  $\mathbf{FIM}(ID, r) \neq \mathbf{FIM}(ID, r) \neq \mathbf{FIM}(ID, r')$  due to the randomness of the values r and r'. Therefore, the function **FIM** offers collision-freeness, except with a negligible probability in the security parameter k.

Anon-SS-ID-CPA Security. The Anon-SS-ID-CPA security of the above IBKEM instance is based on the DBDH assumption in the RO model. The formal result is the following theorem.

**Theorem 6.** Let the hash function H be modeled as the random oracle  $Q_H(\cdot)$ . Suppose a PPT adversary A wins in the Anon-SS-ID-CPA game of the above IBKEM instance with advantage  $Adv_{IBKEM,A}^{Anon-SS-ID-CPA}$ , in which A makes at most

 $q_p$  queries to oracle  $\mathcal{Q}_{DK}^{IBKEM}(\cdot)$ . Then there is a PPT algorithm  $\mathcal{B}$  that solves the DBDH problem in **BGen** $(1^k)$  with advantage approximately

$$Adv_{\mathcal{B}}^{DBDH}(1^k) \approx \frac{4}{(e \cdot q_p)^2} \cdot Adv_{IBKEM,\mathcal{A}}^{Anon-SS-ID-CPA}$$

where e is the base of natural logarithms.

*Proof:* To prove this theorem, we will construct a PPT algorithm  $\mathcal{B}$  that plays the Anon-SS-ID-CPA game with adversary  $\mathcal{A}$  and utilizes the capability of  $\mathcal{A}$  to solve the DBDH problem in **BGen**(1<sup>k</sup>) with advantage approximately  $\frac{4}{(e \cdot q_p)^2} \cdot Adv_{IBKEM,\mathcal{A}}^{\text{Anon-SS-ID-CPA}}$ . Let  $Coin \stackrel{\sigma}{\leftarrow} \{0,1\}$  denote the operation that picks  $Coin \in \{0,1\}$  according to the probability  $Pr[Coin = 1] = \sigma$  (the specified value of  $\sigma$  will be decided latter). The constructed algorithm  $\mathcal{B}$  in the Anon-SS-ID-CPA game is as follows.

- Setup Phase: Algorithm  $\mathcal{B}$  takes as inputs  $(q, \mathbb{G}, \mathbb{G}_1, g, \hat{e}, g^a, g^b, g^c, Z)$  (where Z equals either  $\hat{e}(g, g)^{abc}$  or  $\hat{e}(g, g)^y$ ) and the identity space  $\mathcal{ID}_{IBKEM}$ , and does the following steps:
  - 1) Initialize a list  $\mathbf{HList} = \emptyset \subseteq \mathcal{ID}_{\text{IBKEM}} \times \mathbb{G} \times \mathbb{Z}_q^* \times \{0, 1\};$
  - 2) Set the encapsulated key space  $\mathcal{K}_{\text{IBKEM}} = \mathbf{G}_1$ , the encapsulation space  $\mathcal{C}_{\text{IBKEM}} = \mathbf{G}$  and  $\mathbf{PK}_{\text{IBKEM}} =$  $(q, \mathbb{G}, \mathbb{G}_1, g, \hat{e}, P = g^a, \mathcal{ID}_{\text{IBKEM}}, \mathcal{K}_{\text{IBKEM}}, \mathcal{C}_{\text{IBKEM}});$
  - 3) Send  $\mathbf{PK}_{\text{IBKEM}}$  to adversary  $\mathcal{A}$ ;
- Query Phase 1: Adversary A adaptively issues the following queries multiple times.
  - Hash Query  $\mathcal{Q}_H(ID)$ : Taking as input an identity  $ID \in \mathcal{ID}_{\text{IBKEM}}$ , algorithm  $\mathcal{B}$  does the following steps:

    - Pick x <sup>§</sup> Z<sup>\*</sup><sub>q</sub> and Coin <sup>σ</sup> {0,1};
       If Coin = 0, add (ID, g<sup>x</sup>, x, Coin) into HList and output g<sup>x</sup>;
    - 3) Otherwise, add  $(ID, g^{c \cdot x}, x, Coin)$  into **HList** and output  $g^{c \cdot x}$ ;
  - Decryption Key Query  $\mathcal{Q}_{DK}^{IBKEM}(ID)$ : Taking as input an identity  $ID \in \mathcal{ID}_{IBKEM}$ , algorithm  $\mathcal{B}$  does the following steps:
    - 1) If  $(ID, *, *, *) \notin \mathbf{HList}$ , query  $\mathcal{Q}_H(ID)$ ;
    - 2) According to ID, retrieve (ID, X, x, Coin) from **HList**;
    - 3) If Coin = 0, output  $g^{a \cdot x}$ ; otherwise, abort and output  $\bot$ ;
- Challenge Phase: Adversary  $\mathcal{A}$  sends two challenge identities  $ID_0^* \in \mathcal{ID}_{\text{IBKEM}}$  and  $ID_1^* \in \mathcal{ID}_{\text{IBKEM}}$  to algorithm  $\mathcal{B}$ ;  $\mathcal{B}$  picks  $\hat{d} \stackrel{\$}{\leftarrow} \{0, 1\}$ , and does the following steps:
  - 1) If  $(ID_0^*, *, *, *) \notin \mathbf{HList}$ , query  $\mathcal{Q}_H(ID_0^*)$ ;
  - 2) If  $(ID_1^*, *, *, *) \notin \mathbf{HList}$ , query  $\mathcal{Q}_H(ID_1^*)$ ;
  - 3) According to  $ID_0^*$  and  $ID_1^*$ , retrieve  $(ID_0^*, X_0^*, x_0^*, Coin_0^*)$  and  $(ID_1^*, X_1^*, x_1^*, Coin_1^*)$  from **HList**;
  - 4) If  $Coin_0^* = 0 \bigvee Coin_1^* = 0$ , then abort and output  $\perp$ ;
  - 5) Finally send the challenge key-and-encapsulation pair  $(Z^{\hat{x}_{\hat{d}}}, g^b)$  to adversary  $\mathcal{A}$ ;
- Query Phase 2: This phase is the same as Query Phase 2. Note that in Query Phase 1 and Query Phase 2, adversary  $\mathcal{A}$  cannot query the decryption key corresponding to the challenge identity  $ID_0^*$  or  $ID_1^*$ .
- Guess Phase: Adversary  $\mathcal{A}$  sends a guess d' to algorithm  $\mathcal{B}$ . If d = d',  $\mathcal{B}$  output 1; otherwise, output 0.

Let  $\overline{Abort}$  denote the event that algorithm  $\mathcal{B}$  does not abort in the above game. Next, we will compute the probabilities  $Pr[\overline{Abort}], Pr[\mathcal{B}=1|Z=\hat{e}(g,g)^{abc}] \text{ and } Pr[\mathcal{B}=1|Z=\hat{e}(g,g)^y], \text{ and the advantage } Adv_{\mathcal{B}}^{DBDH}(1^k).$ 

According to the above game, the probability of the event  $\overline{Abort}$  only relies on the probability  $\sigma$  and the number of times of adversary  $\mathcal{A}$  to query oracle  $\mathcal{Q}_{DK}^{IBKEM}(ID)$ . We have that  $Pr[\overline{Abort}] = (1-\sigma)^{q_p} \cdot \sigma^2$ . Let  $\sigma = \frac{2}{2+q_p}$ . We have that  $Pr[\overline{Abort}] \approx \frac{4}{(e \cdot q_p)^2}$ , where e is the base of natural logarithms.

When  $Z = \hat{e}(g, g)^{abc}$  and the event  $\overline{Abort}$  holds, it is easy to find that algorithm  $\mathcal{B}$  simulates a real Anon-SS-ID-CPA game in adversary  $\mathcal{A}$ 's mind. So we have  $Pr[\hat{d} = \hat{d}' | \overline{Abort} \bigwedge Z = \hat{e}(g, g)^{abc}] = (Adv_{IBKEM, \mathcal{A}}^{\text{Anon-SS-ID-CPA}} + \frac{1}{2}).$ 

When  $Z = \hat{e}(g,g)^y$  and the event  $\overline{Abort}$  holds, algorithm  $\mathcal{B}$  generates an incorrect challenge ciphertext, and it is independent of the challenge identities  $ID_0^*$  and  $ID_1^*$ . So we have  $Pr[d = d'|\overline{Abort} \bigwedge Z = \hat{e}(g,g)^y] = \frac{1}{2}$ .

Now, we can compute the advantage  $Adv_{\mathcal{B}}^{DBDH}(1^k)$  as follows:

$$\begin{split} Adv_{\mathcal{B}}^{DBDH}(1^k) &= \Pr[\mathcal{B} = 1 | Z = \hat{e}(g, g)^{abc}] - \Pr[\mathcal{B} = 1 | Z = \hat{e}(g, g)^y] \\ &= \Pr[\hat{d} = \hat{d}' \bigwedge \overline{Abort} | Z = \hat{e}(g, g)^{abc}] - \Pr[\hat{d} = \hat{d}' \bigwedge \overline{Abort} | Z = \hat{e}(g, g)^y] \\ &= \Pr[\hat{d} = \hat{d}' | \overline{Abort} \bigwedge Z = \hat{e}(g, g)^{abc}] \cdot \Pr[\overline{Abort} | Z = \hat{e}(g, g)^{abc}] \\ &- \Pr[\hat{d} = \hat{d}' | \overline{Abort} \bigwedge Z = \hat{e}(g, g)^y] \cdot \Pr[\overline{Abort} | Z = \hat{e}(g, g)^y] \\ &\approx (Adv_{IBKEM,\mathcal{A}}^{\text{Anon-SS-ID-CPA}} + \frac{1}{2}) \cdot \frac{4}{(e \cdot q_p)^2} - \frac{1}{2} \cdot \frac{4}{(e \cdot q_p)^2} = \frac{4}{(e \cdot q_p)^2} \cdot Adv_{IBKEM,\mathcal{A}}^{\text{Anon-SS-ID-CPA}} \end{split}$$

In addition, it is clear that algorithm  $\mathcal{B}$  is a PPT algorithm, if adversary  $\mathcal{A}$  is a PPT adversary. In conclusion, if a PPT adversary  $\mathcal{A}$  wins in the Anon-SS-ID-CPA game of the above IBKEM instance with advantage  $Adv_{IBKEM,\mathcal{A}}^{\text{Anon-SS-ID-CPA}}$ , in which  $\mathcal{A}$  makes at most  $q_p$  queries to oracle  $\mathcal{Q}_{DK}^{IBKEM}(\cdot)$ , then there is a PPT algorithm  $\mathcal{B}$  that solves the DBDH problem in **BGen**(1<sup>k</sup>) with advantage approximately

$$Adv_{\mathcal{B}}^{DBDH}(1^k) \approx \frac{4}{(e \cdot q_p)^2} \cdot Adv_{IBKEM,\mathcal{A}}^{\text{Anon-SS-ID-CPA}}$$

where e is the base of natural logarithms.

#### G. Proof of Theorem 5

*Proof:* Suppose a PPT adversary  $\mathcal{A}$  wins in the Anon-SS-ID-CPA game of the above IBKEM instance with advantage  $Adv_{IBKEM,\mathcal{A}}^{\text{Anon-SS-ID-CPA}}$ , in which  $\mathcal{A}$  makes at most  $q_p$  queries to oracle  $\mathcal{Q}_{DK}^{IBKEM}(\cdot)$ . To prove this theorem, we will construct a PPT algorithm  $\mathcal{B}$  that plays the Anon-SS-ID-CPA game with adversary  $\mathcal{A}$  and utilizes the capability of  $\mathcal{A}$  to break the  $(\ell + 1)$ -MDDH assumption in  $\mathbf{MG}_{\ell+1}(1^k)$ . The constructed algorithm  $\mathcal{B}$  in the Anon-SS-ID-CPA game is as follows.

- Setup Phase: Algorithm  $\mathcal{B}$  gets as input an  $(\ell+1)$ -group system  $\mathbf{MPG}_{\ell+1}$  and group elements  $g, g^{x_1}, \dots, g^{x_{\ell+2}} \in \mathbb{G}_1$  and  $S \in \mathbb{G}_{\ell+1}$ , where either  $S = \hat{e}(g^{x_1}, \dots, g^{x_{\ell+1}})^{x_{\ell+2}}$  (i.e., S is real) or  $S \in \mathbb{G}_{\ell+1}$  uniformly (i.e., S is random).  $\mathcal{B}$  generates a  $(q_p, 2)$ -MPHF H into  $\mathbb{G}_\ell$ , sets up the master public key as  $\mathbf{PK} = (\mathbf{MPG}_{\ell+1}, hk, \mathbf{H}, h, h', \mathcal{ID}, \mathcal{K}, \mathcal{C})$  for  $(h, h') = (g, g^{x_{\ell+1}})$  and  $(hk, td) \leftarrow \mathbf{TGen}(1^k, g^{x_1}, \dots, g^{x_\ell}, g)$ , finally sends  $\mathbf{PK}_{\text{IBKEM}}$  to adversary  $\mathcal{A}$ . Here, we use the **TGen** and **TEval** algorithms of the  $(q_p, 2)$ -MPHF property of H.
- Query Phase 1: Adversary A adaptively issues the following query multiple times.
  - Decryption Key Query  $Q_{DK}^{IBKEM}(ID)$ : Taking as input an identity  $ID \in ID_{IBKEM}$ , algorithm  $\mathcal{B}$  does the following steps:
    - 1) Compute  $\mathbf{TEval}(td, ID) = (a_{ID}, B_{ID});$
    - 2) If  $a_{ID} = 0$ , return  $\hat{S}_{ID} = \hat{e}(B_{ID}, h')$ ; otherwise, abort and output  $\perp$ ;

Note that we have  $\hat{S}_{ID} = \hat{e}(B_{ID}, h') = \hat{e}(B_{ID}, h)^{x_{\ell+1}} = \mathbf{H}_{hk}(ID)^{x_{\ell+1}}$ . So  $\mathcal{B}$  can answer a  $\mathcal{Q}_{DK}^{IBKEM}(ID)$  query of  $\mathcal{A}$  for identity ID precisely when  $a_{ID} = 0$ .

- Challenge Phase: Adversary  $\mathcal{A}$  sends two challenge identities  $ID_0^* \in \mathcal{ID}_{\text{IBKEM}}$  and  $ID_1^* \in \mathcal{ID}_{\text{IBKEM}}$  to algorithm  $\mathcal{B}$ ;  $\mathcal{B}$  picks  $\hat{d} \stackrel{\$}{\leftarrow} \{0, 1\}$ , and does the following steps:
  - 1) Compute  $\mathbf{TEval}(td, ID_0^*) = (a_{ID_0^*}, B_{ID_0^*})$  and  $\mathbf{TEval}(td, ID_1^*) = (a_{ID_1^*}, B_{ID_1^*});$
  - 2) If  $a_{ID_0^*} = 0 \bigvee a_{ID_1^*} = 0$ , then abort and output  $\perp$ ;
  - 3) Send the challenge key-and-encapsulation pair  $(\hat{K}^*_{\hat{d}} = S^{a_{ID^*_{\hat{d}}}} \cdot \hat{e}(B_{ID^*_{\hat{d}}}, g^{x_{\ell+1}}, g^{x_{\ell+2}}), \hat{C}^*_0 = g^{x_{\ell+2}})$  to adversary  $\mathcal{A}$ ;

Suppose algorithm  $\mathcal{B}$  does not abort (i.e., both  $a_{ID_0^*} \neq 0$  and  $a_{ID_1^*} \neq 0$  hold), we have  $\mathbf{H}_{hk}(ID_0^*) = \hat{e}(g^{x_1}, \cdots, g^{x_\ell})^{a_{ID_0^*}} \cdot \hat{e}(B_{ID_0^*}, h)$  and  $\mathbf{H}_{hk}(ID_1^*) = \hat{e}(g^{x_1}, \cdots, g^{x_\ell})^{a_{ID_1^*}} \cdot \hat{e}(B_{ID_1^*}, h)$ . Furthermore, if  $S = \hat{e}(g^{x_1}, \cdots, g^{x_\ell+1})^{x_{\ell+2}}$ , we have  $\hat{K}_{\hat{d}}^* = S^{a_{ID_{\hat{d}}^*}} \cdot \hat{e}(B_{ID_{\hat{d}}^*}, g^{x_{\ell+1}}, g^{x_{\ell+2}}) = \hat{e}(\mathbf{H}_{hk}(ID_{\hat{d}}^*), g^{x_{\ell+1}})^{x_{\ell+2}}$ . This implies that the challenge key-and-encapsulation pair  $(\hat{K}_{\hat{d}}^*, \hat{C}_0^*)$  is a valid one in this case. Otherwise,  $\hat{K}_{\hat{d}}^*$  contains no information about  $\hat{d}$ .

- Query Phase 2: This phase is the same as Query Phase 2. Note that in Query Phase 1 and Query Phase 2, adversary A cannot query the decryption key corresponding to the challenge identity  $ID_0^*$  or  $ID_1^*$ .
- Guess Phase: Adversary  $\mathcal{A}$  sends a guess  $\hat{d}'$  to algorithm  $\mathcal{B}$ . Let  $\overline{Abort'}$  denote the event that  $\mathcal{B}$  does not abort in the previous phases. Let  $\mathcal{I} = \{ID_1, \dots, ID_{q_p}, ID_0^*, ID_1^*\}$  be the set of the queried IDs by  $\mathcal{A}$  and the challenge identities  $ID_0^*$  and  $ID_1^*$ . Let  $P_{\mathcal{I}} = Pr[\overline{Abort'}|\mathcal{I}]$ , which will be decided later. As in [9], [48],  $\mathcal{B}$  "artificially" aborts with probability  $1 1/(P_{\mathcal{I}} \cdot p(k))$  for the polynomial p(k) from Definition 12 and outputs  $\bot$ . If it does not abort,  $\mathcal{B}$  uses the guess of  $\mathcal{A}$ . This means that if  $\hat{d} = \hat{d'}$ ,  $\mathcal{B}$  outputs 1, otherwise it outputs 0.

In **Guess Phase**,  $\mathcal{B}$  did not directly use the guess of  $\mathcal{A}$ , since event  $\overline{Abort'}$  might not be independent of the identities in  $\mathcal{I}$ . So  $\mathcal{B}$  "artificially" aborts to achieve the independence. Let  $\overline{Abort'}$  be the event that  $\mathcal{B}$  does not abort in the above game. We have that  $Pr[\overline{Abort}] = 1 - Pr[Abort'|\mathcal{I}] - Pr[\overline{Abort'}|\mathcal{I}] \cdot (1 - 1/(P_{\mathcal{I}} \cdot p(k))) = 1/p(k)$ . Hence, we have  $Pr[\mathcal{B} = 1|S \text{ is real}] = Pr[\overline{Abort}] \cdot (\frac{1}{2} + Adv_{IBKEM,\mathcal{A}}^{\text{Anon-SS-ID-CPA}})$  and  $Pr[\mathcal{B} = 1|S \text{ is random}] = Pr[\overline{Abort}] \cdot \frac{1}{2}$ , where  $\frac{1}{2} + Adv_{IBKEM,\mathcal{A}}^{\text{Anon-SS-ID-CPA}}$  is the probability that  $\mathcal{A}$  succeeds in the Anon-SS-ID-CPA game of IBKEM. Further, we have

$$Pr[\mathcal{B} = 1|S \text{ is real}] - Pr[\mathcal{B} = 1|S \text{ is random}] = \frac{1}{p(k)} \cdot Adv_{IBKEM,\mathcal{A}}^{\text{Anon-SS-ID-CPA}}$$

Hence,  $\mathcal{B}$  breaks the  $(\ell + 1)$ -MDDH assumption if and only if  $\mathcal{A}$  breaks the Anon-SS-ID-CPA security of the above IBKEM scheme.

Finally, to evaluate  $P_{\mathcal{I}}$ , we can only approximate it (up to an inversely polynomial error, by running **TEval** with freshly generated keys sufficiently often), which introduces an additional error term in the analysis. We refer to [48] for details on this evaluation.