A Secure Cloud Data Sharing Protocol for Enterprise Supporting Hierarchical Keyword Search

Hongbo Li, Qiong Huang, Willy Susilo, *Senior Member, IEEE*

***Abstract*—Cloud storage becomes the priority for storing and sharing data for enterprise users. Encrypting prior to uploading data to the cloud is the best way to protect business secrets, however, it hinders the convenient operations on plaintexts, such as searching over the cloud data. In addition, employees in an enterprise have multiple layer structures and a higher layer employee should have the privilege to monitor the lower layer employees' data to check if these users violate the regulation without letting the employees be aware of. Public key encryption with keyword search (PEKS) is a well-known cryptographic primitive suitable for secure cloud storage, which supports keyword search without decryption in public key encryption settings. Unfortunately, no existing PEKS scheme supports the monitoring function without authorization from the sender. To address this issue, we propose a variant of PEKS named Hierarchical Public Key Encryption with Keyword Search (HPEKS) and provide a semi-generic construction utilizing a public key tree (PKTree) and a PEKS scheme. To better suit for the enterprise secret data sharing, we build an advanced HPEKS scheme, named designated-tester decryptable hierarchical public key encryption with keyword search (dDHPEKS), which enjoys stronger security and integrates the public key and symmetric key encryptions. We prove our dDHPEKS scheme secure under the security definition in the random oracle model. Particularly, it satisfies the security against outside offline keyword guessing attacks and furthermore, enjoys the *transparency* property so that the sender does not need to know the internal hierarchy structure of an enterprise in order to share encrypted data to the enterprise. Theoretical evaluation and concrete experiments show that our dDHPEKS scheme has comparable running efficiency with existing PEKS schemes.**

***Index Terms*—public key encryption, keyword search, keyword guessing attacks, secret data sharing, cloud storage.**

I. Introduction

WITH the rapid development of computing and communication technology, data generated by enterprises or firms expand exponentially with very high speed. Due to the low cost of data management, the public cloud service (PCS) becomes the top priority for most enterprises. Statistics show that most enterprises used at least one PCS [1]. It allows users to access the mass of data everywhere using storage limited mobile devices via the internet. One of the services supplied by the PCS is to store and manage messages and documents sent to users. However, the risk of a security breach also limits some enterprises to use the PCS [1]. According to the statistics, security concern is the most important issue for respondents [1]. Encrypting prior to uploading is a direct measure to counter-attack this risk, but it limits the application on the data.

H. Li and Q. Huang are with College of Mathematics and Informatics, South China Agricultural University, Guangzhou 510642, China. Q. Huang is also with Guangzhou Key Laboratory of Intelligent Agriculture, Guangzhou, China. Emails: {hongbo,qhuang}@scau.edu.cn.

W. Susilo is with Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Wollongong NSW 2522, Australia. Email: wsusilo@uow.edu.au.

Q. Huang is the corresponding author.

This work is supported by National Natural Science Foundation of China (61872152), the Major Program of Guangdong Basic and Applied Research (2019B030302008), Guangdong Natural Science Funds for Distinguished Young Scholar (2014A030306021) and Guangdong Program for Special Support of Top-notch Young Professionals (2015TQ01X796).

A. Motivation - The need for hierarchical structure

The Office Automation (OA) system is one of the applications of the cloud computing. Thousands of messages are generated everyday from an enterprise or the government. In the real world, the OA services are often outsourced to the cloud service provider, because it owns a huge advantage in costs of data storage and management. Messages in the OA system are often stored as plaintexts, which make it possible for adversaries to access and accordingly would lead to economic loss or unfairness for the public.

For instance, a project undertaken by an enterprise would be obstructed by its business competitor if the related information leaks in advance. In another example, the leakage of some government-scheduled policy would give someone a head start, which is unfair to the rest of the public.

To avoid this situation, there should be a reasonable access control strategy, which allows authorized users to access the corresponding messages and denies the access from unauthorized user or the less privileged users. For example, in an enterprise, the chief executive officer (CEO) supervises various other employees, such as chief operating officer (COO), chief financial officer (CFO), chief information officer (CIO) and chief technology officer (CTO). The COO, CFO, CIO and CTO supervise several employees, respectively. To allow the CEO's access to the message sent from COO to CIO helps the CEO effectively manage the enterprise. To deny the CFO's access to the message sent from CTO to CEO can reduce the risk of information leakage.

Even though the OA system has an access control strategy [1], once the system is broken in, all the information leaks. Hence, the ciphertext-level access control strategy is a reasonable and important way to protect data privacy. The basic idea of the ciphertext-level access control strategy is to encrypt messages with different users' public key, in which way even if the OA system is broken in, only ciphertexts are exposed, which leaks

[1]The system-level access control strategy is out of our research scope and is thus omitted in this paper.

less information. For this scenario, the encryption scheme shell meet the following requirements.

1) Searchability. It is inefficient to download and decrypt all the ciphertexts to find some specific documents, such as documents corresponding to the keyword "contract". To efficiently find these documents without leaking information about the content of the document, a method of searching over ciphertexts without decryption is needed.

2) Access control based on user priority. A message or a task in the real world often needs to be sent to and handled by different users. The encrypted message should only be decrypted by the user who has the corresponding or higher access-priority. Considering the structure of the receivers, it is reasonable to apply a tree-like access control strategy to decrypt the ciphertexts.

3) Transparency. It is not necessary for a sender Alice to know the internal structure of the organization the receiver is in. When Alice sends an encrypted message to a receiver Bob, she only needs to know Bob's valid public key. In other words, Alice does not need to know Bob's priority in the enterprise neither who in the enterprise has a higher priority than Bob.

To the best of our knowledge, there is no corresponding cryptographic primitive meeting the above requirements simultaneously. The public key encryption with keyword search (PEKS) supports searching some keyword over ciphertexts without decryption, but it lacks the mechanism of access control. Attribute-based encryption (ABE) and Hierarchical identity-based encryption (HIBE) support ciphertext-level access control. It seems feasible to combine ABE or HIBE with PEKS to construct a suitable searchable encryption scheme, however, in ABE and HIBE, the sender needs to know the receiver's internal organization structure.

Therefore, it is essential to construct a new cryptographic primitive suitable for the aforementioned cloud data sharing scenario.

B. Contributions

1) Public Key Tree: To resolve the problem above, we introduce the public key tree (PKTree) into the searchable encryption and propose a new notion named *hierarchical public key encryption with keyword search* (HPEKS). In HPEKS, it allows users to search over ciphertexts encrypted with their public keys. Specially, if there is a group of users with a hierarchical structure, the user with higher access permission can search over ciphertexts sent to users with lower access permission. For example, if Alice supervises Bob and Carlos, Alice can perform searching operations over ciphertexts encrypted with Bob's and Carlos' public keys.

2) Semi-generic HPEKS Construction from PEKS: To demonstrate the feasibility, we give a semi-generic construction HPEKS leveraging the proposed PKTree technique together with an existing bilinear-pairing based PEKS scheme in the literature.

3) Advanced HPEKS Scheme dDHPEKS: To resist outside offline keyword guessing attacks, we propose an advanced HPEKS scheme, named *designated-tester decryptable hierarchical public key encryption with keyword search* (dDHPEKS), in which only the designated server can search for users. Moreover, our proposal integrates PEKS and PKE, which means it supports not only keyword search but also decryption. Our dDHPEKS scheme enjoys an interesting property, *transparency*, meaning that the sender does not need to know the internal hierarchy structure of the organization that the receiver is in before encrypting a keyword for the receiver. Instead, the sender needs to encrypt the keyword under the public key of the intended receiver only. In contrast, HIBE or ABE integrated with PEKS does not support this property.

4) Security and Efficiency: We give formal security definition for dDHPEKS, including secret key security, keyword privacy and plaintext privacy, and prove our dDHPEKS scheme secure under the given security definitions. We analyze the running overhead of dDHPEKS theoretically and implement it utilizing C language and PBC library [2]. The analysis and experiment results show that our dDHPEKS scheme has comparable running overhead with existing PEKS schemes.

C. Related Works

The notion of searchable symmetric encryption (SSE) and the first searchable encryption scheme was proposed by Song *et al.* [3] in 2000. It allows a user who has the searching key to search some keyword over ciphertexts without decryption. However, Song *et al.*'s scheme and other schemes [4][5][6] based on it have a common restriction that it is hard to share the searching key with others, thus are only suitable for data owner itself to search over the ciphertexts. Boneh *et al.* [7] introduced SE into the public key settings and proposed the first public key SE scheme named Public Key Encryption with Keyword Search (PEKS), which breaks the barrier of data sharing limit. In PEKS, a data owner (encrypter) performs the encryption algorithm using a receiver's (searcher's) public key, and in reverse, the receiver generates a trapdoor to search some keyword over ciphertexts using its secret key.

Other issues in PEKS have also engaged researchers' attention. Park *et al.* and Golle *et al.* proposed schemes named public key encryption with conjunctive keyword search (PECKS) [8][9] which support receivers to search for documents containing all of several keywords in only a single query. To enable a ranked list of the searching result, a new notion called multi-keyword ranked search (MRSE) was proposed [10] [11][12]. Bao *et al.* [13] and others [14] [15][16] proposed searchable encryption schemes in multi-receiver settings. To support fine-access control in PEKS, attribute-based encryption with keyword search (ABEKS) was proposed [17] [14]. Byun *et al.* [18] and Yau *et al.* [19] pointed out that Boneh *et al.*'s PEKS scheme and other existing PEKS schemes are vulnerable under the new attack named offline keyword guessing attacks (KGA). They also emphasized that almost all the existing PEKS cannot resist the offline keyword guessing attacks given by inside adversaries (IKGA), e.g. KGA given by searching server. Fang *et al.* [20] proposed a secure-channel free PEKS scheme witch resists the KGA efficiently, however, cannot resist IKGA. Huang and Li proposed a new notion called public key authenticated encryption with keyword search (PAEKS) [21][22] which resists IKGA by denying adversaries the ability to encrypt keywords. He *et al.* [23] and Li *et al.* [24] introduce PAEKS into Certificateless public key encryption and identity based encryption settings, respectively. Chen *et al.* [25] and Chen *et al.* [26] also solved the IKGA problem by utilizing two server in the system model.

D. Paper Organization

In the next section, we briefly introduce some preliminaries and the system model of our HEPKS. We introduce the public key tree structure and its construction in Section III. We then propose our HPEKS schemes in Section IV, and analyze the security in Section V. We discuss about the efficiency of our scheme and compare our scheme with some other related schemes in Section VI, and finally conclude the paper in Section VII.

II. Preliminaries

A. Bilinear Pairing

Let $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ be three groups with same prime order p, g and h be any two generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. There is a map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ from $\mathbb{G}_1$ and $\mathbb{G}_2$ to $\mathbb{G}_T$. We say $\hat{e}$ is a bilinear pairing map [27] if the following conditions are satisfied:

- Bilinearity: $\forall x, y \in \mathbb{Z}_p$, $\hat{e}(g^x, h^y) = \hat{e}(g, h)^{x \cdot y}$.
- Non-degeneracy: $\hat{e}(g, h) \neq 1$.
- Computability: $\hat{e}(g, h)$ is easily computable.

B. Computational Diffie-Hellman (CDH) Assumption

Definition 1 (CDH Problem): Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear pairing map. Given $g, g^x, g^y \in \mathbb{G}_1$, the CDH problem is to calculate g^{xy}.

Definition 2 (CDH Assumption [28]): The CDH assumption is that the probability that any probabilistic polynomial time (PPT) adversary solves the CDH problem is negligible .

C. Decisional Linear (DLIN) Assumption

Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear pairing. Given $u, u^x \in \mathbb{G}_1, v, v^y, h \in \mathbb{G}_2$, the DLIN assumption is that the advantage of distinguishing h^{x+y} from a random $h^r \in \mathbb{G}_2$ is negligible for any PPT adversary [29]:

$$\begin{aligned} Adv_{\mathcal{A}}^{DLIN} = |&\Pr[\mathcal{A}(u, u^x, v, v^y, h, h^{x+y}) = 1] - \\ &\Pr[\mathcal{A}(u, u^x, v, v^y, h, h^r) = 1]| \leq \mathrm{negl}(\lambda). \end{aligned}$$

D. Decisional Bilinear Diffie-Hellman (DBDH) Assumption

Definition 3 (DBDH Problem): Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear pairing map. Given $g, g^x, g^y \in \mathbb{G}_1, h, h^y, h^z \in \mathbb{G}_2$, the DBDH problem is to distinguish $\hat{e}(g, h)^{xyz}$ from a random element $R \in \mathbb{G}_T$, where $x, y, z \in \mathbb{Z}_p$ are unknown.

Definition 4 (DBDH Assumption): The DBDH assumption is that the advantage of solving the DBDH problem for any PPT adversary is negligible [28]:

$$\begin{aligned} Adv_{\mathcal{A}}^{DBDH} = |&\Pr[\mathcal{A}(g, g^x, g^y, h, h^y, h^z, \hat{e}(g, h)^{xyz}) = 1] - \\ &\Pr[\mathcal{A}(g, g^x, g^y, h, h^y, h^z, R) = 1]| \leq \mathrm{negl}(\lambda). \end{aligned}$$

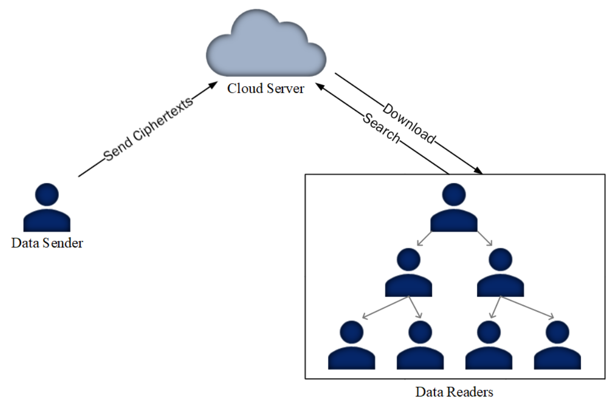

Fig. 1. System Model of Tree-based Hierarchical Multi-receiver Searchable Encryption

E. System Model

Here we present the system model of the HPEKS system (Figure 1). There are three entities in the system, denoted by data sender, data receivers and cloud server, respectively.

- **Data sender**: It encrypts data with a receiver's public parameter and sends the ciphertext to the receiver via the cloud server.
- **Data receivers**: The data receivers have a tree-based hierarchical structure. Each data receiver is denoted by a node in the tree. A data receiver can search over ciphertexts sent to itself and to all its children, not vice versa.
- **Cloud server**: It provides the storing and computing services for data receivers. The public parameters of the group of receivers and the received ciphertexts will be stored by the cloud server. It also supports receivers to run algorithms to search over the ciphertexts.

III. The Public Key Tree

To build HPEKS scheme, we introduce a new notion named *public key tree* (PKTree), which is is a multi-way tree. Each node of a PKTree contains a user's public information, and we say the user who has the corresponding secret key is the owner of the node. As mentioned above, there is a group of data receivers. There is a supervisor $\mathbf{Rt}$ of the group, which initially setups the system and holds the master secret key msk. Using msk, $\mathbf{Rt}$ generates the secret key $\mathsf{sk}_{\mathbf{Rt}}$ of itself and the root node containing $\mathbf{Rt}$'s public information. After the initialization, $\mathbf{Rt}$ should add at least one child node. For instance, $\mathbf{Rt}$ will add a node named Node_i for receiver R_i. Firstly, $\mathbf{Rt}$ gathers the public information $\overline{\mathbf{ID}}_i$ of R_i and generates a secret key sk_i and a public parameter Pub_i for R_i using $(\overline{\mathbf{ID}}_i, \mathsf{sk}_{\mathbf{Rt}})$. Secondly, $\mathbf{Rt}$ signs Pub_i to generate a signature $\sigma_{\mathbf{Rt}}(\mathsf{Pub}_i)$. Finally, $\mathbf{Rt}$ sends sk_i to R_i via a secure channel and adds the node Node_i, containing Pub_i and $\sigma_{\mathbf{Rt}}(\mathsf{Pub}_i)$, into the PKTree. Notice that, in order to resist adversaries from forging a child node, a signature for the content is essential. The work of generating Node_i's child nodes is then transferred to R_i. In this way, each receiver can add its own child nodes itself, which shares the workload of building a PKTree. Another feature of the PKTree is that a receiver can also add a grandchild node directly. Specially, the supervisor $\mathbf{Rt}$ can add a child node for any of its descendant nodes. However, owners of two nodes in different branch cannot add a child node for each other, no matter which node is closer to the root than the other.

Figure 2 shows an example of the PKTree. There is an enterprise and the owner of the enterprise supervises this PKTree. The owner setups the system, publishes the system parameter $\mathcal{GP}$ and holds the key msk secretly. The root node of the PKTree will be assigned to the CEO of the enterprise (Level 1 in Fig. 2). The root node contains the CEO's public information $\mathsf{Pub}_{\mathrm{CEO}}$ and a signature $\mathsf{Sig}_{\mathsf{msk}}(\mathsf{Pub}_{\mathrm{CEO}})$ produced by the enterprise owner. Notice that, $\mathsf{Pub}_{\mathrm{CEO}}$ is generated by the enterprise owner and includes the CEO's identity information, and the $\mathsf{Sig}_{\mathsf{msk}}(\mathsf{Pub}_{\mathrm{CEO}})$ serves as a proof on the validity of $\mathsf{Pub}_{\mathrm{CEO}}$.

Similar to the CEO, the COO, CFO and CTO will be assigned with nodes by the CEO (Level 2 in Fig. 2). Each node in the PKTree contains an employee's public information. For instance, the CTO's public information is generated and signed by the CEO. The child nodes of the CTO's node will be generated by CTO instead of CEO. In this way, it allows every employee of the enterprise to add nodes into the PKTree and share the workload of building the tree. As mentioned above, the CFO can add child nodes for $\mathsf{Node}_{\mathrm{Alice}}$ and $\mathsf{Node}_{\mathrm{Bob}}$, but not for $\mathsf{Node}_{\mathrm{Carter}}$ nor $\mathsf{Node}_{\mathrm{David}}$.

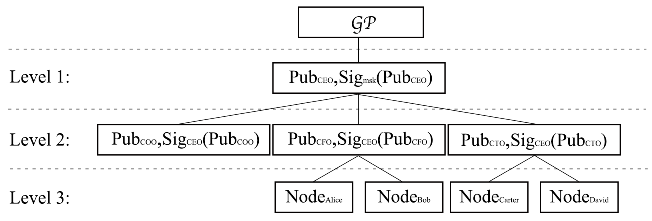

Fig. 2. An Example of Public Key Tree

A. Definition of PKTree

A public key tree (PKTree) consists of the following four algorithms.

- $\mathsf{Setup}(1^\lambda)$: Given the security parameter 1^λ, this algorithm outputs a master secret key for the system msk and the global parameter $\mathcal{GP}$.
- $\mathsf{BuildTree}(\mathcal{GP}, \mathsf{msk}, \mathrm{ID}_{\mathbf{Rt}}, \tau)$: Given the global parameter $\mathcal{GP}$, the master secret key msk, the identity $\mathrm{ID}_{\mathbf{Rt}}$ of the root receiver $\mathbf{Rt}$, and a time stamp τ, this algorithm outputs the root node Node_{root} of a public key tree Tree and the secret key sk_{root} of the node. The Node_{root} contains the identity $\mathrm{ID}_{\mathbf{Rt}}$ and a signature σ_{msk} signed with the master secret key msk. The signature will be used to verify if the Node_{root} is valid and bound to the identity of the root receiver $\mathbf{Rt}$.
- $\mathsf{AddNode}(\mathcal{GP}, \mathsf{Tree}, \mathsf{sk}_i, \mathrm{ID}_j, \tau_j)$: Given $\mathcal{GP}, \mathsf{Tree}$, the secret key sk_i of node Node_i, an identity ID_j and a time stamp τ, this algorithm outputs a child node Node_j of Node_i and Node_j's corresponding secret key sk_j. Notice that the time stamp included in a child node is valid if and only if it is newer than that in the parent node. Similar to the BuildTree algorithm, the newly added child node Node_j contains the identity ID_j and a signature $\sigma_{i,j}$ signed with the parent node's secret key sk_i.
- $\mathsf{DeleteNode}(\mathcal{GP}, \mathsf{Tree}, \mathsf{sk}_i, \tau_i')$: Given $\mathcal{GP}, \mathsf{Tree}$, the secret key sk_i of node Node_i and a new time stamp τ_i', this algorithm deletes all the child nodes of Node_i and renew the Node_i using the new time stamp τ_i'. Because the τ_i' will be newer than any of the child nodes, all the old child nodes will be invalid. Hence, it deletes the child nodes.
- $\mathsf{VerifyTree}(\mathcal{GP}, \mathsf{Tree})$: Given $\mathcal{GP}$ and Tree, this algorithm outputs 1 if the time stamp of every node is newer than its parent node and the signature of every node is validly signed by its parent node (the root node is signed by the master key msk), and outputs 0 otherwise.

B. The proposed PKTree

Based on the bilinear pairing, we give a concrete PKTree which will be utilized to build the HPEKS schemes.

- $\mathsf{Setup}(1^\lambda)$: Select three groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ with the same prime order p and a bilinear pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. Randomly select a master secret key $\mathsf{msk} = k \in \mathbb{Z}_p$ and compute the master public key $\mathsf{MPK} = g^k \in \mathbb{G}_1$, where g is a generator of $\mathbb{G}_1$. Finally, this algorithm outputs the global parameter $\mathcal{GP} = \{\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, h, \hat{e}, \mathsf{MPK}, H^*, H_1, H_2\}$ in which h is a generator of $\mathbb{G}_2$ and $H^* : \{0,1\}^* \rightarrow \mathbb{G}_2, H_1 : \{0,1\}^* \rightarrow \mathbb{G}_2, H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_p$ are cryptographic hash functions.
- $\mathsf{BuildTree}(\mathcal{GP}, \mathsf{msk}, \mathrm{ID}_{\mathbf{Rt}}, \tau)$: Given $\mathcal{GP}, \mathsf{msk}$, $\mathbf{Rt}$'s identity $\mathrm{ID}_{\mathbf{Rt}}$ and a time stamp τ, $\mathbf{Rt}$ generates the first node of the tree as follow.
  - Randomly select a salt ς, and generate $\mathbf{Rt}$'s public parameter $\mathsf{Pub}_{\mathbf{Rt}} = (\mathsf{pk}_{\mathbf{Rt}}, \overline{\mathbf{ID}}_{\mathbf{Rt}})$ and secret key $\mathsf{sk}_{\mathbf{Rt}} = H_1(\overline{\mathbf{ID}}_{\mathbf{Rt}})^{\mathsf{msk}}$, where $\mathsf{pk}_{\mathbf{Rt}} = g_1^{H_2(\mathsf{sk}_{\mathbf{Rt}})}$ and $\overline{\mathbf{ID}}_{\mathbf{Rt}} = (\mathrm{ID}_{\mathbf{Rt}}, \varsigma)$.
  - Sign $\overline{\mathbf{ID}}_{\mathbf{Rt}}$ with the master secret key msk to generate a signature $\sigma_{0,\mathbf{Rt}} = \mathsf{Sig}_{\mathsf{msk}}(\mathsf{Pub}_{\mathbf{Rt}}) = (\sigma_{0,\mathbf{Rt},1}, \sigma_{0,\mathbf{Rt},2}) = (H^*(\mathsf{Pub}_{\mathbf{Rt}} \| \tau)^{\mathsf{msk}}, \tau)$.
  - Set the root node of the PKTree as $\mathsf{Node}_{\mathbf{Rt}} = (\mathsf{Pub}_{\mathbf{Rt}}, \sigma_{\mathbf{Rt}})$ and upload $\mathsf{Tree} = \{(\mathsf{Node}_{\mathbf{Rt}}, \mathrm{P}_0)\}$ to the cloud. Here P_0 indicates the position of the $\mathsf{Node}_{\mathbf{Rt}}$, *i.e.* the root, in the Tree.

  Remark. The salt ς is used to randomize the receiver's public parameter. Users can verify the validity of the root node by check whether the following equation holds:

  $$\hat{e}(g, \sigma_{0,\mathbf{Rt},1}) = \hat{e}(\mathsf{MPK}, H^*(\mathsf{Pub}_{\mathbf{Rt}} \| \sigma_{0,\mathbf{Rt},2})).$$
- $\mathsf{AddNode}(\mathcal{GP}, \mathsf{Tree}, \mathsf{sk}_i, \mathrm{ID}_j, \tau_j)$: This algorithm is run by a receiver i, who owns a public parameter in the Tree, to generate a child node for a new receiver j. Given $\mathcal{GP}$, Tree, the receiver i's secret key sk_i, the lower-layer receiver j's identity ID_j and the time stamp τ_j, the

algorithm runs the following steps and outputs the new node Node_j and its corresponding secret key sk_j:

- Randomly select a salt ς_j and generate the public parameter $\mathsf{Pub}_j = (\mathsf{pk}_j, \overline{\mathbf{ID}}_j)$ and secret key $\mathsf{sk}_j = H_1(\overline{\mathbf{ID}}_j)^{H_2(\mathsf{sk}_i)}$, where $\mathsf{pk}_j = g^{H_2(\mathsf{sk}_j)}$ and $\overline{\mathbf{ID}}_j = (\mathrm{ID}_j, \varsigma_j)$;
- Sign $\overline{\mathbf{ID}}_j$ with sk_i and generate a signature

$$\begin{aligned}\sigma_{i,j} &= \mathsf{Sig}_{\mathsf{sk}_i}(\mathsf{Pub}_j) = (\sigma_{i,j,1}, \sigma_{i,j,2}) \\ &= (H^*(\mathsf{Pub}_j \| \tau_j)^{H_2(\mathsf{sk}_i)}, \tau_j);\end{aligned}$$

- Set the child node as $\mathsf{Node}_j = (\mathsf{Pub}_j, \sigma_{i,j})$.

Remark. The time stamp τ_j of the new node Node_j must be newer than that of the parent node Node_i. Users can verify the validity of this node by check whether the following equation holds:

$$\hat{e}(g, \sigma_{i,j,1}) = \hat{e}(\mathsf{pk}_i, H^*(\mathsf{Pub}_j \| \sigma_{i,j,2})).$$

- $\mathsf{DeleteNode}(\mathcal{GP}, \mathsf{sk}_i, \mathsf{Node}_j, \tau_j')$: This algorithm revokes all the child nodes of Node_j. Given $\mathcal{GP}$, the receiver's secret key sk_i, the child node Node_j, this algorithm runs as follows:
  - Catch the current time stamp τ_j' and generate a new signature $\sigma_{i,j}' = \mathsf{Sig}_{\mathsf{sk}_i}(\mathsf{Pub}_j) = ((H^*(\mathsf{Pub}_j \| \tau_j')^{H_2(\mathsf{sk}_i)}, \tau_j'));$
  - Generate a new node $\mathsf{Node}_j' = (\mathsf{Pub}_j, \sigma_{i,j}')$;
  - Replace Node_j with Node_j' in the Tree and delete all the child nodes of Node_j. Note that, the secret key of the Node_j' is not changed and same as that of Node_j.

Remark. A naive method to revoke the child nodes is to delete them from Node_j directly. However, it cannot be proved that the child nodes have been indeed deleted, because the signatures attached to them and the time stamps of the child nodes are still valid, which may mislead other users. A better solution to the problem is to update the time stamp of the parent node (e.g. Node_j). According to our design of PKTree, time stamps of child nodes should be newer than that of the parent node. Therefore, if we update the parent node's time stamp with the latest one (as well as the corresponding signature on the new time stamp), all the child nodes are invalidated immediately. However, in many cases we do not need to delete all the child nodes. Hence, we design a two-layer-per-receiver structure, i.e. each receiver has two layer nodes. The upper-layer node is assigned by the receiver's parent node and the lower-layer nodes are generated by the receiver itself. Each lower-layer node is associated with a child node. In case the receiver needs delete one of the child nodes, it simply updates the associated lower-layer node, to make the time stamp of the lower-layer node newer than the child node. In such a way, it completes the deletion of the to-be-deleted child node, and keeps the other child nodes unaffected.

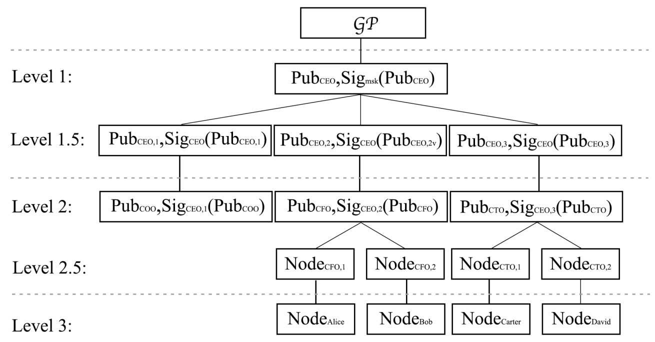
$\mathcal{GP}$
Level 1:
Pub_CEO,Sig_msk(Pub_CEO)
Level 1.5:
Pub_CEO,1,Sig_CEO(Pub_CEO,1)
Pub_CEO,2,Sig_CEO(Pub_CEO,2v)
Pub_CEO,3,Sig_CEO(Pub_CEO,3)
Level 2:
Pub_COO,Sig_CEO,1(Pub_COO)
Pub_CFO,Sig_CEO,2(Pub_CFO)
Pub_CTO,Sig_CEO,3(Pub_CTO)
Level 2.5:
Node_CFO,1
Node_CFO,2
Node_CTO,1
Node_CTO,2
Level 3:
Node_Alice
Node_Bob
Node_Carter
Node_David

Fig. 3. The Concrete PKTree

- $\mathsf{VerifyTree}(\mathcal{GP}, \mathsf{Tree})$: The Tree is valid if and only if, for all nodes in the tree, the equations

$$\begin{aligned}\hat{e}(g, \sigma_{0,\mathbf{Rt},1}) &= \hat{e}(\mathsf{MPK}, H^*(\mathsf{Pub}_{\mathbf{Rt}} \| \sigma_{0,\mathbf{Rt},2})) \\ \hat{e}(g, \sigma_{i,j,1}) &= \hat{e}(\mathsf{pk}_i, H^*(\mathsf{Pub}_j \| \sigma_{i,j,2}))\end{aligned}$$

hold, where $\mathbf{Rt}$ is the subscript of the root node and i, j indicate the subscripts of each pair of parent-child nodes.

C. Application of PKTree

Figure 3 shows the construction of the PKTree in the real-world scenario. Before building the PKTree, the global parameter $\mathcal{GP}$ should be given by the enterprise owner. Slightly different from the aforementioned example, a receiver could own multiple nodes, e.g. the CEO owns a Level-1 node (assigned by the enterprise owner) and three Level-1.5 nodes (assigned by itself) in the PKTree.

To delete a node, e.g. the CTO's node, as shown in Fig. 4, the CEO runs the DeleteNode algorithm to update the Level-1.5 node $\mathsf{Node}_{\mathsf{CEO},3}$ which is the parent node of the CTO's node. The public parameter $\mathsf{Pub}_{\mathsf{CEO},3}$ in the updated node $\mathsf{Node}_{\mathsf{CEO},3}$ is not changed but the corresponding signature $\mathsf{Sig}_{\mathsf{CEO}}{}^{new}$ is updated with the latest time stamp $\tau_{\mathsf{CEO},3}{}^{new}$. Because the $\tau_{\mathsf{CEO},3}{}^{new}$ is newer than the time stamps of its child nodes, all its child nodes will be invalid, in which way the child nodes are revoked. Notice that the rest branches of the $\mathsf{Node}_{\mathsf{CEO}}$ are not effected.

In many cases, the CEO does not want to revoke all of the CTO's child nodes. Thus, after deleting the CTO's node, the CEO should assign a new node to an employee to manage the CTO's child nodes. For example, as shown in Fig. 5, to replace the CTO with the employee "*David*", the CEO runs the AddNode algorithm to replace the CTO's node with David's new node and sends David the corresponding secret key $\mathsf{sk}_{\mathsf{David}}$ via a secure channel. Finally, David and his child nodes use the AddNode algorithm to build up this branch iteratively.

IV. The Proposed HPEKS Schemes

In this section, based on the proposed PKTree, we introduce how to build an HPEKS scheme by giving a semi-generic HPEKS scheme. To keep the integrality of the encrypted keywords and plaintext, we give an enhanced concrete HPEKS scheme which supports decrypting ciphertexts.

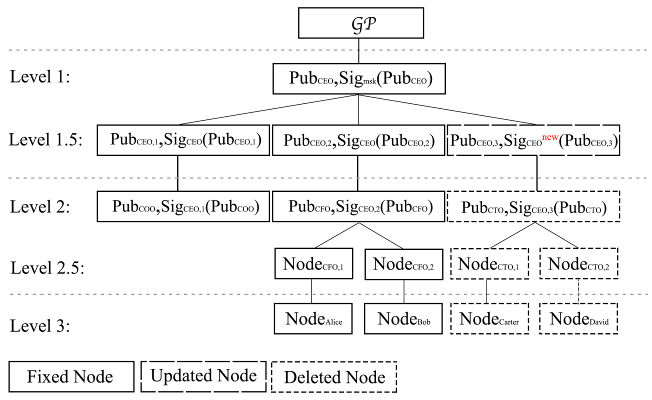
$\mathcal{GP}$
Level 1:
Pub$_{CEO}$,Sig$_{msk}$(Pub$_{CEO}$)
Level 1.5:
Pub$_{CEO,1}$,Sig$_{CEO}$(Pub$_{CEO,1}$)
Pub$_{CEO,2}$,Sig$_{CEO}$(Pub$_{CEO,2}$)
Pub$_{CEO,3}$,Sig$_{CEO}^{new}$(Pub$_{CEO,3}$)
Level 2:
Pub$_{COO}$,Sig$_{CEO,1}$(Pub$_{COO}$)
Pub$_{CFO}$,Sig$_{CEO,2}$(Pub$_{CFO}$)
Pub$_{CTO}$,Sig$_{CEO,3}$(Pub$_{CTO}$)
Level 2.5:
Node$_{CFO,1}$
Node$_{CFO,2}$
Node$_{CTO,1}$
Node$_{CTO,2}$
Level 3:
Node$_{Alice}$
Node$_{Bob}$
Node$_{Carter}$
Node$_{David}$
Fixed Node
Updated Node
Deleted Node

Fig. 4. Delete Nodes from the PKTree

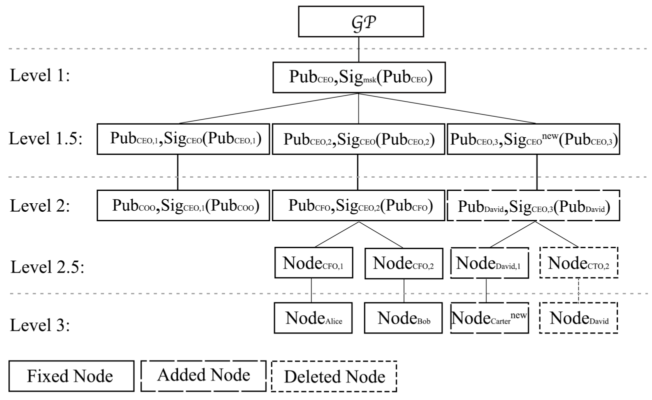
$\mathcal{GP}$
Level 1:
Pub$_{CEO}$,Sig$_{msk}$(Pub$_{CEO}$)
Level 1.5:
Pub$_{CEO,1}$,Sig$_{CEO}$(Pub$_{CEO,1}$)
Pub$_{CEO,2}$,Sig$_{CEO}$(Pub$_{CEO,2}$)
Pub$_{CEO,3}$,Sig$_{CEO}^{new}$(Pub$_{CEO,3}$)
Level 2:
Pub$_{COO}$,Sig$_{CEO,1}$(Pub$_{COO}$)
Pub$_{CFO}$,Sig$_{CEO,2}$(Pub$_{CFO}$)
Pub$_{David}$,Sig$_{CEO,3}$(Pub$_{David}$)
Level 2.5:
Node$_{CFO,1}$
Node$_{CFO,2}$
Node$_{David,1}$
Node$_{CTO,2}$
Level 3:
Node$_{Alice}$
Node$_{Bob}$
Node$_{Carter}^{new}$
Node$_{David}$
Fixed Node
Added Node
Deleted Node

Fig. 5. Add Nodes into the PKTree

A. The Semi-Generic HPEKS Scheme from PEKS

Thanks to the compatibility of PKTree, we can construct a semi-generic HPEKS scheme from pairing-based PEKS schemes. Let $\mathsf{PEKS} = \{\mathsf{KeyGen}', \mathsf{Encrypt}', \mathsf{Trapdoor}', \mathsf{Test}'\}$ be a pairing-based PEKS scheme in which the secret key sk is an element of $\mathbb{Z}_p$ and the public key is of the form g^{sk}.

- $\mathsf{Encrypt}(\mathcal{GP}, \mathsf{Tree}, \mathsf{Node}_j, w)$: Parse the receiver's node Node_j as $(\mathsf{pk}_j, \overline{\mathbf{ID}}_j, \sigma_{i,j})$ and verify the validity of the public parameter $\mathsf{pk}_j, \overline{\mathbf{ID}}_j$ and the signature $\sigma_{i,j}$. Run the PEKS $\mathsf{Encrypt}'(\mathcal{GP}, \mathsf{pk}_j, w)$ algorithm and return the generated ciphertext C_w.
- $\mathsf{Trapdoor}(\mathcal{GP}, \mathsf{sk}_j, w)$: This algorithm returns the trapdoor T_w generated by PEKS $\mathsf{Trapdoor}'(\mathcal{GP}, H_2(\mathsf{sk}_j), w)$.
- $\mathsf{Test}(\mathcal{GP}, C, T_w)$: This algorithm returns the trapdoor T_w generated by PEKS $\mathsf{Test}'(\mathcal{GP}, C, T_w)$.

Remark. The Setup, $\mathsf{BuildTree}$, $\mathsf{AddNode}$, $\mathsf{DeleteNode}$ and $\mathsf{VerifyTree}$ algorithms of the semi-generic HPEKS scheme are the same as those in Section III-B and thus are omitted here.

B. designated-tester Decryptable HEPKS Scheme

The semi-generic HPEKS scheme supports a group of users to delegate the cloud server to search some keyword over the received ciphertexts without decryption. Specially, a receiver can search over ciphertexts sent to receivers supervised by him. However, it is based on the existing PEKS schemes and the traditional PEKS schemes do not support the decryption function. Although the PKE/PEKS scheme proposed by Baek *et al.* [30] supports decryption of ciphertexts, there is a restriction that the plaintext in the PKE/PEKS scheme is length-limited. To resolve this problem, we propose an advanced scheme named *designated-tester decryptable hierarchical public key encryption with keyword search* (dDHPEKS), in which plaintext of variable length can be encrypted and the corresponding ciphertext is decryptable. Additionally, only the designated tester can test whether a ciphertext contains the same keyword as that in the queried trapdoor.

- $\mathsf{Setup}(1^\lambda)$: Given a security parameter λ, this algorithm setups the system via running the following steps.
  - Pick three groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of same prime order p with a bilinear pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.
  - Randomly select two generators $g \in \mathbb{G}_1$, $h \in \mathbb{G}_2$.
  - Randomly select $\mathsf{msk} = k \in \mathbb{Z}_p$ and compute $\mathsf{MPK} = g^k$.
  - Select hash functions $H^* : \{0,1\}^* \to \mathbb{G}_2$, $H : \{0,1\}^* \to \mathbb{Z}_p$, $H_1 : \{0,1\}^* \to \mathbb{G}_2$, $H_2 : \mathbb{G}_2 \to \mathbb{Z}_p$, $H_3 : \{0,1\}^* \to \{0,1\}^{\ell_1}$, $H_4 : \{0,1\}^* \to \mathbb{G}_2$, $H_5 : \{0,1\}^* \to \{0,1\}^{\ell_2}$, where ℓ_1, ℓ_2 are two fixed lengths.
  - Output the msk and the $\mathcal{GP} = (\mathbb{G}_1,\ \mathbb{G}_2,\ \mathbb{G}_T,\ p,\ \hat{e},\ g,\ h,\ \mathsf{MPK},\ H^*,\ H,\ H_1,\ H_2,\ H_3,\ H_4,\ H_5)$.
- $\mathsf{KeyGen}_{server}(\mathcal{GP})$: Given the global parameter $\mathcal{GP}$, this algorithm randomly selects $\mathsf{sk}_s \in \mathbb{Z}_p$ as the designated test server's secret key and computes the corresponding public key $\mathsf{pk}_s = h^{\mathsf{sk}_s}$. Output the server's public/secret key pair $(\mathsf{pk}_s, \mathsf{sk}_s)$.
- $\mathsf{Encrypt}(\mathcal{GP}, \mathsf{Tree}, \mathsf{Node}_j, w, M)$: Randomly select $r, s \in \mathbb{Z}_p, K \in \{0,1\}^{\ell_1}$ and encrypt the keyword w and message M as below:

$$\begin{aligned}
&C_1 = \hat{e}(\mathsf{pk}_j^{r \cdot H(w)}, H_1(\overline{\mathbf{ID}}_j)), C_2 = g^r, C_3 = h^r,\\
&C_4 = K \oplus H_3(\mathsf{pk}_j^s), C_5 = g^s, C_7 = \mathsf{AESEnc}_K(M),\\
&C_6 = H_4(C_1, C_2, C_3, C_4, C_5, H_5(C_7))^s.
\end{aligned}$$

  Output the ciphertext $C_{w,M}$ = $(C_1,\ C_2,\ C_3,\ C_4,\ C_5,\ C_6,\ C_7)$.
- $\mathsf{Decrypt}(\mathcal{GP}, \mathsf{sk}_j, C)$: Parse C as $C = (C_1, \ldots, C_7)$. Return $\perp$ if either of the equations

$$\begin{aligned}
&\hat{e}(C_5, H_3(C_1, \ldots, C_5, H_5(C_7))) = \hat{e}(g, C_6),\\
&\hat{e}(C_2, h) = \hat{e}(g, C_3)
\end{aligned}$$

  does not hold, and return the plaintext M otherwise, where

$$M = \mathsf{AESDec}_K(C_7), \quad K = C_3 \oplus H_3(C_5^{H_2(\mathsf{sk}_j)}).$$

- $\mathsf{Trapdoor}(\mathcal{GP}, \mathsf{sk}_j, \mathsf{pk}_s, w)$: Randomly select $t \in \mathbb{Z}_p$ and generate trapdoor as follows:

$$\begin{aligned}
&T_1 = h^{t_1 + t_2} \cdot H_1(\overline{\mathbf{ID}}_j)^{H_2(\mathsf{sk}_j) \cdot H(w)},\\
&T_2 = \mathsf{pk}_s^{t_1}, T_3 = g^{t_2}.
\end{aligned}$$

- $\mathsf{Test}(\mathcal{GP}, C, T, \mathsf{sk}_s)$: Parse C as $(C_1, \ldots, C_7)$ and parse T as (T_1, T_2, T_3). The algorithm outputs 1 if the equation
$$\hat{e}(C_2, T_1/T_2^{\frac{1}{\mathsf{sk}_s}}) = C_1 \cdot \hat{e}(T_3, C_3)$$
holds, and 0 otherwise.

Remark. The BuildTree, AddNode and ReplaceChild algorithms are the same as those in the Semi-Generic HPEKS scheme, and thus omitted here.

Our scheme is advantageous over HIBE (with keyword search) in the sense that it enjoys the property of *transparency*. To share an encrypted record to multiple receivers in a hierarchy, a sender in our scheme does not need to know the hierarchy of these receivers. It only needs to encrypt the record under the public key of the receiver at the lowest level. In contrast, a sender in HIBE does need to know the identity hierarchy.

V. Security Analysis

In this section, we provide the threats model of HPEKS, then give the formal security definitions, and finally prove our proposal is secure under the security definitions.

A. Security Models

1) Secret Key One-Way Security against Collusion: Here we consider the key privacy of the proposed PKTree structure. The following two games are given to prove the semantic security of PKTree against low layer collusion and same layer collusion attacks, respectively.

Game K1: Security against low layer collusion attacks:

- Setup: The challenger $\mathcal{C}$ sends the adversary $\mathcal{A}$ the global parameter $\mathcal{GP}$ and the challenged node Node_i.
- Phase 1: $\mathcal{A}$ can issue at most q_E queries to extract oracle $\mathcal{O}_E$ to obtain Node_i's child nodes secret keys $\{\mathsf{sk}_j\}_1^{q_E}$:
  - $\mathcal{O}_E(\mathrm{ID}_j)$: Given an identity, the oracle returns the corresponding node Node_j and secret key sk_j.
- Guess: $\mathcal{A}$ outputs a secret key $\mathsf{sk}_\mathcal{A}$ and wins the game if $\mathsf{sk}_\mathcal{A} = \mathsf{sk}_i$.

Let $Adv_\mathcal{A}^{K1} = \Pr[\mathsf{sk}_\mathcal{A} = \mathsf{sk}_i]$ denote the advantage of $\mathcal{A}$ to win Game K1.

Game K2: Security against same layer collusion attacks:

- Setup: The challenger $\mathcal{C}$ sends the adversary $\mathcal{A}$ the global parameter $\mathcal{GP}$ and the parent node Node_i.
- Phase 1: $\mathcal{A}$ can issue at most q_E queries to extract oracle $\mathcal{O}_E$ to obtain Node_i's child nodes secret keys $\{\mathsf{sk}_j\}_1^{q_E}$:
  - $\mathcal{O}_E(\mathrm{ID}_j)$: Given an identity, the oracle returns the corresponding node Node_j and secret key sk_j.
- Challenge: $\mathcal{A}$ chooses an identity ID_c as the challenge identity and sends it to $\mathcal{C}$. The constraint is that ID_c cannot be submitted to $\mathcal{O}_E$ in Phase 1. $\mathcal{C}$ returns the corresponding child node $\mathsf{Node}_{\mathrm{ID}_c}$ of Node_i to $\mathcal{A}$.
- Phase 2: $\mathcal{A}$ issues queries to the oracle same as in Phase 1 with the constraint that ID_c cannot appear in the $\mathcal{O}_E$.
- Guess: $\mathcal{A}$ outputs a secret key $\mathsf{sk}_\mathcal{A}$ and wins the game if $\mathsf{sk}_\mathcal{A} = \mathsf{sk}_{\mathrm{ID}_c}$ where sk_i is the secret key of $\mathsf{Node}_{\mathrm{ID}_c}$.

Let $Adv_\mathcal{A}^{K2} = \Pr[\mathsf{sk}_\mathcal{A} = \mathsf{sk}_{\mathrm{ID}_c}]$ denote the advantage of $\mathcal{A}$ to win Game K2.

Definition 5: A dDHPEKS scheme is secure against key collusion attacks if for any PPT adversary $\mathcal{A}$, $Adv_\mathcal{A}^{K1}$ and $Adv_\mathcal{A}^{K2}$ are both negligible.

2) Keyword Privacy against Chosen Keyword Attacks: We give the following security game to define the keyword privacy against chosen keyword attacks.

Game W1: Trapdoor privacy:

- Setup: The challenger $\mathcal{C}$ sends the adversary $\mathcal{A}$ the global parameter $\mathcal{GP}$ and the parent node Node_i.
- Phase 1: $\mathcal{A}$ can issue at most q_E and q_T queries to extract oracle $\mathcal{O}_E$ and trapdoor oracle $\mathcal{O}_T$, respectively.
  - $\mathcal{O}_E(\mathrm{ID}_j)$: Given an identity ID_j, the oracle returns the corresponding node Node_j and secret key sk_j.
  - $\mathcal{O}_T(\mathrm{ID}_j, w)$: Given an identity ID_j and a keyword w, the oracle returns a corresponding trapdoor $T_{\mathrm{ID}_j,w}$.
- Challenge: $\mathcal{A}$ chooses an identity ID_c and two keywords w_0, w_1. The constraint is that ID_c cannot be submitted to $\mathcal{O}_E$ in Phase 1. $\mathcal{C}$ generates the node $\mathsf{Node}_{\mathrm{ID}_c}$ and randomly selects a bit $b \in \{0,1\}$. In the next step, $\mathcal{C}$ generates a trapdoor T_{w_b}. Finally, $\mathcal{C}$ returns $\mathsf{Node}_{\mathrm{ID}_c}$ and T_{w_b} to $\mathcal{A}$.
- Phase 2: $\mathcal{A}$ issues queries to the oracles same as in Phase 1.
- Guess: $\mathcal{A}$ outputs a bit $b_\mathcal{A}$ and wins the game if $b' = b$.

Let $Adv_\mathcal{A}^{W1} = \Pr[w_\mathcal{A} = w_c]$ denote the advantage of $\mathcal{A}$ to win Game W1.

Definition 6: A dDHPEKS scheme is IND-TW-CPA secure if for any PPT adversary $\mathcal{A}$, the advantage $Adv_\mathcal{A}^{W1}$ is negligible.

Game W2: Ciphertext indistinguishability against chosen keyword attacks:

- Setup: Same as that in Game W1.
- Phase 1: Same as that in Game W1.
- Challenge: $\mathcal{A}$ chooses an identity ID_c, two keywords w_0, w_1 and a plaintext M. The constraint is that ID_c cannot appear in $\mathcal{O}_E$ in Phase 1. $\mathcal{C}$ generates the node $\mathsf{Node}_{\mathrm{ID}_c}$ and randomly selects a bit $b \in \{0,1\}$. In the next step, $\mathcal{C}$ generates a ciphertext $C_{[w_b,M]}$. Finally, $\mathcal{C}$ returns $\mathsf{Node}_{\mathrm{ID}_c}$ and $C_{[w_b,M]}$ to $\mathcal{A}$.
- Phase 2: $\mathcal{A}$ still can issue queries to the oracle same as in phase 1 with the constraint that ID_c cannot appear in $\mathcal{O}_E$.
- Guess: $\mathcal{A}$ outputs a bit b' and wins the game if $b' = b$.

Let $Adv_\mathcal{A}^{W2} = |\Pr[b' = b] - \frac{1}{2}|$ denote the advantage of $\mathcal{A}$ to win Game W2.

Definition 7: A dDHPEKS scheme is IND-CW-CPA secure if for any PPT adversary $\mathcal{A}$, the advantage $Adv_\mathcal{A}^{W2}$ is negligible.

3) Plaintext Privacy against Chosen Ciphertext Attacks: The following game is to define the semantic security against chosen ciphertext attacks.

Game M: Ciphertext indistinguishability against chosen plaintext attacks:

- Setup: Same as that in Game W1.
- Phase 1: $\mathcal{A}$ can issue at most q_E queries to the Extract Oracle $\mathcal{O}_E$ below.
  - $\mathcal{O}_E(\mathrm{ID}_j)$: Given an identity ID_j, the oracle returns the corresponding node Node_j and secret key sk_j.
- Challenge: $\mathcal{A}$ chooses an identity ID_c, a keyword w and two plaintext M_0, M_1. The constraint is that ID_c cannot be submitted to $\mathcal{O}_E$ in Phase 1. $\mathcal{C}$ generates the node $\mathsf{Node}_{\mathrm{ID}_c}$ and randomly selects a bit $b \in \{0,1\}$. In the next step, $\mathcal{C}$ generates a ciphertext $C_{[w,M_b]}$. Finally, $\mathcal{C}$ returns $\mathsf{Node}_{\mathrm{ID}_c}$ and $C_{[w,M_b]}$ to $\mathcal{A}$.
- Phase 2: $\mathcal{A}$ issues queries to the oracle same as in Phase 1 with the constraints that ID_c cannot be submitted to $\mathcal{O}_E$ and $C_{[w,M_b]}$ cannot appear in $\mathcal{O}_D$.
- Guess: $\mathcal{A}$ outputs a bit b' and wins the game if $b' = b$.

Let $Adv_{\mathcal{A}}^{M} = |\Pr[b' = b] - \frac{1}{2}|$ denote the advantage of $\mathcal{A}$ to win Game M.

Definition 8: A dDHPEKS scheme is IND-CCA secure if for any PPT adversary $\mathcal{A}$, the advantage $Adv_{\mathcal{A}}^{M}$ is negligible.

B. Security Analysis of dDHPEKS Scheme

We prove our dDHPEKS scheme is secure under the above security model.

Theorem 1: Our proposed dDHPEKS scheme is secure against key collusion attacks if the CDH assumption holds.

The proof of theorem 1 is straightforward. The structure of our PKTree consists of multiple layers of public identity information. The corresponding secret keys of each layer is generated by their parent node, which is the same as the key extract process of the pairing based IBE schemes. Because the pairing based IBE scheme is secure against collusion attacks if the CDH assumption holds, our proposed dDHPEKS scheme enjoys security against key collusion attacks as well.

Theorem 2: Our dDHPEKS scheme is IND-TW-CPA secure if the DLIN assumption holds.

Proof 1: The challenger $\mathcal{C}$ is given a DLIN instance $\mathrm{Ins}_{\mathrm{DLIN}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \hat{e}, u_1, u_2, u_1^x, u_2^y, v, Z)$. To distinguish whether Z is equal to $v^{x+y} \in \mathbb{G}_2$ or a random element $v^r \in \mathbb{G}_2$, the challenger $\mathcal{C}$ plays the following game with the adversary $\mathcal{A}$ who tries to break the security of our dDHPEKS scheme.

Game W1:

- Setup: $\mathcal{C}$ randomly selects $\mathsf{msk} = k \in \mathbb{Z}_p$, computes the master public key $\mathsf{MPK} = u_2^k$ and builds the PKTree. Given an identity ID_i, $\mathcal{C}$ generates the corresponding node Node_i and secret key sk_i. Finally, $\mathcal{C}$ returns the global parameter $\mathcal{GP} = \{\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \hat{e}, g = u_2, h = v, \mathsf{MPK}, H, H_1, H_2, H_3, H_4, H_5\}$. Additionally, $\mathcal{C}$ returns the server's public key $\mathsf{pk}_s = u_1$ and the node Node_i.
- Phase 1: $\mathcal{A}$ can issue at most q_E, q_T, q_H and q_{H_1} queries to the following oracles, respectively.
  - Extract Oracle $\mathcal{O}_E$: Given an identity ID_j, it returns the corresponding result in records if ID_j has been asked before, otherwise, returns the corresponding results as follows:
  $$\begin{aligned}\mathsf{Node}_j &= (\mathsf{Pub}_j, \sigma_{i,j}) = ((\mathsf{pk}_j \| \overline{\mathbf{ID}}_j), \sigma_{i,j}) \\ &= ((g_1^{H_2(\mathsf{sk}_j)} \| \mathrm{ID}_j \| \varsigma_j), \mathsf{Sig}_{\mathsf{sk}_i}(\mathsf{Pub}_j)), \\ \mathsf{sk}_j &= H_1(\overline{\mathbf{ID}}_i)^{H_2(\mathsf{sk}_i)}.\end{aligned}$$
  - Trapdoor Oracle $\mathcal{O}_T$: Given an identity ID_j and a keyword w, this oracle randomly selects $t_1, t_2 \in \mathbb{Z}_p$ and returns $T_{w,\mathrm{ID}_j} = (T_1, T_2, T_3)$ as follows:
  $$\begin{aligned}T_1 &= h^{t_1+t_2} \cdot H_1(\overline{\mathbf{ID}}_j)^{H_2(\mathsf{sk}_j)\cdot H(w)}, \\ T_2 &= \mathsf{pk}_s^{t_1} = u_1^{t_1}, T_3 = g^{t_2}.\end{aligned}$$
- Challenge: $\mathcal{A}$ sends $\mathcal{C}$ two keywords w_0, w_1 and a node Node_c, parsed as $(\mathsf{pk}_c \| \overline{\mathbf{ID}}_c, \sigma_{i,c})$. $\mathcal{C}$ reveals the secret key sk_c of Node_c and returns a trapdoor $T_{w_b} = (T_1^*, T_2^*, T_3^*)$, where b is a randomly chosen bit to decide which keyword is encrypted:
$$\begin{aligned}T_1^* &= Z \cdot H_1(\overline{\mathbf{ID}}_c)^{H_2(\mathsf{sk}_c)\cdot H(w_b)}, \\ T_2^* &= u_1^x, T_3^* = u_2^y.\end{aligned}$$
- Phase 2: $\mathcal{A}$ issues queries as in Phase 1.
- Guess: $\mathcal{A}$ returns a bit b' and wins the game if $b' = b$.

$Adv_{\mathcal{A}}^{W1} = \Pr[b' = b] - \frac{1}{2} = \epsilon$ denotes the advantage of $\mathcal{A}$'s winning the game. When $\mathcal{A}$ wins the game, $\mathcal{C}$ returns $Z = v^{x+y}$ to DLIN challenger, otherwise, Z is equal to a random element $R \xleftarrow{\$} \mathbb{G}_2$. $\mathcal{C}$ will break the DLIN assumption with the advantage $\epsilon_{\mathcal{C}} = \epsilon$. Hence, $Adv_{\mathcal{A}}^{W1}$ is negligible if the DLIN assumption holds.

Theorem 3: Our proposed dDHPEKS scheme is IND-CW-CPA secure if the DBDH assumption holds.

Proof 2: The challenger $\mathcal{C}$ is given a DBDH instance $\mathrm{Ins}_{\mathrm{DBDH}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \hat{e}, g, g^x, g^y, h, h^y, h^z, Z)$. To distinguish whether Z is equal to $\hat{e}(g,h)^{xyz}$ or Z is a randomly chosen element in $\mathbb{G}_T$, the challenger $\mathcal{C}$ plays the following game with the adversary $\mathcal{A}$ who tries to break the security of our dDHPEKS scheme.

Game W2:

- Setup: $\mathcal{C}$ initializes the system by giving the global parameter $\mathcal{GP} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \hat{e}, g, h, \mathsf{MPK}, H, H_2, H_3, H_4, H_5)$.
- Phase 1: $\mathcal{A}$ can issue at most q_H, q_E and q_T queries to the hash oracle $\mathcal{O}_{H_1}$, extract oracle $\mathcal{O}_E$ and the trapdoor oracle $\mathcal{O}_T$, respectively.
  - Random Oracle $\mathcal{O}_{H_1}$: Given some public information $\overline{\mathbf{ID}}_j$, it randomly selects $r_j \in \mathbb{Z}_p$ and returns the hash value $H_1(\overline{\mathbf{ID}}_j) = (h^z)^{r_j}$.
  - Extract Oracle $\mathcal{O}_E$: Same as that in Game W1.
  - Trapdoor Oracle $\mathcal{O}_T$: Same as that in Game W1.
- Challenge: $\mathcal{A}$ sends $\mathcal{C}$ a new identity ID_c, which did not appear in $\mathcal{O}_E$, two keywords w_0, w_1 and a plaintext M. $\mathcal{C}$ adds a child node Node_c corresponding with this identity into the tree and generates a trapdoor $C_{w_b,M}$, where the the random bit b decides which keyword is encrypted in this ciphertext. Finally, $\mathcal{C}$ randomly selects $s \in \mathbb{Z}_p$

and returns the node Node_c and the Ciphertext $C_{w_b,M} = (C_1^*, \cdots, C_7^*)$ to $\mathcal{A}$:

$$\begin{aligned}\mathsf{Node}_c &= (\mathsf{Pub}_c, \sigma_{i,c}) = ((\mathsf{pk}_c, \overline{\mathbf{ID}}_c), \sigma_{i,c})\\ &= ((g^x, (h^z)^{r_c}), \mathsf{Sig}_{\mathsf{sk}_i}(\mathsf{Pub}_c)),\\ C_1^* &= Z^{r_c \cdot H(w)},\ C_2^* = g^y,\ C_3 = h^y,\\ C_4^* &= K \oplus H_3(g^{x\cdot s}),\ C_5^* = g^s,\ C_7^* = \mathsf{AESEnc}_K(M),\\ C_6^* &= H_4(C_1, \cdots, C_5, H_5(C_7))^s,\end{aligned}$$

where $\overline{\mathbf{ID}}_c = (h^z)^{r_c}$ and r_c is recalled from records of the hash oracle $\mathcal{O}_{H_1}$.

- Phase 2: $\mathcal{A}$ still can issue queries to the oracles same as in phase 1 except that the tuples $\langle \mathrm{ID}_c, w_0\rangle$ and $\langle \mathrm{ID}_c, w_1\rangle$ cannot appears in the trapdoor oracle $\mathcal{O}_T$.
- Guess: $\mathcal{A}$ returns a bit b' and wins the game if $b' = b$.

Assume the advantage $Adv_{\mathcal{A}}^{W2} = \epsilon$ of $\mathcal{A}$' winning the game is non-negligible, $\mathcal{C}$ can break the DBDH assumption with the non-negligible advantage $\epsilon_{\mathcal{C}} = \epsilon$. Hence, $Adv_{\mathcal{A}}^{W2}$ is negligible if the DBDH assumption holds.

Theorem 4: Our proposed dDHPEKS scheme is IND-CPA secure if AES encryption is IND-CPA secure and the CDH assumption holds.

Proof 3: The dDHPEKS scheme leverages the AES to encrypt the plaintext M and hides the session key K into C_4. Hence, if C_4 does not leak any information about the encryption key K, security of our dDHPEKS will be based on that of AES. The following game is played between a PPT adversary $\mathcal{A}$ and the challenger $\mathcal{C}$. Given a CDH instance $\mathrm{Ins}_{\mathrm{CDH}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \hat{e}, g, g^x, g^y)$, $\mathcal{C}$ works as follows.

Game M:

- Setup: $\mathcal{C}$ initializes the system by giving the global parameter $\mathcal{GP} = (\mathbb{G}_1,\ \mathbb{G}_2,\ \mathbb{G}_T,\ p,\ \hat{e},\ g,\ h,\ \mathsf{MPK},\ H,\ H_1,\ H_2,\ H_4)$.
- Phase 1: $\mathcal{A}$ can issue at most q_H and q_E queries to the hash oracles $\mathcal{O}_{H_3}, \mathcal{O}_{H_5}$ and extract oracle $\mathcal{O}_E$, respectively.
  - Random Oracle $\mathcal{O}_{H_3}$: Given an element $g^* \in \mathbb{G}_1$, it returns an ℓ_1-bit random number h^* as the hash value $H_3(g^*)$.
  - Random Oracle $\mathcal{O}_{H_5}$: Given an input T, it randomly selects $r_T \in \mathbb{Z}_p$ and returns $H_5(T) = (g^y)^{r_T}$.
  - Extract Oracle $\mathcal{O}_E$: Same as that in Game W1.
- Challenge: $\mathcal{A}$ sends $\mathcal{C}$ a new identity ID_c, a keyword w and two plaintexts M_0, M_1. $\mathcal{C}$ adds a child node Node_c corresponding with this identity into the tree and generates a trapdoor C_{w,M_b}, where the the random bit b decides which plaintext is encrypted in this ciphertext. Finally, $\mathcal{C}$ returns the node Node_c and the Ciphertext $C_{w,M_b} = (C_1^*, \cdots, C_7^*)$ to $\mathcal{A}$:

$$\begin{aligned}\mathsf{Node}_c &= (\mathsf{Pub}_c, \sigma_{i,c}) = ((\mathsf{pk}_c, \overline{\mathbf{ID}}_c), \sigma_{i,c})\\ &= ((g^x, \overline{\mathbf{ID}}_c), \mathsf{Sig}_{\mathsf{sk}_i}(\mathsf{Pub}_c)),\\ C_1^* &= \hat{e}(\mathsf{pk}_c^{r\cdot H(w)}, H_1(\overline{\mathbf{ID}}_c)),\ C_2^* = g^r,\ C_3^* = h^r,\\ C_4^* &= K \oplus h_c^*,\ C_5^* = g^{y\cdot s},\ C_7^* = \mathsf{AESEnc}_K(M),\\ C_6^* &= H_4(C_1, C_2, C_3, C_4, C_5, H_5(C_7))^s,\end{aligned}$$

where r, s are randomly chosen from $\mathbb{Z}_p$ and h_c^* is an ℓ_1-bit random number.

- Phase 2: $\mathcal{A}$ still can issue queries to the oracles same as in phase 1 except that the ciphertext C_{w,M_b} cannot appears in the decrypt oracle $\mathcal{O}_D$.
- Guess: $\mathcal{A}$ returns a bit b' and wins the game if $b' = b$.

We define an event, denoted by $\mathbf{E}$, that $\mathcal{A}$ issues $g^* = g^{xy\cdot s}$ to the hash oracle $\mathcal{O}_{H_3}$. In case $\mathbf{E}$ happens, the challenger $\mathcal{C}$ solves the CDH problem via computing $g^{xy} = (g^*)^{\frac{1}{s}}$. If the CDH assumption holds, $\mathbf{E}$ happens with a negligible probability. In another case, $\mathbf{E}$ does not happen, the ciphertext C_4 is random in $\mathcal{A}$'s view and the session key K can be revealed with a negligible probability. Hence, in this game, $\mathcal{A}$'s winning advantage $Adv_{\mathcal{A}}^{M}$ is equal to or less than a negligible probability if AES encryption is IND-CPA secure and the CDH assumption holds.

VI. Comparison and Experiments

To evaluate the running efficiency of the proposed dDHPEKS scheme, we compare it with previous searchable encryption schemes [7, 30, 31]. Table I shows that our proposal has the comparable running efficiency with the compared schemes. To achieve higher level security and more functionality, it is reasonable to sacrifices certain running efficiency. Among the four schemes, only our dDHPEKS scheme can not only resist outside keyword guessing attacks but also achieve integration of PKE and PEKS. Moreover, based on the PKTree, our proposal supports node supervision, which cannot be achieved in previous PEKS schemes.

We implemented our dDHPEKS scheme and the compared schemes denoted by BDOP-PEKS [7], BSS-PKE/PEKS [30] and HL-PEKS [31], respectively, utilizing the C language and PBC library [2] in Ubuntu OS 19.04 on a laptop with a 2.3 GHz Intel Core i5 CPU and 8 GB 2133 MHz LPDDR3 memory. A Type-A pairing was chosen and used to initialize the system, which owns the same security level as a 1024-bit RSA encryption. To apply the keyword search to some dataset, the first step is to extract keywords. Each record in the dataset should be associated with a keyword. The keyword will be encrypted with the proposed encryption scheme and the record would be encrypted with some other encryption scheme, such as AES. Inverted index is usually used to support efficient encrypted data search. Basically, inverted index contains a list of (encrypted) keywords, and each keyword is associated with a queue of records which contains the keyword. In this system, the keywords can be found in a common dictionary with a high probability and different bit-length of keywords theoretically do not affect the efficiency of the program, because each keyword will be hashed into a fixed-length string before encryption. Hence, we choose the Oxford dictionary as the keyword space in the implementation and all the words in it are taken as input of the related algorithms of the scheme.

As shown in Figure 6, compared with other three schemes, the encryption algorithm of our dDHPEKS scheme is slower. The reason is that nearly half of the computing overhead is to encrypt the session key K, which does not exist in BDOP-PEKS nor HL-PEKS schemes.

TABLE I
COMPARISONS WITH PEKS SCHEMES IN LITERATURE

Scheme	Encrypt	Trapdoor	Test	KGA	INT	INT/L	HMSE
BDOP-PEKS[7]	$2\text{Exp}_{\mathbb{G}_1}$ + 2Hash + 1Pairing	$1\text{Exp}_{\mathbb{G}_1}$ + 1Hash	1Hash + 1Pairing	No	No	No	No
BSS-PKE/PEKS[30]	$3\text{Exp}_{\mathbb{G}_1}$ + 4Hash + 1Pairing	$1\text{Exp}_{\mathbb{G}_1}$ + 1Hash	1Hash + 1Pairing	No	Yes	No	No
HL-PEKS[31]	$2\text{Exp}_{\mathbb{G}_1}$ + 2Hash + 3Pairing	$3\text{Exp}_{\mathbb{G}_1}$ + 1Hash	$1\text{Exp}_{\mathbb{G}_1}$ + 1Hash + 1Pairing	Yes	No	No	No
Our dDHPEKS	$4\text{Exp}_{\mathbb{G}_1}$ +$2\text{Exp}_{\mathbb{G}_2}$ + 5Hash + 1Pairing + 1AES	$1\text{Exp}_{\mathbb{G}_1}$ + $3\text{Exp}_{\mathbb{G}_2}$ + 3Hash	$1\text{Exp}_{\mathbb{G}_2}$ + 2Pairing	Yes	Yes	Yes	Yes

KGA: The schemes can resist outside keyword guessing attacks;
INT: The schemes integrate the encrypted keyword and plaintext;
INT/L: The schemes are adaptable of any length of the integrated plaintexts;
HMSE: The schemes support hierarchical multi-user keyword search.

Notice that the overhead of AES encryption algorithm of our dDHPEKS scheme was not included in the experiment results, because it is indeterminate and affected by the length of the plaintext.

Figure 7 shows that our dDHPEKS scheme owns an efficient trapdoor algorithm, which is similar to the BDOP-PEKS and the BSS-PKE/PEKS schemes and more efficient than the HL-PEKS scheme.

The overhead of test algorithms of the four schemes are shown in Figure 8. To search a keyword over 1,000 ciphertexts, the average overhead of using the BDOP-PEKS and the BSS-PKE/PEKS schemes are nearly 0.75 second, the HL-PEKS scheme is 2.00 seconds and our dDHPEKS scheme is about 2.85 seconds. Even though the former two schemes are faster, they cannot resist the outside offline keyword guessing attacks. With the same security level, the searching efficiency of our dDHPEKS scheme is comparable with HL-PEKS.

Consider the scenario in which the sender sends an encrypted keyword to multiple receivers in a hierarchy, i.e. receiver R_2 is the child-node of R_1, receiver R_3 is the child-node of R_2, and so on. If we use a traditional PEKS scheme, the sender has to encrypt the keyword multiple times (or use a re-encryption mechanism to re-encrypt an existing ciphertext to new ciphertexts under other receivers' public keys), and sends ciphertexts to the respective receivers. Hence the encryption time she has to spend is linear in the number of receivers. However, if we use the proposed dDHPEKS scheme to implement the scenario, the sender needs only to encrypt the keyword for the receiver at the lowest level among all the receivers, and then all the receivers would be able to test the encrypted keyword, since a parent node in our scheme has the privilege of accessing its child-node's encrypted data. Hence, the encryption time that the sender needs to spend in this case is constant, e.g. independent of the number of receivers. Figure 9 shows the encryption time comparison in this scenario.

In order to search over ciphertexts for a receiver at a low level (say, level 5), a receiver at a high level (say, level 1) needs to generate secret key (and the trapdoor) for each node along the path from the high level receiver to the low level receiver, and the computational cost is thus linearly with the number of levels between the two receivers. Our implementation demonstrates that the additional running cost is much smaller than the encryption time. Figure 10 shows that the time of generating the additional secret keys and trapdoors are both close to 0.13s in case there are 100 levels between the two receiver nodes, which is much smaller than the 55s$\sim$70s encryption time in the compared schemes. Consider that there are usually not many levels in an enterprise/organization, e.g. less than 10 levels, the cost of generating the trapdoor could thus be neglected.

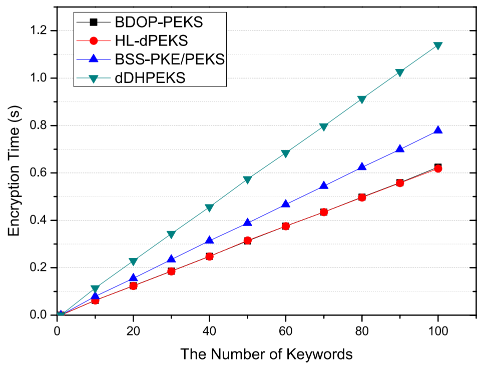

Fig. 6. Comparison of Encryption Algorithms

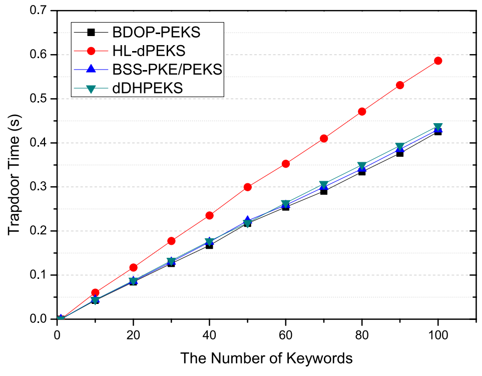

Fig. 7. Comparison of Trapdoor Algorithms

VII. CONCLUSION

In this paper, we proposed a new protocol named hierarchical public key encryption with keyword search (HPEKS), which supports a user to monitor its child users. We introduced a public key tree (PKTree) structure and used it to build a semi-generic HPEKS construction. We also proposed an advanced HPEKS scheme dDHPEKS, which integrates PEKS

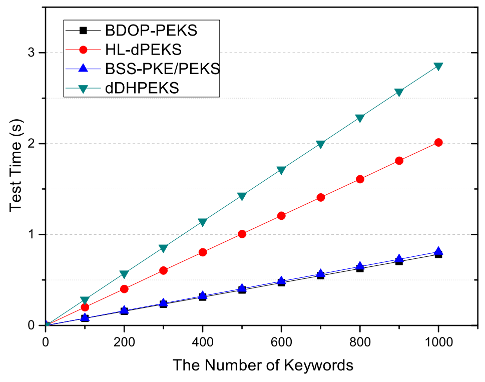

Fig. 8. Comparison of Test Algorithms

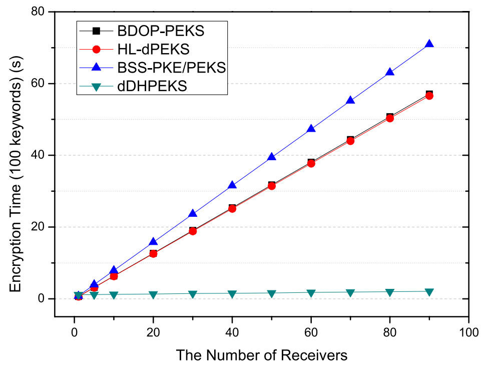

Fig. 9. Encryption Time along with Receiver Number Increasing

and PKE, and can resist outside offline keyword guessing attacks. Experiments show that our dDHPEKS scheme has comparable efficiency with existing related PEKS schemes.

References

[1] W. Kim, "Cloud computing trends: 2018 state of the cloud survey," *URL https://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2018-state-cloud-survey, [Online]*, 2018.

[2] B. Lynn and *et al*, "Pairing-based cryptography library," https://crypto.stanford.edu/pbc/, 2013.

[3] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceedings of IEEE Symposium on Security and Privacy. S&P 2000*, 2000, pp. 44–55.

[4] Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *International Conference on Applied Cryptography and Network Security*. Springer, 2005, pp. 442–455.

[5] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," *Journal of Computer Security*, vol. 19, no. 5, pp. 895–934, 2011.

[6] K. Kurosawa and Y. Ohtaki, "Uc-secure searchable symmetric encryption," in *International Conference on Financial Cryptography and Data Security*. Springer, 2012, pp. 285–298.

[7] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, *Public Key Encryption with Keyword Search*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 506–522.

[8] D. J. Park, K. Kim, and P. J. Lee, "Public key encryption with conjunctive field keyword search," in *International Workshop on Information Security Applications*. Springer, 2004, pp. 73–86.

[9] P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data." in *Applied Cryptography and Network Security, Second International Conference, ACNS 2004, Yellow Mountain, China, June 8-11, 2004, Proceedings*, 2004, pp. 31–45.

[10] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Transactions on parallel and distributed systems*, vol. 25, no. 1, pp. 222–233, Jan 2014.

[11] Z. Fu, X. Sun, Z. Xia, L. Zhou, and J. Shu, "Multi-keyword ranked search supporting synonym query over encrypted data in cloud computing," in *Performance Computing and Communications Conference (IPCCC), 2013 IEEE 32nd International*. IEEE, 2013, pp. 1–8.

[12] H. Li, D. Liu, Y. Dai, T. H. Luan, and X. S. Shen, "Enabling efficient multi-keyword ranked search over encrypted mobile cloud data through blind storage," *IEEE Transactions on Emerging Topics in Computing*, vol. 3, no. 1, pp. 127–138, 2015.

[13] F. Bao, R. H. Deng, X. Ding, and Y. Yang, "Private query on encrypted data in multi-user settings," in *International Conference on Information Security Practice and Experience*. Springer, 2008, pp. 71–85.

[14] F. Zhao, T. Nishide, and K. Sakurai, "Multi-user keyword search scheme for secure data sharing with fine-grained access control," in *International Conference on Information Security and Cryptology*. Springer, 2011, pp. 406–418.

[15] Y. Yang, H. Lu, and J. Weng, "Multi-user private keyword search for cloud computing," in *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*. IEEE, 2011, pp. 264–271.

[16] C. Van Rompay, R. Molva, and M. Önen, "Multi-user searchable encryption in the cloud," in *International*

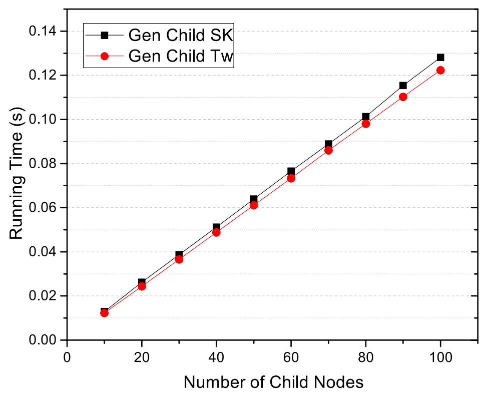

Fig. 10. Time of Generating Child-node's Secret Keys and Trapdoors

Information Security Conference. Springer, 2015, pp. 299–316.

[17] M. Hattori, T. Hirano, T. Ito, N. Matsuda, T. Mori, Y. Sakai, and K. Ohta, "Ciphertext-policy delegatable hidden vector encryption and its application to searchable encryption in multi-user setting," in *IMA International Conference on Cryptography and Coding*. Springer, 2011, pp. 190–209.

[18] J. W. Byun, H. S. Rhee, H.-A. Park, and D. H. Lee, "Off-line keyword guessing attacks on recent keyword search schemes over encrypted data," in *Workshop on Secure Data Management*. Springer, 2006, pp. 75–83.

[19] W.-C. Yau, S.-H. Heng, and B.-M. Goi, "Off-line keyword guessing attacks on recent public key encryption with keyword search schemes," in *International Conference on Autonomic and Trusted Computing*. Springer, 2008, pp. 100–105.

[20] L. Fang, W. Susilo, C. Ge, and J. Wang, "A secure channel free public key encryption with keyword search scheme without random oracle," in *Cryptology and Network Security, International Conference, CANS 2009, Kanazawa, Japan, December 12-14, 2009. Proceedings*, 2009, pp. 248–258.

[21] Q. Huang and H. Li, "An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks," *Information Sciences*, vol. 403-404, pp. 1 – 14, 2017.

[22] ——, "An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks," Cryptology ePrint Archive: Report 2018/007, 2018.

[23] D. He, M. Ma, S. Zeadally, N. Kumar, and K. Liang, "Certificateless public key authenticated encryption with keyword search for industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3618–3627, 2017.

[24] H. Li, Q. Huang, J. Shen, G. Yang, and W. Susilo, "Designated-server identity-based authenticated encryption with keyword search for encrypted emails," *Information Sciences*, vol. 481, pp. 330 – 343, 2019.

[25] R. Chen, Y. Mu, G. Yang, F. Guo, and X. Wang, "Dual-server public-key encryption with keyword search for secure cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 4, pp. 789–798, April 2016.

[26] R. Chen, Y. Mu, G. Yang, F. Guo, X. Huang, X. Wang, and Y. Wang, "Server-aided public key encryption with keyword search," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 12, pp. 2833–2842, Dec 2016.

[27] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Annual International Cryptology Conference*. Springer, 2001, pp. 213–229.

[28] J. Katz and Y. Lindell, *Introduction to modern cryptography, Second Edition*. CRC press, 2014.

[29] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Annual International Cryptology Conference*. Springer, 2004, pp. 41–55.

[30] J. Baek, R. Safavi-Naini, and W. Susilo, *On the Integration of Public Key Data Encryption and Public Key Encryption with Keyword Search*. Springer Berlin Heidelberg, 2006.

[31] C. Hu and P. Liu, "A secure searchable public key encryption scheme with a designated tester against keyword guessing attacks and its extension," in *International Conference on Computer Science, Environment, Ecoinformatics, and Education*. Springer, 2011, pp. 131–136.

Hongbo Li received his B.S., M.S. and PhD degree from South China Agricultural University, and now is a postdoc student at College of Mathematics and Informatics, South China Agricultural University, Guangzhou, China. His research interests include applied cryptography and cloud security.

Qiong Huang received his Ph.D. degree from City University of Hong Kong in 2010. He is now a professor with College of Mathematics and Informatics, South China Agricultural University, Guangzhou, China. His research interests include cryptography and information security, in particular, cryptographic protocols design and analysis. He has published more than 100 research papers in international conferences and journals, and served as a programme committee member in many international conferences.

Willy Susilo (SM'01) received the Ph.D. degree in computer science from the University of Wollongong, Australia. He is a Professor and the Head of School of Computing and Information Technology and the Director of Institute of Cybersecurity and Cryptology with the University of Wollongong. He has authored or co-authored over 300 research papers in the area of cybersecurity and cryptology. His main research interests include cybersecurity, cryptography, and information security. His work has been cited over 13,000 times in Google Scholar. He has served as a Program Committee Member in dozens of international conferences. He was previously awarded the prestigious ARC Future Fellow by the Australian Research Council and the Researcher of the Year award in 2016 by the University of Wollongong. He is Editor-in-Chief of the Information journal. He is currently serving as an Associate Editor for several international journals, including IEEE Transactions in Dependable and Secure Computing, Elsevier Computer Standards and Interface and the International Journal of Information Security (Springer).