Abstract—With the rapid growth of data, it is desirable to outsource data on remote storage server. The emergency of cloud computing makes the dream true and more and more sensitive data are being centralized into cloud for sharing. Since the public cloud server cannot be fully trusted in protecting them, encryption is a promising way to keep confidentiality but leads to high communication and computation overhead for some useful data operations. Searchable encryption initiated by Song et al. provides an efficient solution to support for keyword-based search directly on encrypted data. Nevertheless, existing work depends on key sharing among authorized users, which inevitably causes the risks of key exposure and abuse.

In this paper, the keyword search over encrypted data with differential privileges is addressed. We provide a novel framework for secure outsourcing and sharing of encrypted data on hybrid cloud. The framework is full-featured: i) it enables authorized users to perform keyword-based search directly on encrypted data without sharing the same private key; ii) it provides two-layered access control to achieve fine-grained sharing of encrypted data. The security analysis shows that the proposed generic construction satisfies the requirements of message privacy and keyword privacy.

I. INTRODUCTION

A. Background

With the rapid growth of data within organizations, it is desirable to make data storage and management outsource and enjoy the best-of-breed applications from a shared pool of configurable resources. The new coming term commonly referred to as cloud computing has made the dream come true. It delivers computing as a service rather than a product, whereby shared resources, software, and information are provided to computers and other devices as a utility over a network. Recently, more and more sensitive data are being centralized into cloud, such as emails, personal health records, private videos and photos, company finance data, government documents, etc. Nevertheless, since storage server cannot be fully trusted, one of the critical issues is to protect confidentiality for the outsourced data.

A natural solution is to encrypt data by client before outsourcing. However, encryption may destroy some implicit attributes of original data, and render the encrypted data difficult to utilize. This fact would cause most of data operations impossible. For example, if a user wants to retrieve documents containing certain keyword, he has to download all the data and decrypt them locally, causing huge communication and computation overhead.

Searchable encryption, initiated by Song et al. [1], allows to perform keyword-based search directly on encrypted data without decryption. The technology roughly divides the cryptographic components into the client and the server: client is able to encrypt/decrypt data and generate trapdoor while server processes keyword-based search.

Though the technology is secure and effective, the existing searchable encryption is not suitable for data sharing in cloud computing: it allows authorized users to generate trapdoor for certain keyword and perform decryption on encrypted data through a sharing private key. This implies that the operations are allowed to be executed by a group of users with the private key. As a result, it not only increases the risk of key exposure and key abuse, but also makes it hard to revoke user’s search and decryption ability. For example, a single user’s revocation indicates key changing globally, resulting in decrypting all the data with old keys and re-encrypting it using the new key. For large data sets, this solution is obviously impractical.

Additionally, existing schemes have not efficiently solved the search in multi-user system with differential searching privileges. Nevertheless, the fact is that differential searching privileges are necessary in applications. For example, a manager and some employees are all in one company, but only the manager is able to search for and access to all of the documents while employees are restricted to access to part of them. Therefore, an access control mechanism enabling above differential search is urgent for data sharing in cloud.

B. Contribution

In this paper, we consider a novel keyword search architecture in hybrid cloud, that is, a trusted private cloud and a public cloud are assumed in our system. The utilization of the hybrid cloud computing system not only enables the users to search efficiently, but also reaps the benefits of having access to services and applications from cloud service providers. This allows us to expand our web presence and still maintain some level of autonomy and privacy.

Under the hybrid architecture, we provide a novel framework for outsourcing and sharing searchable encrypted data.
The framework does not rely on shared keys and has the following features.

- **Two-layered access control.** Our framework provides two-layered access control to achieve fine-grained control on shared encrypted data. On the first layer, an authorization and revocation mechanism is provided: each authorized user is able to get his individual private key to perform search and decryption. After a user’s key has been revoked, the user will no longer be able to read and search the shared data. On the second layer, an identity-based access control mechanism is provided: any user is allowed to submit a set of identities along with the uploading encrypted data to public cloud storage to restrict that only the user in the set is able to successfully search for and perform decryption on the shared encrypted data.

- **Keyword-based search.** Authorized user is able to use his individual private key to generate query for certain keyword. Storage server performs “test” on each encrypted data which allows the user to search and return the matched data.

## II. RELATED WORK

The Symmetric searchable encryption (SSE) is proposed by Song et al. [1], in which a user stores his encrypted data in a semi-trusted server and later searches with a certain keyword. In [1], each word is independently encrypted under a specified two-layered encryption. Later, Goh [2] introduced bloom filter to construct secure indexes for the keyword search, which allows server to check if a document contains a keyword without decrypting the entire document. A formal treatment to SSE was presented by Curtmola et al. [3]. They provided the security notions for SSE and presented “index” approach, in which an array and a look-up table are built for the entire document collection. Each entry of the array is used to store an encryption of document identifier set associated with a certain keyword, while the look-up table enables one to locate and decrypt the appropriate element from array. Recently, aiming at providing SSE with efficient search and update, Liesdonk et al. [4] presented two schemes: the first one transforms each unique keyword to a searchable representation such that user can keep track of the set of associated keywords via appropriate trapdoor. The second one deploys a hash chain by applying repeatedly a hash function to an initial seed. Since only the user knows the seed, he is able to traverse the chain forward and backward, while the server is able to traverse the chain forward only.

Searchable encryption in public key setting, originating from store-and-forward system, such as email system, in which a receiver can search data encrypted under the receiver’s public key on an outsourced storage system, is initiated by Boneh et al. [5]. They firstly suggested a public-key encryption with keyword search (PEKS) construction with bilinear Diffie-Hellman assumption in random oracle. Accordingly, Khader [6] constructed a PEKS scheme which is secure under a chosen keyword attack without random oracle, and Crescenzo [7] drew a construction without bilinear map. Recently, several researchers focused on make improvement and extension on Boneh’s construction [5], such as conjunctive keyword search [8][9], fuzzy keyword search [10], etc.

We note that almost all the existing searchable encryption needs to share a private key leading to the risk of key exposure in sharing environment. Though [8] provided an extension to multi-user setting, their scheme needs to generate private key for users in setup phase and cannot support for user’s dynamic authorization or revocation.

Another two related work similar to us are [11] and [12]. In [11], a scheme is provided, which enables efficient multi-dimensional keywords search simultaneously with range query and delegation and revocation of searching capabilities. In [12], the authors proposed an encryption scheme where each authorized user in the system has his own keys to encrypt/decrypt data and perform keyword-based search. Comparing with our work, the two lack of the support for access control on encrypted data. More precisely, their constructions only support for authorization on user according to some authentication information while ours allows two-layered access control mechanism. On the first layer it is similar to the two, but on the second layer the access control takes effect on encrypted data to restrict some users to access it even if they are authorized.

## III. SYSTEM MODEL AND SECURITY DEFINITION

### A. System Model

We present our system model for secure outsourcing and sharing searchable encrypted data on public cloud storage in Fig. 1.

In the proposed model, a group of users make use of the storage service offered by a cloud service provider to outsource their data in a secure way for sharing. More precisely, the data owner in the group uploads a collection of encrypted documents, a set of related keywords and a set of included users to the public cloud storage. It only allows the authorized users in the set to perform keyword-based search on the encrypted documents.

To achieve this goal (i.e. supporting fine-grained access control and keyword-based search simultaneously), a hybrid cloud is used in proposed model. It aims to provide a user fully trusted interface for secure storage. The interface allows to securely submit querying keyword and extract individual private key for decryption. We require that the low bandwidth connection between user and private cloud is protected by a secure channel (e.g. via SSL/TLS).

Actually, hybrid cloud has attracted more and more attention recently. For example, an enterprise might use a public cloud service, such as Amazon S3 for achieved data but continue to maintain in-house storage for operational customer data. Alternatively, the trusted private cloud could be a cluster of virtualized cryptographic co-processors, which are offered as a service by a third party and which provide the necessary hardware-based security features to implement a remote execution environment trusted by the users.
Let $\mathcal{U}$ be the universe of users, $\Sigma$ be the dictionary and $\mathcal{M}$ be the message space. We present the algorithm definitions as follows.

- **Setup($\lambda, n$)** Takes as input the security parameter $\lambda$ and the number of users in the system $n$, it outputs the master key $MK$ and the public key $PK$.
- **Extract($PK, MK, ID_i$)** Takes as input the public key $PK$, the master key $MK$ and the user identity $ID_i$, it outputs the user private key $SK_i$.
- **Encrypt($PK, S, M, W$)** Takes as input the public key $PK$, the set of authorized users $S \subseteq \mathcal{U}$, the message $M \in \mathcal{M}$ and the set of related keywords $W \subseteq \Sigma$, it outputs the ciphertext $CT = (S, hdra, C_M, C_W)$. We refer to $(S, hdra)$ as the full header for ciphertext, and $C_M$ and $C_W$ which is message ciphertext and keyword ciphertext respectively as ciphertext body. We note that the KEM-DEM philosophy [13] is followed in $C_M$ production. That is when a message $M \in \mathcal{M}$ is to be encrypted, a KEM (key encapsulation mechanism) is used to generate a symmetric key while DEM (data encapsulation mechanism) encrypts $M$ with such a session key.
- **Query($SK_i, ID_i, W$)** Takes as input the user’s private key $SK_i$, the user’s identity $ID_i$ and the keyword $W$ for queried, it outputs user’s query $Q_i$.
- **Trapdoor($PK, MK, Q_i$)** Takes as input the public key $PK$, the master key $MK$ and the user query $Q_i$, it outputs the trapdoor $TP$.
- **Test($PK, CT, TP$)** Takes as input the public key $PK$, the ciphertext $CT$ and the trapdoor $TP$, it outputs “yes” or “no”.
- **Decrypt($PK, SK_i, S, hdra, C_M$)** Takes as input the public key $PK$, the user’s private key $SK_i$ and the returned ciphertext $(S, hdra, C_M)$, it outputs the original message $M$ or 1. We note that in order to save bandwidth, only the full header and message ciphertext are transmitted back to users.

**B. Security Definition**

We assume the private cloud is fully trusted and consider the public cloud storage semi-trusted, more precisely, it will follow our proposed protocol but try to find out as much private information as possible based on its possession.

Thus, two types of adversaries are considered: i) A curious server and outside interception adversary, which can potentially access all the trapdoors and ciphertexts. This coincides with the assumption that the public cloud storage is honest-but-curious and the fact that the two communication links (i.e. one is between trusted private cloud and public cloud storage, and the other is between user and public cloud storage) are insecure in confidentiality. ii) A curious user, who can obtain his individual private key and share his authentication with other users.

Since the ciphertext shared in the system is encoded by message and keyword, we have two security demands: The first is message privacy, which requires that adversary could not learn anything from the message ciphertext. We define the message privacy game for our setting in the following.

**Message privacy**

**Init:** The adversary $A_M$ first outputs an authorized user set $S^* \subseteq \mathcal{U}$ and a target keyword set $W^* \subseteq \Sigma$.

**Setup:** The challenger runs Setup($\lambda, n$) to obtain public key $PK$ and master key $MK$. He gives $A_M$ the public key $PK$.

**Phase 1:** $A_M$ adaptively issues queries in one of the two types: i) Given identity $ID_i \in \mathcal{U} - S^*$, private keyword oracle will respond by returning $SK_i = \text{Extract}(PK, MK, ID_i)$; ii) Given any pair $(S, M)$ where $S \cap S^* = \phi$, encryption oracle will respond by returning $CT = \text{Encrypt}(PK, S, M, W^*)$.

**Challenge:** $A_M$ outputs two messages $M_0$ and $M_1$ with the restriction that $M_0$ and $M_1$ have the equal length, and $M_0$ and $M_1$ have not been queried in phase 1. The challenger responds with $(S^*, hdra, C^*_M, C^*_W) = \text{Encrypt}(PK, S^*, M_b, W^*)$ where $b$ is selected randomly from $\{0, 1\}$.

**Phase 2:** $A_M$ continues to issue queries like in phase 1, but with the restriction that $M \neq M_0$ and $M \neq M_1$ in encryption oracle.

**Guess:** $A_M$ outputs a guess $b' \in \{0, 1\}$.

The advantage of an adversary $A_M$ in this game is defined as $\text{Pr}[b' = b] - \frac{1}{2}$.

**Definition 1 (Message privacy).** A scheme is secure for message privacy if all polynomial-time adversaries have at most a negligible advantage in the message privacy game.
The second security notion needed to be considered is keyword privacy. It requires that a keyword ciphertext payload of the ciphertext does not reveal any information about the keyword encrypted. This is covered by the “semantic security of PEKS against chosen keyword attack” [5]. We redefine the security game to accommodate to our setting.

**Keyword privacy**

**Init:** The adversary $A_K$ first outputs an authorized user set $S^* \subseteq U$ and a target message $M^* \in M$.

**Setup:** The challenger runs $\text{Setup}(\lambda, n)$ to obtain public key $PK$ and master key $MK$. He gives $A_K$ the public key $PK$.

**Phase 1:** $A_K$ adaptively issues queries in one of the two queries: i) Given identity $ID_i \in U - S^*$, private key oracle will respond by returning $SK_i = \text{Extract}(PK, MK, ID_i)$; ii) Given any pair $(W, ID_i)$ but $ID_i \notin S^*$, trapdoor oracle will respond by returning $\text{Trapdoor}(PK, MK, Q_i)$, where $Q_i = \text{Query}(\text{Extract}(PK, MK, ID_i), ID_i, W)$.

**Challenge:** $A_K$ outputs two keywords $W_0$ and $W_1$ which are not queried in phase 1. The challenger responds with $(S^*, \text{hdr}^*, C_M^*, C_W^*) = \text{Encrypt}(PK, S^*, M^*, \{W_b\})$ where $b$ is selected randomly from $\{0, 1\}$.

**Phase 2:** $A_K$ continues to issue queries like in phase 1 with the additional restriction that $W_0$ and $W_1$ should not be queried in the trapdoor oracle.

**Guess:** $A_K$ outputs a guess $b' \in \{0, 1\}$.

The advantage of an adversary $A_K$ in this game is defined as $\Pr[b' = b] - \frac{1}{2}$.

**Definition 2** (Keyword privacy). A scheme is secure for keyword privacy if all polynomial-time adversaries have at most a negligible advantage in the keyword privacy game.

IV. PROPOSED FRAMEWORK

A. Identity-based Broadcast Encryption

Before providing the proposed framework, we will briefly review the definition of identity-based broadcast encryption [14] first.

Identity-based broadcast encryption scheme $\text{IBBE}$ with security parameter $\lambda$ and maximal size $n$ of the target set, is a tuple of algorithms described as follows.

- **Setup($\lambda, n$):** Takes as input the security parameter $\lambda$ and the maximal size of the set of receivers $n$ for one encryption, and outputs a master key $MK$ and public key $PK$.
- **Extract($MK, ID_i$):** Takes as input the master key $MK$ and a user identity $ID_i$ and outputs a user private key $SK_i$.
- **Encrypt($S, PK$):** Takes as input the public key $PK$ and a set of included identities $S = \{ID_{1}, ID_{2}, \ldots, ID_{s}\}$ with $s \leq n$, and outputs a pair $(\text{hdr}, K)$ where $\text{hdr}$ is the header and $K$ is a symmetric session key.
- **Decrypt($S, ID_i, SK_i, \text{hdr}, PK$):** Takes as input a subset $S = \{ID_{1}, \ldots, ID_{s}\}$ with $s \leq n$, and identity $ID_i$ and the corresponding private key $SK_i$, a header $\text{hdr}$, and the public key $PK$. If $ID \in S$, the algorithm outputs the message encryption $K$ which is then used to decrypt the broadcast body and recover original message.

We note that $\text{IBBE}$ in the definition works as a KEM, more precisely, $\text{IBBE.Encrypt}()$ will produce a random symmetric key $K$ which is used by a DEM (the role of DEM is played by $SE$ which is a symmetric key encryption scheme) to encrypt short or long messages. Such an identity-based broadcast encryption construction can be referred to in [14].

B. Generic Construction

Let $\text{IBBE}$ denote an identity-based broadcast encryption scheme, $SE$ denote a symmetric encryption scheme, $\text{SIG}$ denote a digital signature scheme and $\text{PEKS}$ denote a public key encryption with keyword search scheme.

We now use these primitives to provide a generic construction for sharing searchable encrypted data on hybrid cloud.

- **Setup($\lambda, n$):** Establish and initialize the authorized user table $T_{\text{user}}$. Run $\text{IBBE.Setup}($\lambda, n$) to obtain $(PK_{\text{IBBE}}, MK_{\text{IBBE}})$. Run $\text{PEKS.Set}($\lambda$)$ to obtain $(PK_{\text{PEKS}}, SK_{\text{PEKS}})$. Finally output the public key $PK = (PK_{\text{IBBE}}, PK_{\text{PEKS}})$ and the master key $MK = (MK_{\text{IBBE}}, SK_{\text{PEKS}}, T_{\text{user}})$.
- **Extract($MK, ID_i$):** Firstly, parse the master key $MK$ as $(MK_{\text{IBBE}}, SK_{\text{PEKS}}, T_{\text{user}})$. Run $\text{SIG.Setup}$ to obtain a pair $(VK_{\text{SIG}}^i, SK_{\text{SIG}}^i)$ and compute $SK_{\text{IBBE}}^i = \text{IBBE.Extract}(MK_{\text{IBBE}}, ID_i)$. Finally add $(ID_i, VK_{\text{SIG}}^i)$ into $T_{\text{user}}$ and output user $ID_i$’s private key $SK_i = (SK_{\text{IBBE}}^i, SK_{\text{SIG}}^i)$.
- **Encrypt($PK, S, M, W$):** Assume for notational simplicity that $W = \{W_1, W_2, \ldots, W_k\}$. The algorithm firstly parses $PK$ as $(PK_{\text{IBBE}}, PK_{\text{PEKS}})$. Then, run $\text{IBBE.Encrypt}(S, PK_{\text{IBBE}})$ to produce the session key $K$ and its header $\text{hdr}$, and compute the encrypted message $C_M \leftarrow SE.\text{Encrypt}(K, M)$. After that, continue to perform searchable encryption on related keywords. For each $i = 1, 2, \ldots, k$, run $\text{PEKS.Encrypt}(PK_{\text{PEKS}}, W_i)$ to obtain $C_{W_i}$. Finally output ciphertext $CT = (S, \text{hdr}, C_M, C_W = \{C_{W_i}\}_{i=1}^k)$.
- **Query($SK_i, ID_i, W$):** Firstly, parse the user private key $SK_i$ as $(SK_{\text{IBBE}}^i, SK_{\text{SIG}}^i)$, and compute the signature $\sigma_{ID_i} = \text{SIG.Sign}(SK_{\text{SIG}}^i, ID_i) || W)$. Finally output user $ID_i$’s query $Q_{ID_i} = (ID_i, W, \sigma_{ID_i})$.
- **Trapdoor($PK, MK, Q_{ID_i}$):** Parse $PK$, $MK$ and $Q_{ID_i}$ as $(PK_{\text{IBBE}}, PK_{\text{PEKS}}), (MK_{\text{IBBE}}, SK_{\text{PEKS}}, T_{\text{user}})$ and $(ID_i, W, \sigma_{ID_i})$ respectively. Find the pair $(ID_i, VK_{\text{SIG}}^i)$ in $T_{\text{user}}$ and check whether the signature is valid by computing and testing $\text{SIG.Verify}(VK_{\text{SIG}}^i, ID_i) || W, \sigma_{ID_i}$ =
Encrypt runs is to be outsourced and shared with users in Fig. 2, when a message submission does not need trusted private cloud involving. As identities along with the encrypted message directly to the MK translation. In the initialization procedure (as shown in subfigure (b) in Fig. 2), it is responsible for user management and query authorization. It is used to encrypt original message with such a session key. If \( \text{SE} \) is IND-CCA2 secure, the storage without the symmetric key cannot get any information about original message. At the same time, the symmetric key is protected by \( \text{IBBE} \), thus the storage is not able to recover the key if \( \text{IBBE} \) is secure. Cramer et al [13] have shown that if the KEM and DEM systems are secure in the sense of IND-CCA2, then the hybrid encryption system is secure in the sense of IND-CCA2 as well. Therefore, message privacy is achieved by proposed framework if \( \text{IBBE} \) and \( \text{SE} \) is IND-CCA2 secure.

\section*{C. Working Procedures}

To support outsourcing and sharing searchable encrypted data, the proposed framework can be split into five procedures: initialization, submission, query, authorization, and revocation. In the following, we will describe the details of the five procedures respectively.

\section*{Initialization.}

Typically in the proposed framework, the trusted private cloud works as the control and query center. It is responsible for user management and query translation. In the initialization procedure (as shown in subfigure (a) in Fig. 2), the trusted private cloud runs Setup\((\lambda, n)\) to obtain \( PK = (PK_{\text{IBBE}}, PK_{\text{PEKS}}) \) and \( MK = (MK_{\text{IBBE}}, SK_{\text{PEKS}}, T_{\text{user}}) \). Finally, it publishes \( (PK_{\text{PEKS}}, PK_{\text{IBBE}}) \) to users and \( PK_{\text{PEKS}} \) to storage, and maintains \( MK \) at local.

\section*{Submission.}

Any user is able to submit a specified set \( S \) of identities along with the encrypted message directly to the storage and only authorized users in \( S \) is able to search for and decrypt the encrypted message with his private key. The whole submission does not need trusted private cloud’s involving. As shown in subfigure (b) in Fig. 2, when a message \( M \in M \) is to be outsourced and shared with users in \( S \), the owner runs Encrypt\((PK, S, M, W)\) and finally submits the whole ciphertext \( CT = (S, hdr, C_M, \{C_W\}^k_{i=1}) \) to public storage.

\section*{Authorization.}

A new coming user needs to apply to trusted private cloud for his individual private key which is used for keyword-based search and performing decryption on the returned encrypted message. The authorization is similar to private key extraction in identity-based broadcast encryption. The only difference is that we use the digital signature \( SIG \) to avoid unauthorized user forging query. In the authorization procedure, the new coming user submits his identity \( ID_i \) to the trusted private cloud. Trusted private cloud runs Extract\((MK, ID_i)\) to obtain the user’s private key \( (SK_{\text{IBBE}}, SK_{\text{SIG}}) \) and sends it back through secure channel.

\section*{Query.}

Query is a bit complex in the proposed framework. Assume an authorized user \( ID_i \) wants to search the encrypted message correlated with a certain keyword \( W \). i) Firstly, he should run Query\((SK_i, ID_i, W)\) to produce the query \( (ID_i, W, \sigma_{ID_i}) \) and submit it to trusted private cloud through secure channel. ii) Secondly, after receiving the query, trusted private cloud executes Trapdoor\((PK, MK, Q_{ID_i})\) to generate the trapdoor \( TP \) and transmits it to storage. iii) Thirdly, when receiving the trapdoor, the storage uses test\((PK, CT, TP)\) to perform a test on each ciphertext \( CT \) and returns the matched ones to user. iv) Finally, user \( ID_i \) performs a two-phased decryption on returned data with \( SK_i \) and recovers original message.

\section*{Revocation.}

Since each user has his individual private key, revocation is simple in the proposed framework. If trusted private cloud determines to revoke user \( ID_i \)’s keyword-based searching ability, it only needs to delete the pair \( (ID_i, V K_{i_{eq}}) \) in \( T_{\text{user}} \). Afterwards, for each query from user \( ID_i \), the trusted private cloud cannot find the pair \( (ID_i, VK_{SIG}) \) in \( T_{\text{user}} \), accordingly, such a query is not allowed.

\section*{D. Security Analysis}

We will make security analysis on message privacy and keyword privacy for the proposed framework below.

\section*{Message privacy.}

In the proposed framework, message is encrypted using KEM-DEM philosophy, more precisely, \( \text{IBBE} \) works as a KEM to produce a random symmetric key and \( \text{SE} \) is used to encrypt original message with such a session key. If \( \text{SE} \) is IND-CCA2 secure, the storage without the symmetric key cannot get any information about original message. If \( \text{IBBE} \) is IND-CCA2 secure, thus the storage is not able to recover the key if \( \text{IBBE} \) is secure. Cramer et al [13] have shown that if the KEM and DEM systems are secure in the sense of IND-CCA2, then the hybrid encryption system is secure in the sense of IND-CCA2 as well. Therefore, message privacy is achieved by proposed framework if \( \text{IBBE} \) and \( \text{SE} \) is IND-CCA2 secure.

\section*{Keyword privacy.}

PEKS is used in the proposed framework to keep security for both the keyword and trapdoor. The mechanism not only uses encryption algorithm to protect original keywords, but also generates trapdoor to make instead of the queried keyword being submitted to storage.

Suppose the proposed framework fails to achieve keyword privacy, which means that there exists an algorithm \( A \) which can distinguish an encryption of \( W_0 \) from that of \( W_1 \). Then, we will build an algorithm \( A' \) that utilizes \( A \) to attack PEKS on semantic security [5]. \( A' \) has an access to an oracle \( O_{\text{Trapdoor}}(\cdot) \) that responds with the trapdoor of the queried keyword. Upon receiving any request \( ID_i \) and \( (W, ID_i) \) from \( A, A' \) answers them with Extract\((MK_{\text{IBBE}}, ID_i)\) and \( (ID_i, O_{\text{Trapdoor}}(W)) \) respectively. After making queries, \( A \) outputs two keywords \( W_0 \) and \( W_1 \). \( A' \) directly sends these
keywords to challenger and receives $C_W^b$ which is the encryption of $W_b$ where $b$ is a random bit chosen by challenger. $A'$ continues to construct the whole ciphertext by computing $(\text{hdr}, K) = \text{IBBE.Encrypt}(PK_{\text{IBBE}}, S)$ and $C_M^b = \text{SE.Encrypt}(K, M)$ and returns $(S^b, \text{hdr}, C_{M^b}, C_W^b)$ back to $A$, which answers with $b' \in \{0, 1\}$. $A'$ finally outputs $b'$ as a guess for $b$. Obviously, the view of $A$ is identical to that in keyword privacy security game shown in section III-B. Suppose $A$ answers guesses $b$ correctly with non-negligible probability, $A'$ will have a non-negligible probability in distinguishing $\mathcal{PEKS.Encrypt}(PK_{\text{PEKS}}, W_0)$ from $\mathcal{PEKS.Encrypt}(PK_{\text{PEKS}}, W_1)$ as well. Therefore, keyword privacy is achieved by proposed framework if $\mathcal{PEKS}$ is semantically secure.

V. CONCLUSION

In the paper, we present a novel framework for data outsourcing and sharing on the hybrid cloud computing. It consists of a trusted private cloud and a public cloud storage. In the framework, the storage server is able to perform test on encrypted data and return the matched value without knowing the data or the decryption key. Unlike the previous work, a two-layered access control is supported in our proposed framework. On the first layer, the access control mechanism is provided by the trusted private cloud to realize users’ authorization and revocation. On the second layer, the access control mechanism is controlled by data owner and to restrict users’ access to the encrypted data. The security analysis shows that proposed framework achieves message privacy as
well as keyword privacy.

ACKNOWLEDGEMENT

This research is supported by the National Natural Science Foundation of China (Grant No. 60973141) and the Specialized Research Fund for the Dotoral Program of Higher Education of China (Grant No. 20100031110030). In addition, Jin Li is supported by the National Natural Science Foundation of China (Grant No. 61100224) and Grant for Universities in Guangzhou City (Grant No. 10A008) and Natural Science Foundation of Guangdong Province (Grant No. 10451009101004573). Zheli Liu is supported by the Fundamental Research Funds for the Central Universities.

REFERENCES