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, Josep Domingo-Ferrer, 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 to organize *Searchable Public-Key Ciphertexts with Hidden Structures* (SPCHS) for fast keyword search 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. It is shown semantically secure based on the decisional bilinear Diffle-Hellman assumption in the Random Oracle (RO) model. The scheme enjoys its search complexity 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

PUBLIC-KEY encryption with keyword search (PEKS), introduced by Boneh *et al.* in [11], 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. In more details, each sender separately encrypts a file and its extracted keyword and sends the resulting ciphertexts to a server; when the receiver would like to retrieve the files containing a specific keyword, he delegates a keyword search trapdoor to the server; the server can find the encrypted files containing the queried keyword without knowing the original files or the keyword itself, and return the corresponding encrypted files to the receiver; finally, the receiver can decrypt these encrypted files[1]. The authors of PEKS [11] 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.

One of the prominent works to accelerate search over encrypted keywords in the public-key setting is deterministic encryption introduced by Bellare *et al*. in [7]. An encryption scheme is deterministic if the encryption algorithm is deterministic. Bellare *et al*. [7] 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 the time complexity logarithmic in the total number of all ciphertexts. This is reasonable because the encrypted keywords can form a tree-like structure when being 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* hard-to-guess by the adversary (*i.e.*, keywords with high min-entropy 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 high efficient search performance without sacrificing semantic security in PEKS.

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, 430074, China. E-Mail: {xupeng, hjin}@mail.hust.edu.cn.

Q. Wu is with School of Electronics and Information Engineering, Beihang Univerisity, Beijing, 100191, China. E-Mail: qianhong.wu@buaa.edu.cn.

W. Wang is with Embedded and Pervasive Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, 430074, China. E-Mail: viviawangww@gmail.com.

W. Susilo is with 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, Tarragona, Catalonia. E-Mail: josep.domingo@urv.cat.

[1]Since the encryption of the original files can be separately processed with an independent public-key encryption scheme as in [11], we only describe the encryption of the keywords unless clearly emphasized throughout the paper.

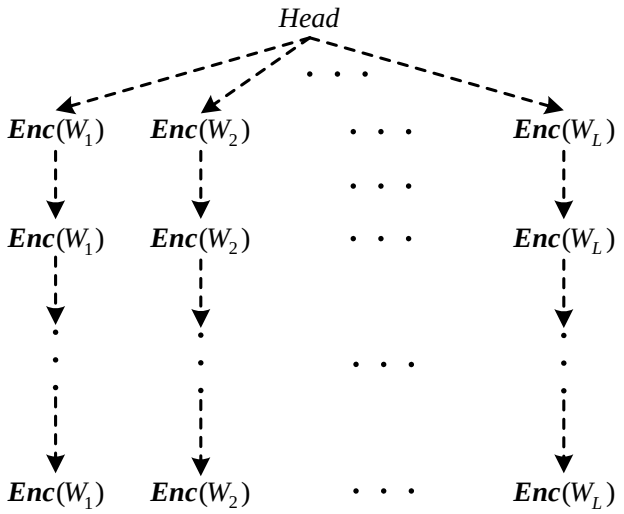


Figure 1: The Hidden Star-like Structure formed by Keyword Searchable Ciphertexts. (The dashed arrows denote the hidden relations. $Enc(W_i)$ denotes the searchable ciphertext of keyword W_i.)

Observe that a keyword space is usually of no high min-entropy 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 keyword searchable 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. So on and so forth, 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 keep 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 should be indistinguishable, and no information should be 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 from formally defining the concept of Searchable Public-key Ciphertexts with Hidden Structures (SPCHS) and its semantic security. In this new concept, keyword searchable ciphertexts with their hidden structures can be generated in the public key setting; with a keyword search trapdoor, partial relations can be disclosed and guide one to find out all matching ciphertexts. The semantic security is defined for both the keywords and the hidden structures. It is worth to note 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 (see Section III). 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 [6] 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 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 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 of 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 [20], [34], [4], [28] in either the RO model or 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[2] (VRF) [3]. We show that one of these IBKEM schemes is anonymous and collision-free full-identity malleable in the RO model. In [31], Freire *et al.* utilized the "approximation" of multilinear maps [35] to construct a standard-model version of Boneh-and-Franklin (BF) IBE scheme [14]. We transform this IBE scheme into a collision-free full-identity malleable IBKEM scheme with the 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 collision-free full-identity malleable IBKEM is of independent interest. In addition to work as a building block for generic SPCHS construction, it may also find other applications, as outlined as follows.

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 identity-based cryptography setting, e.g., encapsulating a session key K to a single user ID. According to the collision-freeness 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 only cost of 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.* [8] and Libert *et al.* [41] in the traditional public-key setting. With collision-free full-identity malleable IBKEM, a sender generates an identity-based broadcast ciphertext $\langle C_1, C_2, (K_1^1||SE(K_2^1, F_1)), ..., (K_1^N||SE(K_2^N, F_N))\rangle$, 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 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 keeps 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.

[2]VRF behaves like a pseudo-random function but one can verify that the output was pseudo-random.

D. Related Works

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 [23] 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 [47]. Their instantiated scheme takes search time linear with the size of the database. A number of efforts [33], [9], [5], [27], [10] follow this research line and refine Song *et al.*'s original work. The SEKS scheme due to Curtmola *et al*. [23] has been proven to be semantically secure against an adaptive adversary. It allows 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 [23], [13] extend SEKS to a multi-sender scenario. The work in [42] realizes fuzzy keyword search in the SEKS setting. The work in [48] shows practical applications of SEKS and employs it to realize secure and searchable audit logs. Chase *et al.* [25] 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 [40] and further enhanced its security in [39] at the cost of large index. The very recent work [24] due to Cash *et al.* simultaneously achieves strong security and high efficiency.

Following the seminal work on PEKS, Abdella *et al.* [1] 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 [44], [36], [17], [37], [45], [19], range search [12], [46], [21], subset search [21], time-scope search [30], [1], and similarity search [29].

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

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 search on encrypted keywords. One may note that the chain in [26] cannot be fully hidden to the server and leaks the frequency of the keywords (see Appendix A for details). To realize an efficient keyword search, Bellare *et al.* [7] introduced deterministic public-key encryption (PKE) and formalized a security notion "as strong as possible" (stronger than one-wayness but weaker than semantic security). A deterministic searchable encryption scheme allows efficient keyword search as if the keywords were not encrypted. Bellare *et al.* [7] 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.* [16] and Boldyreva *et al.* [15]. 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.* [18] 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.

II. Modeling Searchable Public-key Ciphertexts with Hidden Structures

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

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 $(\mathbf{Pri}, \mathbf{Pub})$ be the hidden structure. The encryption algorithm must take $\mathbf{Pri}$ as input, otherwise the hidden relation can not be generated since $\mathbf{Pub}$ does not contain anything about the hidden relations. At the end of the encryption procedure, the $\mathbf{Pri}$ should be updated since a hidden relation is newly generated (but the specific method to update $\mathbf{Pri}$ relies on the specific instance of SPCHS). In addition, SPCHS needs an algorithm to initialize $(\mathbf{Pri}, \mathbf{Pub})$ 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 for guidance to find out the ciphertexts containing the queried keyword with the hidden structure.

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

- $\mathbf{SystemSetup}(1^k, \mathcal{W})$*: Take as input a security parameter* 1^k *and a keyword space* $\mathcal{W}$*, and probabilistically output a pair of master public-and-secret keys* $(\mathbf{PK}, \mathbf{SK})$*, where* $\mathbf{PK}$ *includes the keyword space* $\mathcal{W}$ *and the ciphertext space* $\mathcal{C}$*.*
- $\mathbf{StructureInitialization}(\mathbf{PK})$*: Take as input* $\mathbf{PK}$*, and probabilistically initialize a hidden structure by outputting its private and public parts* $(\mathbf{Pri}, \mathbf{Pub})$*.*
- $\mathbf{StructuredEncryption}(\mathbf{PK}, W, \mathbf{Pri})$*: Take as inputs* $\mathbf{PK}$*, a keyword* $W \in \mathcal{W}$ *and a hidden structure's private part* $\mathbf{Pri}$*, and probabilistically output a keyword searchable ciphertext* C *of keyword* W *with the hidden structure, and update* $\mathbf{Pri}$*.*
- $\mathbf{Trapdoor}(\mathbf{SK}, W)$*: Take as inputs* $\mathbf{SK}$ *and a keyword* $W \in \mathcal{W}$*, and output a keyword search trapdoor* T_W *of* W*.*
- $\mathbf{StructuredSearch}(\mathbf{PK}, \mathbf{Pub}, \mathbb{C}, T_W)$*: Take as inputs* $\mathbf{PK}$*, a hidden structure's public part* $\mathbf{Pub}$*, all keyword searchable ciphertexts* $\mathbb{C}$ *and a keyword search trapdoor* T_W *of keyword* W*, disclose partial relations for guidance to find 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* $\mathbf{Pub}$*, algorithm* $\mathbf{StructuredSearch}(\mathbf{PK}, \mathbf{Pub}, \mathbb{C}, T_W)$ *finds out all ciphertexts of keyword* W *with the hidden structure* $\mathbf{Pub}$*.*

In the application of SPCHS, a receiver runs algorithm $\mathbf{SystemSetup}$ to set up SPCHS. Each sender uploads the public part of his hidden structure and keyword searchable ciphertexts to a server respectively by algorithms $\mathbf{StructureInitialization}$ and $\mathbf{StructuredEncryption}$. The algorithm $\mathbf{Trapdoor}$ allows the receiver to delegates a keyword search trapdoor to the server. Then the server runs algorithm $\mathbf{StructuredSearch}$ 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 $\mathbf{StructuredEncryption}$. 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 $\mathbf{Pri}$. We could let each sender be stateless

by storing his/her $\mathbf{Pri}$ in encrypted form at a server and having each sender download and re-encrypt his/her $\mathbf{Pri}$ for each update of $\mathbf{Pri}$. The similar method also has been suggested by [24].

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, 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 chooses 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 $Adv^{\text{SS-CKSA}}_{SPCHS,\mathcal{A}}$ to win in the following SS-CKSA game:*

- ***Setup Phase**: A challenger sets up the SPCHS scheme by running algorithm* $\mathbf{SystemSetup}$ *to generate a pair of master public-and-secret keys* $(\mathbf{PK}, \mathbf{SK})$*, and initializes N hidden structures by running algorithm* $\mathbf{StructureInitialization}$ *with N times (let* $\mathbf{PSet}$ *be the set of all public parts of these N hidden structures.), finally sends* $\mathbf{PK}$ *and* $\mathbf{PSet}$ *to $\mathcal{A}$;*
- ***Query Phase 1**: $\mathcal{A}$ adaptively issues the following queries multiple times.*
  - *Trapdoor Query $\mathcal{Q}_{Trap}(W)$: Taking as input a keyword $W \in \mathcal{W}$, the challenger outputs the keyword search trapdoor of keyword W;*
  - *Privacy Query $\mathcal{Q}_{Pri}(\mathbf{Pub})$: 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 $\mathcal{Q}_{Enc}(W, \mathbf{Pub})$: Taking as inputs a keyword $W \in \mathcal{W}$ 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**: $\mathcal{A}$ sends two challenge keyword-and-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 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 with **Query Phase 1**. Note that in **Query Phase 1** and **Query Phase 2**, $\mathcal{A}$ can not 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**: $\mathcal{A}$ sends a guess d' to the challenger. We say that $\mathcal{A}$ wins if $d = d'$. And let $Adv^{\text{SS-CKSA}}_{SPCHS,\mathcal{A}} = Pr[d = d'] - \frac{1}{2}$ be the advantage of $\mathcal{A}$ to win in the above game.*

A weaker security definition of SPCHS is the selective-keyword security. We referred 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 $Adv^{\text{SS-sK-CKSA}}_{SPCHS,\mathcal{A}}$ denote the advantage of adversary $\mathcal{A}$ to win in this game.

III. A Simple SPCHS Scheme from Scratch

Let $\gamma \xleftarrow{\$} \Re$ denote an element γ 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 $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ [43], [32] is an efficiently computable and non-degenerate function, with the bilinearity property $\hat{e}(g^a, g^b) = \hat{e}(g, g)^{ab}$, where $(a, b) \xleftarrow{\$} \mathbb{Z}_q^*$ 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.

- $\mathbf{SystemSetup}(1^k, \mathcal{W})$: Take as input 1^k and the keyword space $\mathcal{W}$, compute $(q, \mathbb{G}, \mathbb{G}_1, g, \hat{e}) = \mathbf{BGen}(1^k)$, pick $s \xleftarrow{\$} \mathbb{Z}_q^*$, set $P = g^s$, choose a cryptographic hash function $H : \mathcal{W} \to \mathbb{G}$, set the ciphertext space $\mathcal{C} \subseteq \mathbb{G}_1 \times \mathbb{G} \times \mathbb{G}_1$, and finally output the master public key $\mathbf{PK} = (q, \mathbb{G}, \mathbb{G}_1, g, \hat{e}, P, H, \mathcal{W}, \mathcal{C})$, and the master secret key $\mathbf{SK} = s$.
- $\mathbf{StructureInitialization}(\mathbf{PK})$: Take as input $\mathbf{PK}$, pick $u \xleftarrow{\$} \mathbb{Z}_q^*$, and initialize a hidden structure by outputting a pair of private-and-public parts $(\mathbf{Pri} = (u), \mathbf{Pub} = g^u)$. Note that $\mathbf{Pri}$ here is a variable list formed as $(u, \{(W, Pt[u, W]) | W \in \mathcal{W}\})$, which is initialized as (u).
- $\mathbf{StructuredEncryption}(\mathbf{PK}, W, \mathbf{Pri})$: Take as inputs $\mathbf{PK}$, a keyword $W \in \mathcal{W}$, a hidden structure's private part $\mathbf{Pri}$, pick $r \xleftarrow{\$} \mathbb{Z}_q^*$ and do the following steps:
  1) Seek $(W, Pt[u, W])$ by W in $\mathbf{Pri}$;
  2) If not exists, insert $(W, Pt[u, W] \xleftarrow{\$} \mathbb{G}_1)$ to $\mathbf{Pri}$, and output the keyword searchable ciphertext $C = (\hat{e}(P, H(W))^u, g^r, \hat{e}(P, H(W))^r \cdot Pt[u, W])$;
  3) Otherwise, pick $R \xleftarrow{\$} \mathbb{G}_1$, 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;

- $\mathbf{Trapdoor}(\mathbf{SK}, W)$: Take as inputs $\mathbf{SK}$ and a keyword $W \in \mathcal{W}$, and output a keyword search trapdoor $T_W = H(W)^s$ of keyword W.
- $\mathbf{StructuredSearch}(\mathbf{PK}, \mathbf{Pub}, \mathbb{C}, T_W)$: Take as inputs $\mathbf{PK}$, a hidden structure's public part $\mathbf{Pub}$, all keyword searchable ciphertexts $\mathbb{C}$ (let $\mathbb{C}[i]$ denote one ciphertext of $\mathbb{C}$, and this ciphertext can be parsed as $(\mathbb{C}[i,1], \mathbb{C}[i,2], \mathbb{C}[i,3]) \in \mathbb{G}_1 \times \mathbb{G} \times \mathbb{G}_1$) and a keyword trapdoor T_W of keyword W, set $\mathbb{C}' = \phi$, and do the following steps:
  1) Compute $Pt' = \hat{e}(\mathbf{Pub}, T_W)$;
  2) Seek a ciphertext $\mathbb{C}[i]$ having $\mathbb{C}[i,1] = Pt'$; if exists, add $\mathbb{C}[i]$ into $\mathbb{C}'$;
  3) If no matching ciphertext is found, output $\mathbb{C}'$;
  4) Compute $Pt' = \hat{e}(\mathbb{C}[i,2], T_W)^{-1} \cdot \mathbb{C}[i,3]$, go to step 2;

Figure 2 shows a hidden star-like structure, which is generated by the SPCHS instance. When running algorithm $\mathbf{StructuredSearch}(\mathbf{PK}, \mathbf{Pub}, \mathbb{C}, T_{W_i})$, 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 $(Pt[u, W_i], g^r, \hat{e}(P, H(W_i))^r \cdot R)$. So on and so forth, 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. Coarsely, algorithm $\mathbf{StructuredSearch}$ 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 $\mathbf{StructuredEncryption}$), 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 $\mathbf{StructuredSearch}$ only can disclose the values of Pt', which are corresponding to the queried keyword. Formally, we have Theorem 1 on consistency.

Theorem 1. *Suppose the hash function H is collision-free, 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.*

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 $\mathbf{StructuredSearch}(\mathbf{PK}, \mathbf{Pub}, \mathbb{C}, T_{W_i})$ only finds out all ciphertexts of keyword W_i with the hidden structure $\mathbf{Pub}$. Note that $P = g^s$.

Algorithm $\mathbf{StructuredSearch}(\mathbf{PK}, \mathbf{Pub}, \mathbb{C}, T_{W_i})$ 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 $\mathbf{StructuredSearch}(\mathbf{PK}, \mathbf{Pub}, \mathbb{C}, T_{W_i})$ 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 $(\hat{e}(P, H(W_i))^u, g^r, \hat{e}(P, H(W_i))^r \cdot Pt[u, W_i])$ is found with the overwhelming probability in this step.

Then algorithm $\mathbf{StructuredSearch}(\mathbf{PK}, \mathbf{Pub}, \mathbb{C}, T_{W_i})$ discloses $Pt[u, W_i]$ from the ciphertext

$$(\hat{e}(P, H(W_i))^u, g^r, \hat{e}(P, H(W_i))^r \cdot Pt[u, W_i])$$

by computing

$$\begin{aligned} Pt' &= Pt[u, W_i] \\ &= \hat{e}(g^r, T_{W_i})^{-1} \cdot \hat{e}(P, H(W_i))^r \cdot Pt[u, W_i] \end{aligned}$$

Recall that in algorithm $\mathbf{StructuredEncryption}$, $Pt[u, W_i]$ 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 $\mathbf{Pub}$. So when algorithm $\mathbf{StructuredSearch}(\mathbf{PK}, \mathbf{Pub}, \mathbb{C}, T_{W_i})$ goes back to its second step, only the ciphertext

$$(Pt[u, W_i], g^r, \hat{e}(P, H(W_i))^r \cdot R)$$

is found with the overwhelming probability.

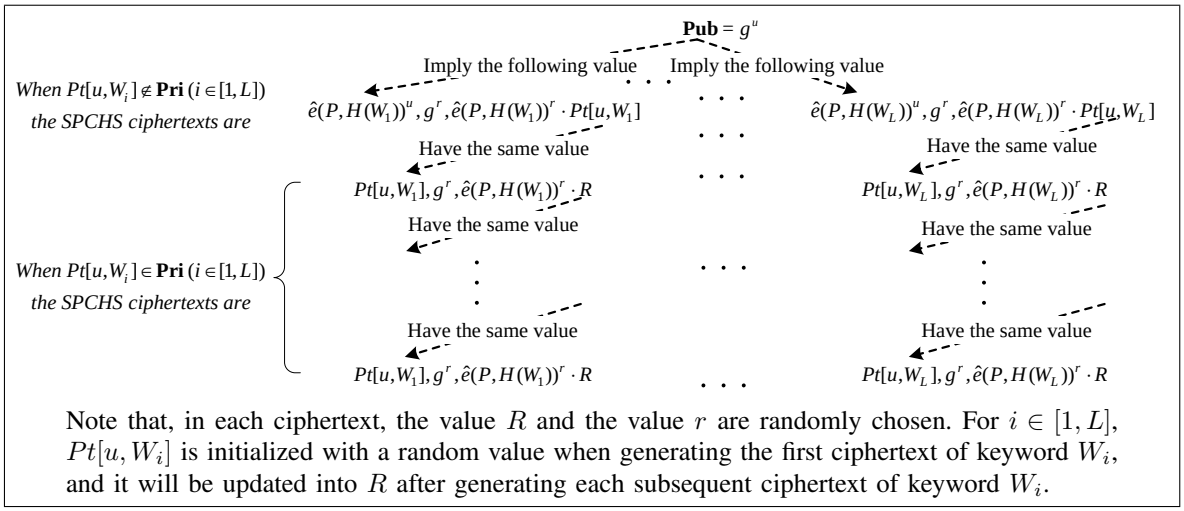
So on and so forth, algorithm $\mathbf{StructuredSearch}(\mathbf{PK}, \mathbf{Pub}, \mathbb{C}, T_{W_i})$ only finds out all ciphertexts of keyword W_i with the hidden structure $\mathbf{Pub}$, except with a negligible probability in the security parameter k . And the algorithm will stop, since the random value R contained in the last found ciphertext is un-matching with any other ciphertext's first part. ■

Semantic Security. The SS-CKSA security of the above SPCHS scheme relies on the DBDH assumption in $\mathbf{BGen}(1^k)$. The definition of DBDH assumption [6] is as follows.

Definition 3 (The DBDH Assumption)**.** *The DBDH problem in $\mathbf{BGen}(1^k) = (q, \mathbb{G}, \mathbb{G}_1, g, \hat{e})$ 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 $(a, b, c, y) \xleftarrow{\$} \mathbb{Z}_q^{*4}$. 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 $\mathbf{BGen}(1^k)$, 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 $\mathbf{BGen}(1^k)$. Formally we have Theorem 2 whose proof can be found in Appendix B.

Theorem 2. *Let the hash function H be modeled as the random oracle $\mathcal{Q}_H(\cdot)$. Suppose there are at most $N \in \mathbb{N}$*



Note that, in each ciphertext, the value R and the value r are randomly chosen. For $i \in [1, L]$, $Pt[u, W_i]$ is initialized with a random value when generating the first ciphertext of keyword W_i, and it will be updated into R after generating each subsequent ciphertext of keyword W_i.

Figure 2: A Hidden Star-like Structure Generated By The Above SPCHS Instance

hidden structures, and a PPT adversary $\mathcal{A}$ wins in the SS-CKSA game of the above SPCHS instance with advantage $Adv^{SS\text{-}CKSA}_{SPCHS,\mathcal{A}}$, 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 $\mathbf{BGen}(1^k)$ with advantage

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

where e is the base of natural logarithms.

Search Performance. 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 $W \in \mathcal{W}$. 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 $n = \sum_{W \in \mathcal{W}} n_w$ and $n_w = \sum_{i=1}^{n_s} n_{w,i}$, it implies that $n_s \ll n_w \ll n$. Thus the above SPCHS instance allows much more efficient search on encrypted keyword than existing PEKS schemes, which require $O(n)$ search complexity.

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's review the concept of IBE and its Anon-SS-ID-CPA security.

Definition 4 (IBE [14])**.** *IBE consists of four algorithms:*

- $\mathbf{Setup}_{IBE}(1^k, \mathcal{ID}_{IBE})$*: Take as inputs a security parameter 1^k and an identity space $\mathcal{ID}_{IBE}$, and probabilistically output the master public-and-secret-key pair $(\mathbf{PK}_{IBE}, \mathbf{SK}_{IBE})$, where $\mathbf{PK}_{IBE}$ includes the message space $\mathcal{M}_{IBE}$, the ciphertext space $\mathcal{C}_{IBE}$ and the identity space $\mathcal{ID}_{IBE}$.*
- $\mathbf{Extract}_{IBE}(\mathbf{SK}_{IBE}, ID)$*: Take as inputs $\mathbf{SK}_{IBE}$ and an identity $ID \in \mathcal{ID}_{IBE}$, and output a decryption key $\tilde{S}_{ID}$ of ID.*
- $\mathbf{Enc}_{IBE}(\mathbf{PK}_{IBE}, ID, M)$*: Take as inputs $\mathbf{PK}_{IBE}$, an identity $ID \in \mathcal{ID}_{IBE}$ and a message M, and probabilistically output a ciphertext $\tilde{C}$.*
- $\mathbf{Dec}_{IBE}(\tilde{S}_{ID'}, \tilde{C})$*: 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 $\tilde{C} = \mathbf{Enc}_{IBE}(\mathbf{PK}_{IBE}, ID, M)$ and $\tilde{S}_{ID'} = \mathbf{Extract}_{IBE}(\mathbf{SK}_{IBE}, ID')$, $\mathbf{Dec}_{IBE}(\tilde{S}_{ID'}, \tilde{C}) = M$ 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 as follows.

Definition 5 (Anon-SS-ID-CPA security of IBE [2])**.** *An IBE scheme is Anon-SS-ID-CPA secure if any PPT adversary $\mathcal{B}$ has only a negligible advantage $Adv^{Anon\text{-}SS\text{-}ID\text{-}CPA}_{IBE,\mathcal{B}}$ to win in the following Anon-SS-ID-CPA game:*

- ***Setup Phase**: A challenger sets up the IBE scheme by running algorithm $\mathbf{Setup}_{IBE}$ to generate the master public-and-secret-keys pair $(\mathbf{PK}_{IBE}, \mathbf{SK}_{IBE})$, and sends $\mathbf{PK}_{IBE}$ to $\mathcal{B}$.*
- ***Query Phase 1**: Adversary $\mathcal{B}$ adaptively issues the following query multiple times.*
  - *Decryption Key Query $\mathcal{Q}^{IBE}_{DK}(ID)$: Taking as*

input an identity $ID \in \mathcal{ID}_{IBE}$, the challenger outputs the decryption key of identity ID;

- ***Challenge Phase**: Adversary $\mathcal{B}$ sends two challenge identity-and-message pairs (ID_0^*, M_0^*) and (ID_1^*, M_1^*) to the challenger; The challenger picks $\tilde{d} \xleftarrow{\$} \{0,1\}$, sends the challenge IBE ciphertext $\tilde{C}_{\tilde{d}}^* = \mathbf{Enc}_{IBE}(\mathbf{PK}_{IBE}, ID_{\tilde{d}}^*, M_{\tilde{d}}^*)$ to $\mathcal{B}$.*
- ***Query Phase 2**: This phase is the same with **Query Phase 1**. Note that in **Query Phase 1** and **Query Phase 2**, $\mathcal{B}$ can not query the decryption key corresponding to the challenge identity ID_0^* or ID_1^*.*
- ***Guess Phase**: Adversary $\mathcal{B}$ sends a guess $\tilde{d}'$ to the challenger. We say that $\mathcal{B}$ wins if $\tilde{d}' = \tilde{d}$. And let $Adv_{IBE,\mathcal{B}}^{Anon\text{-}SS\text{-}ID\text{-}CPA} = Pr[\tilde{d}' = \tilde{d}] - \frac{1}{2}$ 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 [38]. Their difference only appears in algorithm $\mathbf{Encaps}_{\text{IBKEM}}$. In order to highlight that the generator of an IBKEM encapsulation knows the chosen random value used in algorithm $\mathbf{Encaps}_{\text{IBKEM}}$, we take the random value as an input of the algorithm.

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

- $\mathbf{Setup}_{IBKEM}(1^k, \mathcal{ID}_{IBKEM})$*: Take as inputs a security parameter 1^k and an identity space $\mathcal{ID}_{IBKEM}$, and probabilistically output the master public-and-secret-keys pair $(\mathbf{PK}_{IBKEM}, \mathbf{SK}_{IBKEM})$, where $\mathbf{PK}_{IBKEM}$ includes the identity space $\mathcal{ID}_{IBKEM}$, the encapsulated key space $\mathcal{K}_{IBKEM}$ and the encapsulation space $\mathcal{C}_{IBKEM}$.*
- $\mathbf{Extract}_{IBKEM}(\mathbf{SK}_{IBKEM}, ID)$*: Take as inputs $\mathbf{SK}_{IBKEM}$ and an identity $ID \in \mathcal{ID}_{IBKEM}$, and output a decryption key $\hat{S}_{ID}$ of ID.*
- $\mathbf{Encaps}_{IBKEM}(\mathbf{PK}_{IBKEM}, ID, r)$*: Take as inputs $\mathbf{PK}_{IBKEM}$, 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.*
- $\mathbf{Decaps}_{IBKEM}(\hat{S}_{ID'}, \hat{C})$*: Take as inputs the decryption key $\hat{S}_{ID'}$ of identity ID' and an encapsulation $\hat{C}$, and output an encapsulated key or $\perp$, 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 characters: 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 $\mathbf{FIM}$ that for any $(\hat{K}, \hat{C}) = \mathbf{Encaps}_{IBKEM}(\mathbf{PK}_{IBKEM}, ID, r)$, the function $\mathbf{FIM}$ 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 $ID' \in \mathcal{ID}_{IBKEM}$ 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 with the traditional version [38] 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 $\mathcal{B}$ has only a negligible advantage $Adv_{IBKEM,\mathcal{B}}^{Anon\text{-}SS\text{-}ID\text{-}CPA}$ to win in the following Anon-SS-ID-CPA game:*

- ***Setup Phase**: A challenger sets up the IBKEM scheme by running algorithm $\mathbf{Setup}_{IBKEM}$ to generate the master public-and-secret-keys pair $(\mathbf{PK}_{IBKEM}, \mathbf{SK}_{IBKEM})$, and sends $\mathbf{PK}_{IBKEM}$ to $\mathcal{B}$.*
- ***Query Phase 1**: $\mathcal{B}$ 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 $\hat{d} \xleftarrow{\$} \{0,1\}$, computes $(\hat{K}_0^*, \hat{C}_0^*) = \mathbf{Encaps}_{IBKEM}(PK_{IBKEM}, ID_0^*, r_0)$ and $(\hat{K}_1^*, \hat{C}_1^*) = \mathbf{Encaps}_{IBKEM}(PK_{IBKEM}, ID_1^*, r_1)$, 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 with **Query Phase 1**. Note that in **Query Phase 1** and **Query Phase 2**, $\mathcal{B}$ can not query the decryption keys both of the challenge identities ID_0^* and ID_1^*.*
- ***Guess Phase**: $\mathcal{B}$ sends a guess $\hat{d}'$ to the challenger. We say that $\mathcal{B}$ wins if $\hat{d}' = \hat{d}$. And let $Adv_{IBKEM,\mathcal{B}}^{Anon\text{-}SS\text{-}ID\text{-}CPA} = Pr[\hat{d}' = \hat{d}] - \frac{1}{2}$ 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 $\hat{C}_{\hat{d}}^*$), 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 selective-identity 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.

- $\mathbf{SystemSetup}(1^k, \mathcal{W})$: Take as inputs a security parameter 1^k and the keyword space $\mathcal{W}$, 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 $\mathcal{K}_{\text{IBKEM}} = \mathcal{M}_{\text{IBE}}$.
- $\mathbf{StructureInitialization}(\mathbf{PK})$: Take as input $\mathbf{PK}$, arbitrarily pick a keyword $W \in \mathcal{W}$ 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 $(u, \{(W, Pt[u, W]) | W \in \mathcal{W}\})$, which is initialized as (u).
  (In the above, an IBKEM encapsulation and its related random value are respectively taken as the public-and-private parts of a hidden structure. To generate these two parts , an arbitrary keyword have to be chosen to run algorithm $\mathbf{Encaps}_{\text{IBKEM}}$.)
- $\mathbf{StructuredEncryption}(\mathbf{PK}, W, \mathbf{Pri})$: Take as inputs $\mathbf{PK}$, a keyword $W \in \mathcal{W}$, a hidden structure's private part $\mathbf{Pri}$, and do the following steps:
  1) Seek $(W, Pt[u, W])$ by W in $\mathbf{Pri}$;
  2) If not exists, insert $(W, Pt[u, W] \xleftarrow{\$} \mathcal{M}_{\text{IBE}})$ to $\mathbf{Pri}$, 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 $R \xleftarrow{\$} \mathcal{M}_{\text{IBE}}$, set $C = (Pt[u, W], \mathbf{Enc}_{\text{IBE}}(\mathbf{PK}_{\text{IBE}}, W, R))$, update $Pt[u, W] = R$, and output the keyword searchable ciphertext C;
- $\mathbf{Trapdoor}(\mathbf{SK}, W)$: Take as inputs $\mathbf{SK}$ and a keyword $W \in \mathcal{W}$, 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.
- $\mathbf{StructuredSearch}(\mathbf{PK}, \mathbf{Pub}, \mathbb{C}, T_W)$: Take as inputs $\mathbf{PK}$, a hidden structure's public part $\mathbf{Pub}$, 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 $Pt' = \mathbf{Decaps}_{\text{IBKEM}}(\hat{S}_W, \mathbf{Pub})$;
  2) Seek a ciphertext $\mathbb{C}[i]$ having $\mathbb{C}[i, 1] = Pt'$; If exits, add $\mathbb{C}[i]$ into $\mathbb{C}'$;
  3) If no matching ciphertext is found, output $\mathbb{C}'$;
  4) Compute $Pt' = \mathbf{Dec}_{\text{IBE}}(\tilde{S}_{ID'}, \mathbb{C}[i, 2])$, go to step 2;

Figure 3 shows a hidden star-like structure generated by the generic SPCHS construction. When running algorithm $\mathbf{StructuredSearch}(\mathbf{PK}, \mathbf{Pub}, \mathbb{C}, T_{W_i})$, the full-identity malleability of IBKEM allows the algorithm to disclose the value $\mathbf{FIM}(W_i, u)$ by computing $\mathbf{FIM}(W_i, u) = \mathbf{Decaps}_{\text{IBKEM}}(\hat{S}_{W_i}, \mathbf{Pub})$ and find out the ciphertext $(\mathbf{FIM}(W_i, u), \mathbf{Enc}_{\text{IBE}}(\mathbf{PK}_{\text{IBE}}, W_i, Pt[u, W_i]))$. Then the consistency of IBE allows the algorithm to disclose $Pt[u, W_i]$ by decrypting $\mathbf{Enc}_{\text{IBE}}(\mathbf{PK}_{\text{IBE}}, W_i, Pt[u, W_i])$ and find out the ciphertext $(Pt[u, W_i], \mathbf{Enc}_{\text{IBE}}(\mathbf{PK}_{\text{IBE}}, W_i, R))$. So on and so forth, the consistency of IBE allows the algorithm to find out the rest ciphertexts of keyword W_i with the hidden star-like structure, and stop the search if no more ciphertext is found.

Consistency. When running the above algorithm $\mathbf{StructuredSearch}(\mathbf{PK}, \mathbf{Pub}, \mathbb{C}, T_W)$, the consistency and full-identity malleability of IBKEM assures that $\mathbf{FIM}(W, u) = \mathbf{Decaps}_{\text{IBKEM}}(\hat{S}_W, \mathbf{Pub})$ holds. The collision-freeness of IBKEM assures that only one ciphertext containing keyword W has the value $\mathbf{FIM}(W, u)$ as its first part. Therefore the algorithm can find out the first ciphertext of keyword W with the hidden structure $\mathbf{Pub}$. Then the consistency of IBE allows the algorithm $\mathbf{StructuredSearch}$ to find out the rest ciphertexts containing keyword W with the hidden structure $\mathbf{Pub}$. Formally we have Theorem 3.

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

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 $\mathbf{StructuredSearch}(\mathbf{PK}, \mathbf{Pub}, \mathbb{C}, T_{W_i})$ only finds out all ciphertexts of keyword W_i with the hidden structure $\mathbf{Pub}$, where $\hat{S}_{W_i} = \mathbf{Extract}_{\text{IBKEM}}(\mathbf{SK}_{\text{IBKEM}}, W_i)$, $\tilde{S}_{W_i} = \mathbf{Extract}_{\text{IBE}}(\mathbf{SK}_{\text{IBE}}, W_i)$, $\hat{C}$ is from $(\hat{K}, \hat{C}) = \mathbf{Encaps}_{\text{IBKEM}}(\mathbf{PK}_{\text{IBKEM}}, W, u)$, keyword W is arbitrarily chosen in $\mathcal{W}$, and u is a random value.

Algorithm $\mathbf{StructuredSearch}(\mathbf{PK}, \mathbf{Pub}, \mathbb{C}, T_{W_i})$ computes $Pt' = \mathbf{Decaps}_{\text{IBKEM}}(\hat{S}_{W_i}, \mathbf{Pub})$ in its first step. According to the full-identity malleability of IBKEM in Definition 7, we have

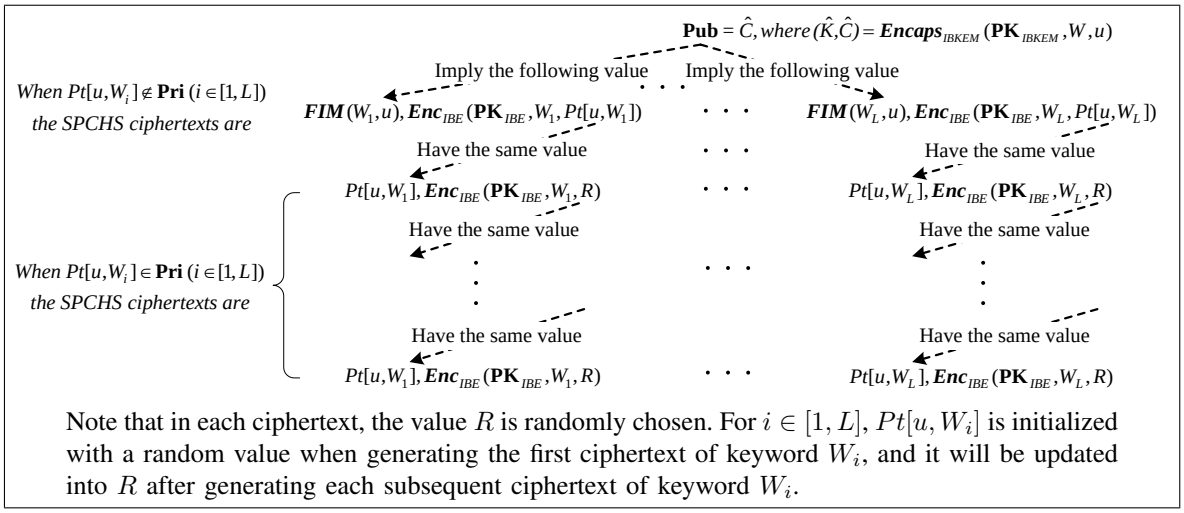


Figure 3: A Hidden Star-like Structure Generated By The Generic SPCHS Construction

$\mathbf{FIM}(W_i, u) = \mathbf{Decaps}_{\text{IBKEM}}(\hat{S}_{W_i}, \mathbf{Pub})$. So algorithm $\mathbf{StructuredSearch}(\mathbf{PK}, \mathbf{Pub}, \mathbb{C}, T_{W_i})$ finds out the ciphertext $(\mathbf{FIM}(W_i, u), \mathbf{Enc}_{\text{IBE}}(\mathbf{PK}_{\text{IBE}}, W_i, Pt[u, W_i]))$ 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 $(\neq W_i)$ to meet $\mathbf{FIM}(W_i, u) = \mathbf{FIM}(W_j, u)$, and no hidden structure $\mathbf{Pub}'$ $(\neq \mathbf{Pub})$ to meet $\mathbf{FIM}(W_i, u) = \mathbf{FIM}(W_i, u')$, where $\mathbf{Pub}'$ is generated by algorithm $\mathbf{StructureInitialization}(\mathbf{PK})$ with the random value u'. So only the ciphertext $(\mathbf{FIM}(W_i, u), \mathbf{Enc}_{\text{IBE}}(\mathbf{PK}_{\text{IBE}}, W_i, Pt[u, W_i]))$ is found in this step, except with a negligible probability in the security parameter k.

Then, according to the consistency of IBE, algorithm $\mathbf{StructuredSearch}(\mathbf{PK}, \mathbf{Pub}, \mathbb{C}, T_{W_i})$ can decrypt out $Pt[u, W_i]$ by algorithm $\mathbf{Dec}_{\text{IBE}}(\hat{S}_{W_i}, \mathbf{Enc}_{\text{IBE}}(\mathbf{PK}_{\text{IBE}}, W_i, Pt[u, W_i]))$.

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

So on and so forth, algorithm $\mathbf{StructuredSearch}(\mathbf{PK}, \mathbf{Pub}, \mathbb{C}, T_{W_i})$ 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 with any other ciphertext's first part. ■

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-sID-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 Appendix C.

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 $Adv^{\text{SS-sK-CKSA}}_{SPCHS,\mathcal{A}}$. 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 $\frac{1}{4N} \cdot Adv^{\text{SS-sK-CKSA}}_{SPCHS,\mathcal{A}}$.*

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 [3]. An IBKEM instance is VRF-suitable if it provides *unique decapsulation*. It 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 run of the key extraction algorithm. It is clear that our proposed collision-free full-identity 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 Appendix D, we prove that the VRF-suitable IBKEM instance proposed in Appendix A.2 of [3] is collision-free full-identity malleable. Even though this IBKEM scheme has the traditional Anon-SS-ID-CPA security, we futher 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 [31], Freire *et al.* utilized the "approximation" of multilinear maps [35]

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 [14] and reconstructed this IBE scheme in the multilinear setting. They finally constructed a new IBE scheme with the 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 the 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. It 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 [31])**.** *An ℓ-group system in the multilinear setting consists of ℓ cyclic groups $\mathbb{G}_1, \cdots, \mathbb{G}_\ell$ 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 $i, j \geq 1$ with $i+j \leq \ell$. 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 $\hat{e}(h_1, \cdots, h_j) = \hat{e}(h_1, \hat{e}(h_2, \cdots, \hat{e}(h_{j-1}, h_j) \cdots))$ for $h_j \in \mathbb{G}_{i_j}$ and $i = (i_1 + i_2 + \cdots + i_j) \leq \ell$. 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 ℓ-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 ℓ-MDDH Assumption [31])**.** *Given $(g, g^{x_1}, \cdots, g^{x_{\ell+1}})$ (for $g \xleftarrow{\$} \mathbb{G}_1$ and uniform exponents x_i), the ℓ-MDDH assumption is that the element $\hat{e}(g^{x_1}, \cdots, g^{x_\ell})^{x_{\ell+1}} \in \mathbb{G}_\ell$ is computationally indistinguishable from a uniform $\mathbb{G}_\ell$-element.*

Definition 11 (Group hash function [31])**.** *A group hash function $\mathbf{H}$ into $\mathbb{G}$ consists of two polynomial-time algorithms: the probabilistic algorithm $\mathbf{HGen}(1^k)$ output 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 [31])**.** *Assume an ℓ'-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 $\mathbf{TGen}$ and $\mathbf{TEval}$ as follows.*

- $\mathbf{TGen}(1^k, c_1, \cdots, c_l, h)$ *(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* $\mathbf{HGen}$.
- $\mathbf{TEval}(td, X)$ *(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 $\{X_i\}_i \bigcap \{Z_j\}_j = \emptyset$, $P_{hk,\{X_i\},\{Z_j\}} = Pr[(a_{X_1} = \cdots = a_{X_m} = 0) \wedge (a_{Z_1}, \cdots, a_{X_n} \neq 0)] \geq 1/p(k)$, where the probability is over possible trapdoors td output by $\mathbf{TGen}$ along with the given hk. Furthermore, we require that $P_{hk,\{X_i\},\{Z_j\}}$ is close to statistically independent of hk. (Formally, $|P_{hk,\{X_i\},\{Z_j\}} - P_{hk',\{X_i\},\{Z_j\}}| \leq v(k)$ for all hk, hk' in the range of $\mathbf{TGen}$, all $\{X_i\}, \{Z_j\}$, and negligible $v(k)$.)*

We say that $\mathbf{H}$ is a (poly, n)-MPHF if it is a $(q(k), n)$-MPHF for every polynomial $q(k)$. Note that $\mathbf{TEval}$ 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.

- $\mathbf{Setup}_{\text{IBKEM}}(1^k, \mathcal{ID}_{\text{IBKEM}})$: Take as input a security parameter 1^k and the identity space $\mathcal{ID}_{\text{IBKEM}}$, 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 $(\mathsf{poly}, 2)$-MPHF $\mathbf{H}$ into $\mathbb{G}_\ell$ and $hk \leftarrow \mathbf{HGen}(1^k)$, choose $h \xleftarrow{\$} \mathbb{G}_1$ and $x \xleftarrow{\$} \mathbb{Z}_p$, 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 $\mathbf{SK}_{\text{IBKEM}} = (hk, x)$.
- $\mathbf{Extract}_{\text{IBKEM}}(\mathbf{SK}_{\text{IBKEM}}, ID)$: Take as inputs $\mathbf{SK}_{\text{IBKEM}}$ and an identity $ID \in \mathcal{ID}_{\text{IBKEM}}$, and output a decryption key $\hat{S}_{ID} = \mathbf{H}_{hk}(ID)^x$ of ID.
- $\mathbf{Encaps}_{\text{IBKEM}}(\mathbf{PK}_{\text{IBKEM}}, ID, r)$: Take as inputs $\mathbf{PK}_{\text{IBKEM}}$, an identity $ID \in \mathcal{ID}_{\text{IBKEM}}$ 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 $\hat{C} = h^r$.
- $\mathbf{Decaps}_{\text{IBKEM}}(\hat{S}_{ID'}, \hat{C})$: 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 Definition 9 and 11, it is very easy to verify the consistency of the above IBKEM scheme.

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

Anon-SS-ID-CPA Security. In [31], Freire *et al.* utilized a $(\mathsf{poly}, 1)$-MPHF to construct a standard-model version of the BF IBE scheme with the SS-ID-CPA security. On the contrary, we utilizes a $(\mathsf{poly}, 2)$-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 Appendix E.

Theorem 5. *Assume the above IBKEM scheme is implemented in an $(\ell+1)$-group system, and with a $(\mathsf{poly}, 2)$-MPHF $\mathbf{H}$ 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 the 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 to find 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 structures. 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 content searchable encryption is useful in practice, e.g., to filter the encrypted spams. Specially, by forming a hidden tree-like 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.

References

[1] 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)

[2] Abdalla M., Bellare M., Neven G.: Robust encryption. In: Micciancio G. (ed.) TCC 2010. LNCS, vol. 5978, pp. 480497. Springer, Heidelberg (2010)

[3] 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)

[4] 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)

[5] 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)

[6] 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)

[7] 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)

[8] 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)

[9] Bellovin S. M., Cheswick W.R.: Privacy-Enhanced Searches Using Encrypted Bloom Filters. Cryptography ePrint Archive, Report 2004/022 (2004)

[10] 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)

[11] 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)

[12] 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)

[13] 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)

[14] 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)

[15] 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)

[16] 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)

[17] 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)

[18] 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)

[19] 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)

[20] 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)

[21] 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)

[22] Chow Sherman S.M.: Removing Escrow from Identity-Based Encryption. In:Jarecki S. and Tsudik G. (eds.) PKC '09. LNCS, vol. 5443, pp: 256-276. Springer, Heidelberg, (2009)

[23] 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)
[24] 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.
[25] 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)
[26] 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)
[27] 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)
[28] 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)
[29] 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)
[30] 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)
[31] Freire Eduarda S.V., Hofheinz Dennis, Paterson Kenneth G., Striecks Christoph: Programmable Hash Functions in the Multilinear Setting. In: Canetti R., Garay Juan A. (eds.) Advances in Cryptology - CRYPTO 2013. LNCS, vol. 8042, pp. 513-530. Springer, Heidelberg (2013)
[32] 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)
[33] Goh E.-J.: Secure Indexes. Cryptography ePrint Archive, Report 2003/216 (2003)
[34] Gentry C.: Practical Identity-Based Encyrption Without Random Oracles. In: Vaudenay S. (Ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp.445-464. Springer, Heidelberg (2006)
[35] 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)
[36] 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)
[37] 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 Tatsuaki, Okamoto E. and Okamoto Takeshi (eds.) Pairing 2007. LNCS, vol. 4575, pp. 2-22. Springer, Heidelberg (2007)
[38] 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)
[39] 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)
[40] Kamara S., Papamanthou C., Roeder T.: Dynamic searchable symmetric encryption. In ACM Conference on Computer and Communications Security, pp. 965976 (2012)
[41] Libert B., Paterson Kenneth G., Quaglia Elizabeth 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)
[42] Li J., Wang Q., Wang C., Cao N., Ren K., Lou W.: Fuzzy Keyword Search over Encrypted Data in Cloud Computing. In: IEEE INFOCOM 2010, pp. 1-5. (2010)
[43] Menezes A. J., Okamoto T., Vanstone S. A.: Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field. IEEE Transactions on Information Theory, vol. 39, no. 5, pp. 1639-1646 (1993)
[44] 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)
[45] 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)
[46] 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)
[47] Song D. X., Wagner D., Perrig A.: Practical techniques for searches on encrypted data. In: IEEE S&P 2000, pp. 44-55. IEEE (2000)
[48] Waters B. R., Balfanz D., Durfee G., Smetters D. K.: Building an Encrypted and Searchable Audit Log. In: NDSS 2004.
[49] 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)

APPENDIX A
THE ANALYSIS ON THE WORK [26]

For a sender, he generates the searchable ciphertexts of any keyword $W_i \in \mathcal{W}$ by the following steps:

1) At 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 ask the server to store $E(K_i^1, P_i^2||K_i^2||\mathcal{P}_i^1)$ in the position P_i^1 and store a flag in the 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, finally output the IBE ciphertext and C_2, where symmetric-key K_i^1 and C_2 are randomly chosen. Algorithm $E(K_i^1, P_i^2||K_i^2||\mathcal{P}_i^1)$ denotes using symmetric-key K_i^1 to encrypt $P_i^2||K_i^2||\mathcal{P}_i^1$, where symmetric-key K_i^2 is randomly chosen, and $\mathcal{P}_i^1$ denotes the parameters for private information retrieve (it will be used to retrieve the corresponding data when the keyword W_i is queried).
2) At 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 ask the server to store $E(K_i^2, P_i^3||K_i^3||\mathcal{P}_i^2)$ in the position P_i^2 and store the flag in the position P_i^3 .
3) The subsequent encryptions of keyword W_i are similar with the step 2.

Figure 4: The Procedure to Generate Keyword Searchable Ciphertexts in [26].

In Fig. 4, we first review how to generate keyword searchable ciphertexts by the work [26] 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. 4, the server trivially knows the relation between ciphertexts $PEKS(Pub, W_i, K_i^1||P_i^1)$ and $E(K_i^1, P_i^2||K_i^2||\mathcal{P}_i^1)$, and knows that if a subsequent ciphertext is stored in the position P_2, this subsequent ciphertext is related to $E(K_i^1, P_i^2||K_i^2||\mathcal{P}_i^1)$. So in the second step, the server knows the relation between ciphertexts $E(K_i^1, P_i^2||K_i^2||\mathcal{P}_i^1)$ and $E(K_i^2, P_i^3||K_i^3||\mathcal{P}_i^2)$, and knows that if another subsequent ciphertext is stored in the position P_3, this subsequent ciphertext is related to $E(K_i^2, P_i^3||K_i^3||\mathcal{P}_i^2)$. 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.

For a sender, he generates the searchable ciphertexts of any keyword $W_i \in \mathcal{W}$ by the following steps:

1) At the setup phase, he uploads
$$\{PEKS(Pub, W_i, K_i^1||P_i^1)|i \in [1, |\mathcal{W}|]\}$$
to the server, where $|\mathcal{W}|$ denotes the size of keyword space $\mathcal{W}$.
2) At 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 ask the server to store $E(K_i^1, P_i^2||K_i^2||\mathcal{P}_i^1)$ in the position P_i^1.
3) At 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 ask the server to store $E(K_i^2, P_i^3||K_i^3||\mathcal{P}_i^2)$ in the position P_i^2.
4) The subsequent encryptions of keyword W_i are similar with the step 3.

Figure 5: The New Procedure to Generate Keyword Searchabel Ciphertexts for [26].

In order to keep the privacy of 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. 5. 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 hided. But it seems that this method makes the performance impractical, 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.

APPENDIX B
PROOF OF THEOREM 2

The proof of Theorem 2 is as follows.

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 $\mathbf{BGen}(1^k)$ with advantage approximately $\frac{27}{(e\cdot q_t\cdot q_p)^3}\cdot Adv^{\text{SS-CKSA}}_{SPCHS,\mathcal{A}}$.

Let $Coin \xleftarrow{\sigma} \{0,1\}$ denote the operation that picks $Coin \in \{0,1\}$ according to the probability $Pr[Coin=1]=\sigma$ (the specified value of σ 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 $\hat{e}(g,g)^{abc}$ or $\hat{e}(g,g)^y$) and the keyword space $\mathcal{W}$, and does the following steps:
  1) Initializes the three lists $\mathbf{Pt}=\emptyset\subseteq\mathcal{W}\times\mathbb{G}\times\mathbb{G}_1$, $\mathbf{SList}=\emptyset\subseteq\mathbb{G}\times\mathbb{Z}_q^*\times\{0,1\}$ and $\mathbf{HList}=\emptyset\subseteq\mathcal{W}\times\mathbb{G}\times\mathbb{Z}_q^*\times\{0,1\}$;
  2) Set the ciphertext space $\mathcal{C}=\mathbb{G}_1\times\mathbb{G}\times\mathbb{G}_1$ and $\mathbf{PK}=(q,\mathbb{G},\mathbb{G}_1,g,\hat{e},P=g^a,\mathcal{W},\mathcal{C})$;
  3) Initialize N hidden structures by repeating the following steps for $i\in[1,N]$:
     a) Pick $u_i \xleftarrow{\$} \mathbb{Z}_q^*$ and $Coin_i \xleftarrow{\sigma} \{0,1\}$;
     b) If $Coin_i=1$, compute $\mathbf{Pub}_i=g^{b\cdot u_i}$;
     c) Otherwise, compute $\mathbf{Pub}_i=g^{u_i}$;
  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 $\mathbf{PK}$ and $\mathbf{PSet}$ to adversary $\mathcal{A}$;
- **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 $x \xleftarrow{\$} \mathbb{Z}_q^*$ and $Coin \xleftarrow{\sigma} \{0,1\}$;
    2) If $Coin=0$, add $(W,g^x,x,Coin)$ into $\mathbf{HList}$ and output g^x;
    3) Otherwise, add $(W,g^{c\cdot x},x,Coin)$ into $\mathbf{HList}$ 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 $\mathbf{HList}$;
    3) If $Coin=0$, output $g^{a\cdot x}$;
    4) Otherwise, abort and output $\perp$;
  - Privacy Query $\mathcal{Q}_{Pri}(\mathbf{Pub})$: Taking as input a structure's public part $\mathbf{Pub}\in\mathbf{PSet}$, algorithm $\mathcal{B}$ does the following steps:
    1) According to $\mathbf{Pub}$, retrieve $(\mathbf{Pub},u,Coin)$ from $\mathbf{SList}$;
    2) If $Coin=0$, output u;
    3) Otherwise, abort and output $\perp$;
  - 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 $\mathbf{Pub}$, retrieve $(W,X,x,Coin)$ and $(\mathbf{Pub},u,Coin')$ respectively from $\mathbf{HList}$ and $\mathbf{SList}$;
    3) Pick $r \xleftarrow{\$} \mathbb{Z}_q^*$;
    4) Seek $(W,\mathbf{Pub},Pt[u,W])$ by W and $\mathbf{Pub}$ in $\mathbf{Pt}$;
    5) If not exists, insert $(W,\mathbf{Pub},Pt[u,W] \xleftarrow{\$} \mathbb{G}_1)$ to $\mathbf{Pt}$ 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
       $$C=(\hat{e}(g^a,X)^u,g^r,\hat{e}(g^a,X)^r\cdot Pt[u,W]);$$

6) Otherwise, pick $R \xleftarrow{\$} \mathbb{G}_1$, set $C = (Pt[u, W], g^r, \hat{e}(g^a, X)^r \cdot R)$, 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 $d \xleftarrow{\$} \{0, 1\}$, 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 $\mathbf{SList}$; 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 $\mathbf{HList}$; And if $Coin = 0$, then abort and output $\perp$;
  4) Seek $(W_d^*, \mathbf{Pub}_d^*, Pt[u_d^*, W_d^*])$ by W_d^* and $\mathbf{Pub}_d^*$ in $\mathbf{Pt}$;
  5) If not exists, insert $(W_d^*, \mathbf{Pub}_d^*, Pt[u_d^*, W_d^*] \xleftarrow{\$} \mathbb{G}_1)$ 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 $R \xleftarrow{\$} \mathbb{G}_1$, 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 with **Query Phase 1**. Note that in **Query Phase 1** and **Query Phase 2**, adversary $\mathcal{A}$ can not 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 $\mathcal{A}$ 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 $Pr[\overline{Abort}]$, $Pr[\mathcal{B} = 1 | Z = \hat{e}(g, g)^{abc}]$ 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 σ and the number of times of adversary $\mathcal{A}$ to query 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

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

When $Z = \hat{e}(g, g)^y$ and the event $\overline{Abort}$ 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:

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

In addition, it is clear that algorithm $\mathcal{B}$ is a PPT algorithm, if adversary $\mathcal{A}$ is a PPT adversary. Conclusively, if a PPT adversary $\mathcal{A}$ wins in the SS-CKSA game of the above SPCHS instance with advantage $Adv_{SPCHS, \mathcal{A}}^{\text{SS-CKSA}}$, 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 $\mathbf{BGen}(1^k)$ 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. ■

APPENDIX C
PROOF OF THEOREM 4

The proof of Theorem 4 is as follows.

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 exiting, since we have $\mathcal{W} \subset \mathcal{ID}_{\text{IBKEM}}$.)
  3) $\mathcal{G}_1$ generates $(\mathbf{PK}_{\text{IBKEM}}, \mathbf{SK}_{\text{IBKEM}})$ by algorithm $\mathbf{Setup}_{\text{IBKEM}}$ and sends $\mathbf{PK}_{\text{IBKEM}}$ to $\mathcal{B}$.
  4) $\mathcal{B}$ queries $\mathcal{G}_1$ for the challenge key-and-encapsulation pair.
  5) $\mathcal{G}_1$ picks $\hat{d} \xleftarrow{\$} \{0,1\}$, generates $(\hat{K}_0^*, \hat{C}_0^*) = \mathbf{Encaps}_{\text{IBKEM}}(\mathbf{PK}_{\text{IBKEM}}, W_0^*, r_0)$ and $(\hat{K}_1^*, \hat{C}_1^*) = \mathbf{Encaps}_{\text{IBKEM}}(\mathbf{PK}_{\text{IBKEM}}, I_1^*, r_1)$, and sends $(\hat{K}_{\hat{d}}^*, \hat{C}_0^*)$ 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 $(\hat{K}_i, \hat{C}_i) = \mathbf{Encaps}_{\text{IBKEM}}(\mathbf{PK}_{\text{IBKEM}}, W_i, u_i)$, 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 $\mathbf{PK}$ and $\mathbf{PSet}$ to $\mathcal{A}$.
- **Query Phase 1**: In this phase, adversary $\mathcal{A}$ adaptively issues the following queries multiple times.
  - Trapdoor Query $\mathcal{Q}_{Trap}(W)$: Taking as input a keyword $W \in \mathcal{W}$, $\mathcal{B}$ forwards the query W both to the decryption key oracles $\hat{S}_W = \mathcal{Q}_{DK}^{IBKEM}(W)$ and $\tilde{S}_W = \mathcal{Q}_{DK}^{IBE}(W)$, and sends $T_W = (\hat{S}_W, \tilde{S}_W)$ to $\mathcal{A}$.
    (In this query, $\mathcal{A}$ can not query the keyword search trapdoor corresponding to the challenge keyword W_0^* or W_1^*. In addition, one may find that $\mathcal{B}$ can not respond the query $\mathcal{Q}_{Trap}(I_1^*)$. However, it 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 $\mathcal{Q}_{Pri}(\mathbf{Pub})$: 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 $\mathbf{SList}$ according to $\mathbf{Pub}$ and outputs u.
  - Encryption Query $\mathcal{Q}_{Enc}(W, \mathbf{Pub})$: Taking as inputs a keyword $W \in \mathcal{W}$ and a structure's public part $\mathbf{Pub}$, $\mathcal{B}$ does the following steps:
    1) If $\mathbf{Pub} = \hat{C}_0^* \bigwedge W \neq W_0^*$, then
       a) Seek $(W, \mathbf{Pub}, Pt[u^*, W])$ by W and $\mathbf{Pub}$ in $\mathbf{Pt}$;
       (Note that u^* is not a really known value. It is just a symbol to denote the random value used to generate $\mathbf{Pub} = \hat{C}_0^*$.)
       b) If not exists, query $\hat{S}_W = \mathcal{Q}_{DK}^{IBKEM}(W)$, insert $(W, \mathbf{Pub}, Pt[u^*, W] \xleftarrow{\$} \mathcal{M}_{\text{IBE}})$ to $\mathbf{Pt}$ and output
       $$C = (\mathbf{Decaps}_{\text{IBKEM}}(\hat{S}_W, \mathbf{Pub}), \mathbf{Enc}_{\text{IBE}}(\mathbf{PK}_{\text{IBE}}, W, Pt[u^*, W]))$$
       (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, picks $R \xleftarrow{\$} \mathcal{M}_{\text{IBE}}$, 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 $\mathbf{Pub} = \hat{C}_0^* \bigwedge W = W_0^*$, then
       a) Seek $(W, \mathbf{Pub}, Pt[u^*, W])$ by W and $\mathbf{Pub}$ in $\mathbf{Pt}$;
       b) If not exists, insert $(W, \mathbf{Pub}, Pt[u^*, W] \xleftarrow{\$} \mathcal{M}_{\text{IBE}})$ to $\mathbf{Pt}$, and output
       $$C = (\hat{K}_{\hat{d}}^*, \mathbf{Enc}_{\text{IBE}}(\mathbf{PK}_{\text{IBE}}, W, Pt[u^*, W]))$$
       (Note that if $\hat{d} = 0$, the output ciphertext C is a correct one, since the full-identity malleability of the IBKEM scheme allows $\mathbf{FIM}(\hat{C}_0^*, W_0^*, u^*) = \hat{K}_{\hat{d}}^*$. Otherwise, the output ciphertext C is an incorrect one. 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, picks $R \xleftarrow{\$} \mathcal{M}_{\text{IBE}}$, 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 $\mathbf{Pub}$, retrieve $(\mathbf{Pub}, u)$ from $\mathbf{SList}$;
   b) Seek $(W, \mathbf{Pub}, Pt[u, W])$ by W and $\mathbf{Pub}$ in $\mathbf{Pt}$;
   c) If not exists, insert $(W, \mathbf{Pub}, Pt[u, W] \xleftarrow{\$} \mathcal{M}_{\text{IBE}})$ to $\mathbf{Pt}$ and output
   $$C = (\mathbf{FIM}(W, u), \mathbf{Enc}_{\text{IBE}}(\mathbf{Pk}_{\text{IBE}}, W, Pt[u, W]))$$
   d) Otherwise, picks $R \xleftarrow{\$} \mathcal{M}_{\text{IBE}}$, set
   $$C = (Pt[u, W], \mathbf{Enc}_{\text{IBE}}(\mathbf{PK}_{\text{IBE}}, W, R)),$$
   update $Pt[u, W] = R$ and output C;

- **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 $M_0^* \xleftarrow{\$} \mathcal{M}_{\text{IBE}}$ and $M_1^* \xleftarrow{\$} \mathcal{M}_{\text{IBE}}$;
  3) $\mathcal{G}_2$ picks $\tilde{d} \xleftarrow{\$} \{0, 1\}$, and sends the challenge IBE ciphertext $\tilde{C}_{\tilde{d}}^* = \mathbf{Enc}_{\text{IBE}}(\mathbf{PK}_{\text{IBE}}, W_0^*, M_0^*)$ to $\mathcal{B}$ if $\tilde{d} = 0$, otherwise sends $\tilde{C}_{\tilde{d}} = \mathbf{Enc}_{\text{IBE}}(\mathbf{PK}_{\text{IBE}}, I_1^*, M_1^*)$ to $\mathcal{B}$.
  4) $\mathcal{B}$ does the following steps:
     a) Seek $(W_0^*, \mathbf{Pub}_0^*, Pt[u^*, W_0^*])$ by W_0^* and $\mathbf{Pub}_0^*$ in $\mathbf{Pt}$;
     b) If not exists, insert $(W_0^*, \mathbf{Pub}_0^*, Pt[u^*, W_0^*] = M_0^*)$ to $\mathbf{Pt}$, output the challenge ciphertext C^* to $\mathcal{A}$ and stop this phase, where
     $$C^* = (\hat{K}_{\hat{d}}^*, \tilde{C}_{\tilde{d}})$$
     (Note that if $\hat{d} = 0$ and $\tilde{d} = 0$, the C^* is a correct one. Otherwise, it is an incorrect one. If $\mathcal{A}$ confirms the incorrectness of C^*, it implies that $\hat{d} = 1$ or $\tilde{d} = 1$ holds. Accordingly, $\mathcal{B}$ 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 with **Query Phase 1**. Note that in **Query Phase 1** and **Query Phase 2**, adversary $\mathcal{A}$ can not 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 $\tilde{d}$, 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 $\mathcal{A}$ can not 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 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 $Win_{IBKEM,\mathcal{B}}^{\text{SS-sID-CPA}}$ 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 $Win_{IBKEM,\mathcal{B}}^{\text{SS-sID-CPA}}$ or $Win_{IBE,\mathcal{B}}^{\text{Anon-SS-ID-CPA}}$ holds under the condition that $\mathcal{B}$ does not abort, we clearly have

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

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 $Pr[Win^{\text{SS-sID-CPA}}_{IBKEM,\mathcal{B}}|\overline{Abort}] + Pr[Win^{\text{Anon-SS-ID-CPA}}_{IBE,\mathcal{B}}|\overline{Abort}]$ as follows.

$$
\begin{aligned}
&Pr[Win^{\text{SS-sID-CPA}}_{IBKEM,\mathcal{B}}|\overline{Abort}] + Pr[Win^{\text{Anon-SS-ID-CPA}}_{IBE,\mathcal{B}}|\overline{Abort}] \\
&= Pr[d' = \hat{d}|\overline{Abort} \bigwedge \overline{Belong}] \cdot Pr[\overline{Belong}] + Pr[d' = \hat{d}|\overline{Abort} \bigwedge Belong] \cdot Pr[Belong] \\
&\quad + Pr[d' = \tilde{d}|\overline{Abort} \bigwedge \overline{Belong}] \cdot Pr[\overline{Belong}] + Pr[d' = \tilde{d}|\overline{Abort} \bigwedge Belong] \cdot Pr[Belong] \\
&= (Pr[d' = \hat{d}|\overline{Abort} \bigwedge \overline{Belong} \bigwedge \hat{d} = 0 \bigwedge \tilde{d} = 0] \cdot Pr[\hat{d} = 0 \bigwedge \tilde{d} = 0] \\
&\quad + Pr[d' = \hat{d}|\overline{Abort} \bigwedge \overline{Belong} \bigwedge \hat{d} = 1 \bigwedge \tilde{d} = 0] \cdot Pr[\hat{d} = 1 \bigwedge \tilde{d} = 0] \\
&\quad + Pr[d' = \hat{d}|\overline{Abort} \bigwedge \overline{Belong} \bigwedge \hat{d} = 0 \bigwedge \tilde{d} = 1] \cdot Pr[\hat{d} = 0 \bigwedge \tilde{d} = 1] \\
&\quad + Pr[d' = \hat{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}] \\
&\quad + (Pr[d' = \hat{d}|\overline{Abort} \bigwedge Belong \bigwedge \hat{d} = 0 \bigwedge \tilde{d} = 0] \cdot Pr[\hat{d} = 0 \bigwedge \tilde{d} = 0] \\
&\quad + Pr[d' = \hat{d}|\overline{Abort} \bigwedge Belong \bigwedge \hat{d} = 1 \bigwedge \tilde{d} = 0] \cdot Pr[\hat{d} = 1 \bigwedge \tilde{d} = 0] \\
&\quad + Pr[d' = \hat{d}|\overline{Abort} \bigwedge Belong \bigwedge \hat{d} = 0 \bigwedge \tilde{d} = 1] \cdot Pr[\hat{d} = 0 \bigwedge \tilde{d} = 1] \\
&\quad + Pr[d' = \hat{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] \\
&\quad + (Pr[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] \\
&\quad + Pr[d' = \tilde{d}|\overline{Abort} \bigwedge \overline{Belong} \bigwedge \hat{d} = 1 \bigwedge \tilde{d} = 0] \cdot Pr[\hat{d} = 1 \bigwedge \tilde{d} = 0] \\
&\quad + Pr[d' = \tilde{d}|\overline{Abort} \bigwedge \overline{Belong} \bigwedge \hat{d} = 0 \bigwedge \tilde{d} = 1] \cdot Pr[\hat{d} = 0 \bigwedge \tilde{d} = 1] \\
&\quad + Pr[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}] \\
&\quad + (Pr[d' = \tilde{d}|\overline{Abort} \bigwedge Belong \bigwedge \hat{d} = 0 \bigwedge \tilde{d} = 0] \cdot Pr[\hat{d} = 0 \bigwedge \tilde{d} = 0] \\
&\quad + Pr[d' = \tilde{d}|\overline{Abort} \bigwedge Belong \bigwedge \hat{d} = 1 \bigwedge \tilde{d} = 0] \cdot Pr[\hat{d} = 1 \bigwedge \tilde{d} = 0] \\
&\quad + Pr[d' = \tilde{d}|\overline{Abort} \bigwedge Belong \bigwedge \hat{d} = 0 \bigwedge \tilde{d} = 1] \cdot Pr[\hat{d} = 0 \bigwedge \tilde{d} = 1] \\
&\quad + Pr[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] \\
&= (2 \cdot Adv^{\text{SS-sK-CKSA}}_{SPCHS,\mathcal{A}} + 3 + Pr[d' = \hat{d}|\overline{Abort} \bigwedge \overline{Belong} \bigwedge \hat{d} = 1 \bigwedge \tilde{d} = 1] \\
&\quad + Pr[d' = \tilde{d}|\overline{Abort} \bigwedge \overline{Belong} \bigwedge \hat{d} = 1 \bigwedge \tilde{d} = 1]) \cdot \frac{1}{4} \cdot Pr[\overline{Belong}] \\
&\quad + (2 \cdot Adv^{\text{SS-sK-CKSA}}_{SPCHS,\mathcal{A}} + 3 + Pr[d' = \hat{d}|\overline{Abort} \bigwedge Belong \bigwedge \hat{d} = 1 \bigwedge \tilde{d} = 1] \\
&\quad + Pr[d' = \tilde{d}|\overline{Abort} \bigwedge Belong \bigwedge \hat{d} = 1 \bigwedge \tilde{d} = 1]) \cdot \frac{1}{4} \cdot Pr[Belong] \\
&= (2 \cdot Adv^{\text{SS-sK-CKSA}}_{SPCHS,\mathcal{A}} + 3 + 2 \cdot Pr[d' = \hat{d} = \tilde{d}|\overline{Abort} \bigwedge \overline{Belong} \bigwedge \hat{d} = 1 \bigwedge \tilde{d} = 1]) \cdot \frac{1}{4} \cdot Pr[\overline{Belong}] \\
&\quad + (2 \cdot Adv^{\text{SS-sK-CKSA}}_{SPCHS,\mathcal{A}} + 3 + 2 \cdot 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] \\
&= (2 \cdot Adv^{\text{SS-sK-CKSA}}_{SPCHS,\mathcal{A}} + 3) \cdot \frac{1}{4} + 2 \cdot Pr[d' = \hat{d} = \tilde{d}|\overline{Abort} \bigwedge \hat{d} = 1 \bigwedge \tilde{d} = 1] \cdot \frac{1}{4} \\
&= \frac{1}{2} \cdot Adv^{\text{SS-sK-CKSA}}_{SPCHS,\mathcal{A}} + 1
\end{aligned}
$$

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

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

According to the above computations, we have

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

In addition, it is clear that adversary $\mathcal{B}$ is a PPT adversary, if $\mathcal{A}$ is a PPT adversary. Conclusively, we have that if a PPT adversary $\mathcal{A}$ wins in the SS-sK-CKSA game of the generic SPCHS construction with advantage $Adv^{\text{SS-sK-CKSA}}_{SPCHS,\mathcal{A}}$, 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 $\frac{1}{4N} \cdot Adv^{\text{SS-sK-CKSA}}_{SPCHS,\mathcal{A}}$. ■

APPENDIX D
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 [3]. Then we prove its collision-free full-identity malleability and the Anon-SS-ID-CPA security in the RO model. Let identity space $\mathcal{ID}_{\text{IBKEM}} = \{0,1\}^*$. This IBKEM instance is as follows.

- $\mathbf{Setup}_{\text{IBKEM}}(1^k, \mathcal{ID}_{\text{IBKEM}})$: Take as input a security parameter 1^k and the identity space $\mathcal{ID}_{\text{IBKEM}}$, compute $(q, \mathbb{G}, \mathbb{G}_1, g, \hat{e}) \xleftarrow{\$} \mathbf{BGen}(1^k)$, pick $s \xleftarrow{\$} \mathbb{Z}_q^*$, set $P \leftarrow g^s$, choose a cryptographic hash function $H : \{0,1\}^* \rightarrow \mathbb{G}$, set the encapsulated key space $\mathcal{K}_{\text{IBKEM}} = \mathbb{G}_1$, set the encapsulation space $\mathcal{C}_{\text{IBKEM}} = \mathbb{G}$, and output the master public key $\mathbf{PK}_{\text{IBKEM}} = (q, \mathbb{G}, \mathbb{G}_1, g, \hat{e}, P, H, \mathcal{ID}_{\text{IBKEM}}, \mathcal{K}_{\text{IBKEM}}, \mathcal{C}_{\text{IBKEM}})$ and the master secret key $\mathbf{SK}_{\text{IBKEM}} = s$.
- $\mathbf{Extract}_{\text{IBKEM}}(\mathbf{SK}_{\text{IBKEM}}, ID)$: Take as inputs $\mathbf{SK}_{\text{IBKEM}}$ and an identity $ID \in \mathcal{ID}_{\text{IBKEM}}$, and output a decryption key $\hat{S}_{ID} = H(ID)^s$ of ID.
- $\mathbf{Encaps}_{\text{IBKEM}}(\mathbf{PK}_{\text{IBKEM}}, ID, r)$: Take as inputs $\mathbf{PK}_{\text{IBKEM}}$, an identity $ID \in \mathcal{ID}_{\text{IBKEM}}$ 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 $\hat{C} = g^r$.
- $\mathbf{Decaps}_{\text{IBKEM}}(\hat{S}_{ID'}, \hat{C})$: 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 $\mathbf{FIM}$ is efficient. Moreover, it is easy to find that the function $\mathbf{FIM}$ 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 $\mathbf{FIM}$ 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 $\mathbf{FIM}(ID, r) \neq \mathbf{FIM}(ID, r')$ due to the randomness of the values r and r'. Therefore, the function $\mathbf{FIM}$ has the 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 $\mathcal{Q}_H(\cdot)$. Suppose a PPT adversary $\mathcal{A}$ wins in the Anon-SS-ID-CPA game of the above IBKEM instance with advantage $Adv^{\text{Anon-SS-ID-CPA}}_{IBKEM,\mathcal{A}}$, in which $\mathcal{A}$ makes at most q_p queries to oracle $\mathcal{Q}^{IBKEM}_{DK}(\cdot)$. Then there is a PPT algorithm $\mathcal{B}$ that solves the DBDH problem in $\mathbf{BGen}(1^k)$ with advantage approximately*

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

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 $\mathbf{BGen}(1^k)$ with advantage approximately $\frac{4}{(e \cdot q_p)^2} \cdot Adv^{\text{Anon-SS-ID-CPA}}_{IBKEM,\mathcal{A}}$. Let $Coin \xleftarrow{\sigma} \{0,1\}$ denote the operation that picks $Coin \in \{0,1\}$ according to the probability $Pr[Coin = 1] = \sigma$ (the specified value of σ will be decided latter). The constructed algorithm $\mathcal{B}$ in the Anon-SS-ID-CPA game are 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 $\mathcal{ID}_{\text{IBKEM}}$, and does the following steps:
  1) Initializes 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 $\mathcal{K}_{\text{IBKEM}} = \mathbb{G}_1$, the encapsulation space $\mathcal{C}_{\text{IBKEM}} = \mathbb{G}$ 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 $\mathcal{A}$ 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:
    1) Pick $x \xleftarrow{\$} \mathbb{Z}^*_q$ and $Coin \xleftarrow{\sigma} \{0,1\}$;
    2) If $Coin = 0$, add $(ID, g^x, x, Coin)$ into $\mathbf{HList}$ and output g^x;
    3) Otherwise, add $(ID, g^{c \cdot x}, x, Coin)$ into $\mathbf{HList}$ and output $g^{c \cdot x}$;
  - Decryption Key Query $\mathcal{Q}^{IBKEM}_{DK}(ID)$: Taking as input an identity $ID \in \mathcal{ID}_{\text{IBKEM}}$, 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 $\mathbf{HList}$;
    3) If $Coin = 0$, output $g^{a \cdot x}$;
    4) Otherwise, abort and output $\perp$;
- **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 algorihm $\mathcal{B}$; $\mathcal{B}$ picks $\hat{d} \xleftarrow{\$} \{0,1\}$, 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 $\mathbf{HList}$;
  4) If $Coin^*_0 = 0 \bigvee Coin^*_1 = 0$, then abort and output $\perp$;
  5) Finally send the challenge key-and-encapsulation pair $(Z^{x^*_{\hat{d}}}, g^b)$ to adversary $\mathcal{A}$;
- **Query Phase 2**: This phase is the same with **Query Phase 2**. Note that in **Query Phase 1** and **Query Phase 2**, adversary $\mathcal{A}$ can not 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 $\mathcal{B}$. If $\hat{d} = \hat{d}'$, $\mathcal{B}$ 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 $Pr[\overline{Abort}]$, $Pr[\mathcal{B} = 1 | Z = \hat{e}(g,g)^{abc}]$ and $Pr[\mathcal{B} = 1 | Z = \hat{e}(g,g)^y]$, and the advantage $Adv^{DBDH}_{\mathcal{B}}(1^k)$.

According to the above game, the probability of the event $\overline{Abort}$ only relies on the probability σ and the number of times of adversary $\mathcal{A}$ to query oracle $\mathcal{Q}^{IBKEM}_{DK}(ID)$. 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 SS-ASC-CKA game in adversary $\mathcal{A}$'s mind. So we have $Pr[\hat{d} = \hat{d}' | \overline{Abort} \bigwedge Z = \hat{e}(g,g)^{abc}] = (Adv^{\text{Anon-SS-ID-CPA}}_{IBKEM,\mathcal{A}} + \frac{1}{2})$.

When $Z = \hat{e}(g,g)^y$ and the event $\overline{Abort}$ holds, algorithm $\mathcal{B}$ generates the incorrect challenge ciphertext, and it is independent of the challenge identities ID^*_0 and ID^*_1. So we have $Pr[\hat{d} = \hat{d}' | \overline{Abort} \bigwedge Z = \hat{e}(g,g)^y] = \frac{1}{2}$.

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

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

In addition, it is clear that algorithm $\mathcal{B}$ is a PPT algorithm, if adversary $\mathcal{A}$ is a PPT adversary. Conclusively, 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 $\mathbf{BGen}(1^k)$ with advantage approximately

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

where e is the base of natural logarithms. ■

APPENDIX E
PROOF OF THEOREM 5

The proof of Theorem 5 is as follows.

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 are as follows.

- **Setup Phase**: Algorithm $\mathcal{B}$ gets as input an $(\ell+1)$-group system $\mathbf{MPG}_{\ell+1}$ and group elements $g, g^{x_1}, \cdots, g^{x_{\ell+2}} \in \mathbb{G}_1$ and $S \in \mathbb{G}_{\ell+1}$, where either $S = \hat{e}(g^{x_1}, \cdots, g^{x_{\ell+1}})^{x_{\ell+2}}$ (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 $\mathbf{H}$ 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 $(hk, td) \leftarrow \mathbf{TGen}(1^k, g^{x_1}, \cdots, g^{x_\ell}, g)$, 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 $\mathcal{A}$ adaptively issues the following query multiple times.
  - Decryption Key Query $\mathcal{Q}_{DK}^{IBKEM}(ID)$: Taking as input an identity $ID \in \mathcal{ID}_{\text{IBKEM}}$, algorithm $\mathcal{B}$ does the following steps:
    1) Compute $\mathbf{TEval}(td, ID) = (a_{ID}, B_{ID})$;
    2) If $a_{ID} = 0$, return $\hat{S}_{ID} = \hat{e}(B_{ID}, h')$;
    3) 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 algorihm $\mathcal{B}$; $\mathcal{B}$ picks $\hat{d} \xleftarrow{\$} \{0,1\}$, 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 $\mathcal{A}$;

  Note that 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 with **Query Phase 2**. Note that in **Query Phase 1** and **Query Phase 2**, adversary $\mathcal{A}$ can not 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 $\mathcal{I} = \{ID_1, \cdots, ID_{q_p}, ID_0^*, ID_1^*\}$ 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 latter. As in [31], [49], $\mathcal{B}$ "artificially" aborts with probability $1 - 1/(P_{\mathcal{I}} \cdot p(k))$ for the polynomial $p(k)$ from Definition 12 and outputs $\perp$. If not abort, $\mathcal{B}$ uses the guess of $\mathcal{A}$. It means that if $\hat{d} = \hat{d}'$, $\mathcal{B}$ outputs 1, otherwise 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 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 [49] for details on this evaluation. ■