Dual-Server Public-Key Authenticated Encryption With Keyword Search

Biwen Chen, Libing Wu, Sherali Zeadally, and Debiao He

***Abstract*—In cloud storage, how to search sensitive data efficiently and securely is a challenging problem. The searchable encryption technique provides a secure storage method without loss of data confidentiality and usability. As an important branch of searchable encryption, public-key encryption with keyword search (PEKS) is widely studied by scholars. However, most of the traditional PEKS schemes are vulnerable to the inside keyword guessing attack (IKGA). Resisting the inside keyword guessing attack is likely to become an essential property of all new PEKS schemes. For a long time, mitigating IKGA has been inefficient and difficult, and most existing PEKS schemes fail in achieving their security goals. To address the above problems, we define the notion of Dual-server Public-key Authenticated Encryption with Keyword Search (DPAEKS), which protects against IKGA by leveraging two servers that do not cooperate, and supports the authentication property. Then, we provide a construction of DPAEKS without bilinear pairings. Experimental results obtained using a real-world dataset show that our scheme is highly efficient and provides strong security, making it suitable for deployment in practical applications.**

***Keywords*—Cloud storage, public-key encryption with keyword search, dual-server, authorization, inside keyword guessing attacks.**

I. INTRODUCTION

Cloud storage has become a promising paradigm with the explosive growth of data in recent years. It not only provides an on-demand storage service for users, but also facilitates users' access to data. However, data outsourced to cloud server may contain some sensitive information (e.g., company financial data, health records), which may incur security and privacy issues. To protect data confidentiality, one general approach is to encrypt the data before transferring it to the cloud server. But the encrypted data makes its utilization more difficult, particularly the ability of data retrieval.

To implement the searchable feature of encrypted data, Song *et al.* [1] were the first to propose the notion of searchable encryption (SE) based on the symmetric crypto-system. Subsequently, to avoid the key management and distribution, Boneh *et al.* [2] introduced the notion of public-key encryption with keyword search (PEKS) and constructed a concrete scheme based on the asymmetric crypto-system. Under the PEKS framework, there are three entities namely, the data owner, the data receiver (user) and the cloud server. Using the public key of the data receiver, the data owner encrypts the files and each keyword which is extracted from these files, and then uploads the ciphertexts to the cloud server. The data user sends a trapdoor containing the keyword which he/she wants to search to the cloud server. The cloud server tests whether the keyword in corresponding to the trapdoor is equal to the keyword underlying the ciphertext. The cloud server returns the encrypted data corresponding to the trapdoor. Fig.1 describes the process.

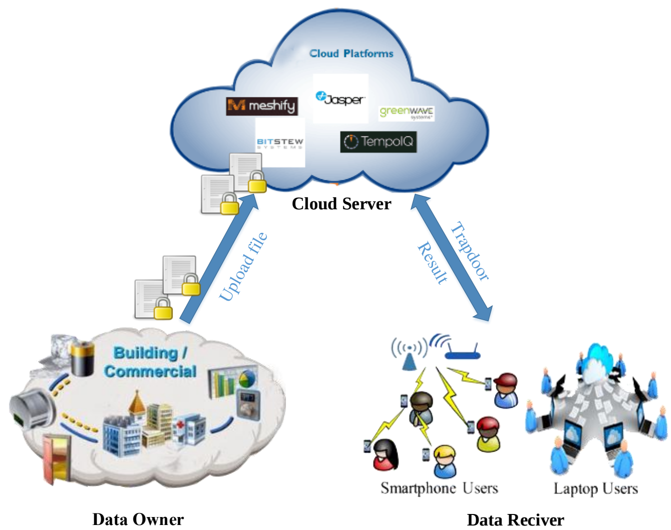

Fig. 1. General framework of PEKS

Unfortunately, despite the benefits of PEKS schemes based on Boneh's framework [2], most of them do not resist the inside keyword guessing attack (IKGA). The IKGA is usually launched by the malicious cloud server and may reveal the receiver's privacy (e.g. interest, identity). The IKGA is possible because the malicious cloud server can obtain the trapdoor, generate the ciphertext of any keyword and execute the test algorithm of PEKS multiple times. Another reason is that the keywords space is usually not large in real applications. Therefore, the adversary can repeat the above procedure until (s)he finds the correct keyword. Security designs against the

B. Chen is with Computer School,Wuhan University, Wuhan, China
E-mail: macrochen@whu.edu.cn

L. Wu is with Computer School,Wuhan University, Wuhan, China
E-mail: whuwlb@126.com

S. Zeadally is with the College of Communication and Information at the University of Kentucky, USA.
E-mail: szeadally@uky.edu

D. He is with Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan, China
E-mail: hedebiao@163.com

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2945714, IEEE Transactions on Cloud Computing

IKGA are urgently needed to facilitate the application of searchable encryption.

Recently, some new works [3]–[5] of PEKS have been proposed against the IKGA. In Xu *et al*'s work [3], there are two trapdoors for searching data: a fuzzy trapdoor and an accurate trapdoor. Because the cloud server returns ambiguous results, the adversary cannot obtain accurate information about keywords. Nevertheless, the adversary still has access to relevant information and the communication complexity of Xu's scheme is very high. Chen *et al.* [4] proposed a new framework for secure PEKS to resist IKGA and has low computation costs. However, there is still a significant deficiency in their design that their scheme does not support the authentication [6], which means any adversary can generate a legal trapdoor using public information if no other authentication mechanism is introduced. One noteworthy work introduced by Huang *et al.* [5] is secure against IKGA without using multiple servers [7]. Unfortunately, this scheme fails in achieving ciphertext indistinguishability, which means the adversary can successfully distinguish the ciphertexts corresponding to different keywords. Therefore, how to construct a PEKS scheme that can provide strong security, resist the IKGA and be efficient, remains an open problem.

A. Our Contributions

In this paper, our investigation started with two questions: 1) How to defend against the inside KGA in a public-key searchable encryption scheme? 2) How to avoid computation-intensive operations in the public-key infrastructure subjected to security constraints? We provide a positive answer to these two questions by constructing a **D**ual-server **P**ublic-key **A**uthenticated **E**ncryption with **K**eyword **S**earch (DPAEKS). The proposed scheme can prevent the malicious server from guessing the keywords which are queried by the data user. On the other hand, our scheme can efficiently protect the data receiver's privacy. We summarize the contributions of this work as follows:

- We define the notion of dual-server public-key authenticated encryption with keyword search. The design of DPAEKS follows the dual-server method used by Chen *et al.* [4], but with a different idea. In work [4], the test functionality is split into two parts for two independent cloud servers. The public keys of both cloud servers are used to generate the ciphertexts and trapdoors. The idea in DPAEKS is inspired by the classical Diffie-Hellman Exchange algorithm. Both ciphertext generation and trapdoor generation for each keyword require not only the public keys of two servers, but also the shared key between the data owner and the data user. This ensures that only authenticated users can search the ciphertexts.
- We present a concrete construction of DPAEKS without bilinear pairing operation, and prove its security under the decisional Diffie-Hellman (DDH) assumption.
- We implement our scheme and compare its performance with previous works, and evaluate it on a real-world dataset. The experimental results demonstrate the practicability of DPAEKS over searching encrypted data.

B. Organization

The rest of the paper is organized as follows. The related works are provided in Section II. Section III presents the notations used in the rest of the paper and introduces some concepts, including the security assumptions, the system model, the formal definition of DPAEKS. We present a concrete construction of DPAEKS in Section IV. The performance analysis of DPAEKS is presented in Section V. Finally, we make some concluding remarks in Section VI.

II. Related Works

This paper addresses the topic on how to defend against the inside keyword guessing attacks and improve the efficiency of the PEKS scheme. For simplicity, we have only reviewed one area of searchable encryption namely, public-key encryption with keyword search.

Boneh *et al.* [2] proposed the first public-key encryption with keyword search (PEKS) scheme which consists of four polynomial time randomized algorithms. Their scheme enables a data user to search ciphertext data in the asymmetric encryption setting. Then, PEKS has attracted the attention of many researchers. Subsequently, Baek *et al.* [8] presented a new scheme based-on Boneh's framework as a secure channel-free PEKS (SCF-PEKS). Their scheme eliminates the need for a secure channel by adding a key pair for the cloud server. Then, Rhee *et al.* [9] and Emura *et al.* [10] improved the security model of SCF-PEKS in different ways, respectively.

In addition, because the keyword space is usually not large in real applications, Byun *et al.* [11] found that the traditional PEKS may be vulnerable to the off-line keyword guessing attack [12], which can reveal the keyword of a trapdoor by the data receiver. To protect data privacy, Rhee *et al.* [13] defined the concept of trapdoor indistinguishability, which means that the trapdoor generation function is probabilistic and the trapdoors would be different even though they contain the same keyword. Subsequently, other works focused on trapdoor privacy [14]–[16]. Fang *et al.* [14] presented two important security notions, but their scheme has been shown to fail in achieving their security goal in recent work [17]. The scheme introduced by Guo *et al.* [15] requires less computation overhead as well as shorter ciphertext length as compared to the scheme of [14] and the work [16] protects trapdoor privacy by using an identity-based encryption with key unlinkability. But the above schemes are vulnerable to the inside KGA and Jeong *et al.* [18] pointed out that it is difficult to construct a secure PEKS scheme for resisting the IKGA under the original framework.

To address the above drawbacks, Xu *et al.* [3] introduced the concept of public-key encryption with fuzzy keyword search (PEFKS). The malicious server cannot learn the exact keyword by a fuzzy trapdoor since multiple keywords may share the same fuzzy trapdoor. Huang *et al.* [5] also proposed an efficient PEKS against IKGA. The data owner is given a key pair to prevent the inside malicious server from guessing the keyword in their scheme. But if the keyword is not changed, the generated trapdoor will be same. This means that the malicious cloud server would know the statistic knowledge of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2945714, IEEE Transactions on Cloud Computing

the trapdoors and learn the keywords corresponding to them. Besides, the schemes of Chen *et al.* [4], [19], [20] use two cloud servers to prevent IKGA, where the two servers do not collude with each other. Unfortunately, the works [4] and [19] are insecure wherein the adversary can generate a valid trapdoor of a keyword to search for encrypted data, the main reason is that their schemes lack the authorization property [6], [21]. Nevertheless, the work [20] requires a trusted third party to help users generate the pre-processed keyword. Besides, in work [22], a new PEKS scheme is proposed to resist the KGA by leveraging the authorization tokens. But the scheme requires an additional interaction between the data owner and the data user.

In addition, unlike symmetric searchable encryption (SSE), most PEKS schemes require some computation-intensive operations (e.g., map-to-point hash, bilinear pairing) [2], [4], [5], [7]–[9], [19] [11] [20], [23]–[25], which will become a major performance bottleneck in practical applications [26]. To reduce the computational complexity, the work [27] introduced a PEKS scheme based on quadratic residues based IBE scheme, which does not use bilinear pairing operation, but the protocol is still not practical [19]. Recently, Xu *et al.* [28] constructed a lightweight searchable public-key encryption by utilizing a hidden structure [29] to reduces the number of computation-intensive operations. Therefore, all new PEKS schemes should avoid implementing too many computation-intensive operations as much as possible.

III. PRELIMINARIES

A. Notations

Table I summarizes the notations used throughout this paper.

TABLE I. NOTATIONS

NOTATION	DESCRIPTION
λ	Security parameter
W	The universe of keywords
$\mathbb{G}_1$	Elliptic curve group $\mathbb{G}_1$
$\lvert\mathbb{G}_1\rvert, \lvert\mathbb{G}_2\rvert$	The binary sizes of groups $\mathbb{G}_1, \mathbb{G}_2$
q	The prime order of $\mathbb{G}_1$
P_1, P_2, P_3	Three different generators of $\mathbb{G}_1$
h_1	Collision resistant hash function
DO	Data Owner
DR	Data Receiver
AS	Assistant server
TS	Test server
ICT	Intermediate ciphertext
(PK_{as}, SK_{as})	Public/secret key pair of AS
(PK_{ts}, SK_{ts})	Public/secret key pair of TS
(PK_{dr}, SK_{dr})	Public/secret key pair of DR
(PK_{do}, SK_{do})	Public/secret key pair of DO
$s \xleftarrow{R} \mathcal{S}$	Random element s from the set $\mathcal{S}$
$C_{w_i} = (C_{1w_i}, C_{2w_i}, C_{3w_i})$	Ciphertext of $w_i \in W$
$T_{w_i} = (T_{1w_i}, T_{2w_i}, T_{3w_i})$	Trapdoor of $w_i \in W$
$ICT_{w_iw_j} = (ICT^1_{w_iw_j}, ICT^2_{w_iw_j})$	Intermediate ciphertext of the keywords
$negl(\lambda)$	Negligible probability

B. Assumptions

Decisional Diffie-Hellman Assumption (DDH). Given a quad $(P, xP, yP, zP) \in \mathbb{G}_1$, where $(x, y, z) \xleftarrow{R} \mathbb{Z}_q^*$ and $P \in \mathbb{G}_1$ is a generator. For any probabilistic polynomial time adversary A and a security parameter λ, the advantage $Adv_A^{DDH}(\lambda)$ of A is defined as:

$$\begin{aligned} Adv_A^{DDH}(\lambda) = & \lvert Pr[A(P, xP, yP, xyP) = 1]\rvert \\ & - \lvert Pr[A(P, xP, yP, zP) = 1]\rvert \end{aligned}$$

We say that DDH assumption holds if $Adv_A^{DDH}(\lambda)$ is negligible.

Diffie-Hellman Key Exchange. It can be devoted to establishing a session key between two communicating clients. The general process is described below:

- Setup: the client **A** and the client **B** determine together on a finite group $\mathbb{G}$ of order q and a generator $P \in \mathbb{G}$.
- Client **A**: it selects a random number $r_a \in \mathbb{Z}_q^*$ and publishes the element $P_a = r_aP \in \mathbb{G}_1$.
- Client **B**: it selects a random number $r_b \in \mathbb{Z}_q^*$ and publishes the element $P_b = r_bP \in \mathbb{G}_1$.
- Client **A**, Client **B**: they compute the session key $k = r_aP_b = r_bP_a = r_ar_bP \in \mathbb{G}_1$.

C. System Model

In the single-server framework, a centralized server is not only responsible for storing the encrypted data, but also for testing whether the submitted trapdoor and the stored ciphertext contain the same keyword. The centralized server can try each possible keyword, generates the ciphertext of it, and test the ciphertext with the given trapdoor. If the test succeeds, the centralized server determines which keyword is encapsulated in the given trapdoor. To address the above problem, DPAEKS adopts a dual-server framework, wherein the test functionality is split into two parts which are handled by two independent servers. None of the servers can do stand-alone testing. The security against the inside KGA is achieved assuming that the two servers are not colluded. Although the security of DPAEKS can be improved by introducing more servers, the multi-server scheme may suffer from the high communication complexity. The dual-server framework is considered to be a trade-off between security and efficiency.

Fig.2 depicts the system model of DPAEKS, which has four entities, namely the data owner (DO), the data receiver (DR), the assistant server (AS) and the test server (TS), respectively. Firstly, the DO sends the encrypted data to the AS and the TS. Then, the DR can query the stored data by sending a trapdoor. After receiving the request from the DR, the AS computes and sends the intermediate ciphertexts (ICTs) to TS. Upon receiving the ICTs, the TS tests the ICTs and returns the results to the DR. The roles of the AS and the TS are interchangeable meaning that the TS also can generate the ICTs and send them to the AS for executing the test algorithm. The feature can be used to improve the efficiency of the DPAEKS scheme in practical applications. Some lightweight authentication protocols [30], [31] can be introduced before the communication between the AS and the TS to ensure the security of our design framework. In addition, we focus on addressing the security of the index of keyword and the privacy protection of trapdoor in searchable public-key encryption schemes but how to encrypt files is not considered in this paper.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2945714, IEEE Transactions on Cloud Computing

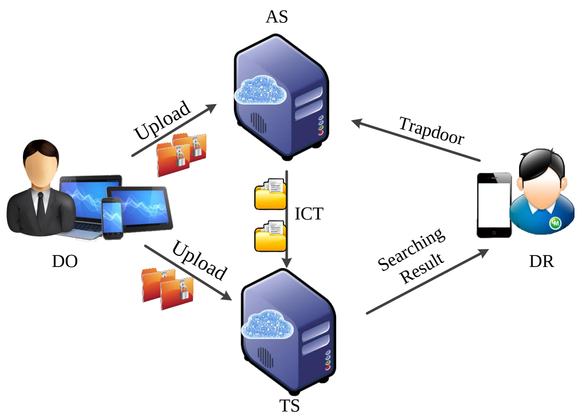

Fig. 2. System model of DPAEKS

D. Formal Definition

A DPAEKS scheme includes the following six polynomial algorithms: **Setup**, **KeyGen**, **DPAEKS**, **Trapdoor**, **Transition**, **Test**.

- **Setup**(λ): Takes as input a security parameter λ and outputs the public parameters $Para$.
- **KeyGen**($Para$): Takes as input $Para$ and outputs the public/secret key pairs (PK_{as}, SK_{as}), (PK_{ts}, SK_{ts}), (PK_{do}, SK_{do}) and (PK_{dr}, SK_{dr}) for AS, TS, DO and DR, respectively.
- **DPAEKS**($Para, PK_{as}, PK_{ts}, PK_{dr}, SK_{do}, w_i$): Takes as inputs the public parameters $Para$, AS's public key PK_{as}, TS's public key PK_{ts}, DR's public key PK_{dr}, DO's private key SK_{do} and a keyword w_i , and outputs the ciphertext CT_{w_i} of w_i.
- **Trapdoor**($Para, PK_{as}, PK_{ts}, PK_{do}, SK_{dr}, w_i$): Takes as inputs $Para$, AS's public key PK_{as}, TS's public key PK_{ts}, DO's public key PK_{do}, DR's private key SK_{dr} and a keyword $w_i^{'}$, and outputs the trapdoor $T_{w_i^{'}}$ of $w_i^{'}$.
- **Transition**($Para, SK_{as}, CT_{w_i}, T_{w_i^{'}}$): Takes as inputs $Para$, AS's secret key SK_{as}, the ciphertext CT_{w_i} and the trapdoor $T_{w_i^{'}}$, and outputs the intermediate ciphertext $ICT_{w_i w_i^{'}}$.
- **Test**($Para, SK_{ts}, ICT_{w_i w_i^{'}}$): Takes as inputs $Para$, TS's secret key SK_{ts} and the intermediate ciphertext $ICT_{w_i w_i^{'}}$, and outputs the testing result 0 or 1.

E. Security Models

We formalize the security models for DPAEKS, including indistinguishability against the chosen keyword attack (IND-CKA), indistinguishability against keyword guessing attack (IND-KGA) and indistinguishability against intermediate ciphertext guessing attack (IND-ICGA). To demonstrate the security of DPAEKS, we present the following security models against the adversarial assistant server and the adversarial test server. It is worth noting that our security models can protect against the external adversaries since these adversaries have less information than the inside servers (AS or TS). In addition, we assume that the AS and the TS honestly follow the proposed scheme, but they are curious about additional privacy information (such as the data, user's identity, user's interests). We also assume, as in schemes [4], [19], that the two servers do not collude, which can be achieved at the expense of some communication cost. For example, in a cloud-based data sharing system, the data owner provides data services through cloud servers from two different providers with competing interests.

Adversarial Assistant Server. If the AS is an honest-but-curious entity, (s)he should have no access to any information about the keyword by collecting the ciphertexts or the trapdoors sent by the DO. Formally, we define the following games, namely IND-CKA and IND-KGA. The games consist of two roles: a challenger C and an adversary A.

$Definition$ 1 (IND-CKA). IND-CKA guarantees that the DPAEKS ciphertext reveal no information about the underlying keyword.

$Setup$. Given a security parameter λ, C outputs the public parameters $Para$, and executes the **KeyGen**($Para$) algorithm to obtain four public/secret key pairs (PK_{as}, SK_{as}), (PK_{ts}, SK_{ts}), (PK_{do}, SK_{do}) and (PK_{dr}, SK_{dr}). It then gives (PK_{as}, SK_{as}, PK_{ts}, PK_{do}, PK_{dr}) to the adversary A.

$Phase$ 1. A can issue queries to the following oracles for polynomial times.

- **Ciphertext Oracle** $\mathcal{O}_c$: Given a keyword w_i, the oracle computes the ciphertext CT_{w_i} of w_i with respect to $PK_{as}, PK_{ts}, PK_{dr}$ and SK_{do}, and returns CT_{w_i} to A.
- **Trapdoor Oracle** $\mathcal{O}_t$: Given a keyword w_i, the oracle computes the trapdoor T_{w_i} of w_i with respect to $PK_{as}, PK_{ts}, PK_{do}$ and SK_{dr}, and returns T_{w_i} to A.
- **Test Oracle** $\mathcal{O}_{test}$: Given an intermediate ciphertext $ICT_{w_i w_i^{'}}$, the oracle computes the test result (1 or 0) using SK_{ts} and returns it to A.

$Challenge$. If A decides that phase 1 is over, it sends a challenge keyword pair (w_0, w_1) to C. Upon receiving them, C first picks $b \xleftarrow{R} \{0, 1\}$ and calculates:

$$CT_{w_b} \leftarrow \textbf{DPAEKS}(Para, PK_{as}, PK_{ts}, PK_{dr}, SK_{do}, w_b)$$

The challenger C then sends CT_{w_b} to A.

$Phase$ 2. A can query $\mathcal{O}_c, \mathcal{O}_t$ and $\mathcal{O}_{test}$ like phase 1. The only restriction is that neither w_0 nor w_1 could be submitted to the above oracles.

$Guess$. Finally, A gives a value $b^{'} \in \{0, 1\}$ as its guess and wins the game if $b = b^{'}$.

The adversarial assistant server A in the above game is defined as an IND-CKA adversary. A's advantage is defined as:

$$Adv_{as,A}^{IND-CKA}(\lambda) = |Pr[b = b^{'}] - \frac{1}{2}|$$

$Definition$ 2 (IND-KGA). IND-KGA ensures that a trapdoor reveal no information about keyword to the adversarial assistant server.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2945714, IEEE Transactions on Cloud Computing

Setup. Given a security parameter λ, C executes $Para \leftarrow$ **Setup**(λ) and runs the **KeyGen**$(Para)$ algorithm to obtain four public/secret key pairs (PK_{as}, SK_{as}), (PK_{ts}, SK_{ts}), (PK_{do}, SK_{do}) and (PK_{dr}, SK_{dr}), respectively. It then gives $(PK_{as}, SK_{as}, PK_{ts}, PK_{do}, PK_{dr})$ to the adversary A.

Phase 1. A is allowed to issue queries to oracles $\mathcal{O}_c$, $\mathcal{O}_t$ and $\mathcal{O}_{test}$ for polynomial times as in the IND-CKA game.

Challenge. When A decides that phase 1 is over, it sends a challenge keyword pair (w_0, w_1) to the challenger C. Upon receiving the keyword pair, C first picks $b \xleftarrow{R} \{0, 1\}$ and computes:

$$T_{w_b} \leftarrow \textbf{Trapdoor}(Para, PK_{as}, PK_{ts}, PK_{do}, SK_{dr}, w_b)$$

The challenger C then sends T_{w_b} to A.

Phase 2. A can query $\mathcal{O}_c, \mathcal{O}_t$ and $\mathcal{O}_{test}$ like phase 1. The only restriction is that neither w_0 nor w_1 could be submitted to the oracles.

Guess. Finally, A outputs a value $b' \in \{0, 1\}$ as its guess and wins the game if $b = b'$.

The adversarial assistant server A in the IND-KGA game is defined as an IND-KGA adversary, and then its advantage is defined as:

$$Adv_{as,A}^{IND-KGA}(\lambda) = |Pr[b = b'] - \frac{1}{2}|$$

Adversarial Test Server. If the test server is an honest-but-curious entity, (s)he should have no access to any information about the keyword through collecting the DPAEKS ciphertexts, the trapdoors, and the intermediate ciphertexts. Since the role of the test server is similar to the assistant server, the models of IND-CKA and IND-KGA closely resemble those against the adversarial assistant server. The only difference between the two adversaries is that the own secret key is different. Therefore, the details of the security models are omitted here. In addition, since the intermediate ciphertexts are generated and transmitted during the execution of our scheme, consideration should be given to prevent the intermediate ciphertext from leaking any information about the keyword. Formally, we give a new game to define the security model of intermediate ciphertext guessing attack, namely IND-ICGA.

Definition 3 (IND-ICGA). IND-ICGA ensures that the adversarial test server can not learn any information about keywords from the intermediate ciphertexts.

Setup. Given a security parameter λ, C executes $Para \leftarrow$ **Setup**(λ) and runs the **KeyGen**$(Para)$ to obtain four public/secret key pairs (PK_{as}, SK_{as}), (PK_{ts}, SK_{ts}), (PK_{do}, SK_{do}) and (PK_{dr}, SK_{dr}). It then gives $(PK_{as}, PK_{ts}, SK_{ts}, PK_{do}, PK_{dr})$ to A.

Phase 1. A can issue queries to the following oracles for polynomial times.

- **Ciphertext Oracle** $\mathcal{O}_c$: Given a keyword w_i, the oracle computes the ciphertext CT_{w_i} of w_i with respect to $PK_{as}, PK_{ts}, PK_{dr}$ and SK_{do}, and returns CT_{w_i} to A.
- **Trapdoor Oracle** $\mathcal{O}_t$: Given a keyword w_i, the oracle computes the corresponding trapdoor T_{w_i} with respect to $PK_{as}, PK_{ts}, PK_{do}$ and SK_{dr}, and returns T_{w_i} to A.
- **Transition Oracle** $\mathcal{O}_{tran}$: Given a ciphertext CT_{w_i} and a trapdoor $T_{w_i'}$, the oracle computes the intermediate ciphertext $ICT_{w_i w_i'}$ with SK_{as}, returns the result to A.

Challenge. When A claims that phase 1 is over, it sends three challenge keywords (w_0, w_1, w_2) to C. Upon receiving them, C first picks $\{b_0, b_1\} \xleftarrow{R} \{0, 1, 2\}$ and computes

$$\begin{aligned} CT_{w_{b_0}} \leftarrow & \textbf{DPAEKS}(Para, PK_{as}, PK_{ts}, \\ & PK_{dr}, SK_{do}, w_{b_0}); \\ T_{w_{b_1}} \leftarrow & \textbf{Trapdoor}(Para, PK_{as}, PK_{ts}, \\ & PK_{do}, SK_{dr}, w_{b_1}); \\ ICT_{w_{b_0} w_{b_1}} \leftarrow & \textbf{Transition}(Para, SK_{as}, CT_{w_{b_0}}, T_{w_{b_1}}); \end{aligned}$$

The challenger C then sends $ICT_{w_{b_0} w_{b_1}}$ to A.

Guess. Finally, the adversary A outputs its guess on $\{b_0, b_1\}$ as $\{b_0', b_1'\}$, where $b_0', b_1' \in \{0, 1, 2\}$

We define an adversarial test server A as an IND-ICGA adversary in above game and its advantage is defined as follows:

$$Adv_{ts,A}^{IND-ICGA}(\lambda) = |Pr[\{b_0, b_1\} = \{b_0', b_1'\}] - \frac{1}{3}|$$

IV. CONSTRUCTION OF DPAEKS

We present a concrete construction of DPAEKS based on the DDH assumption. Then, we present an analysis of the soundness and security of DPAEKS.

A. DPAEKS

Our scheme comprises six algorithms: the **Setup** algorithm, the **Keygen** algorithm, the **DPAEKS** algorithm, the **Trapdoor** algorithm, the **Transition** algorithm and the **Test** algorithm. The scheme works as follows:

Setup(λ): It takes as input a security parameter λ, and selects an elliptic curve group $\mathbb{G}_1$ which has a prime order q. Suppose P_1, P_2, P_3 are three different generators of $\mathbb{G}_1$. Next, it selects a secure hash function $h_1 : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$. Finally, it publishes the public parameters $Para = \{q, \mathbb{G}_1, P_1, P_2, P_3, h_1\}$.

KeyGen$(Para)$: It takes as input the public parameters $Para$. Next, it selects $(a, b, c, d) \xleftarrow{R} \mathbb{Z}_q^*$ and outputs the public/secret key pair (PK_{as}, SK_{as}), (PK_{ts}, SK_{ts}), (PK_{do}, SK_{do}) and (PK_{dr}, SK_{dr}) for AS, TS, DO and DR respectively. The process is as follows:

$$\begin{aligned} PK_{as} = aP_1, SK_{as} = a \\ PK_{ts} = bP_2, SK_{ts} = b \\ PK_{do} = cP_3, SK_{do} = c \\ PK_{dr} = dP_3, SK_{dr} = d \end{aligned}$$

DPAEKS$(Para, PK_{as}, PK_{ts}, PK_{dr}, SK_{do}, w_i)$: It takes as inputs $Para$, the public keys $PK_{as}, PK_{ts}, PK_{dr}$ of AS, TS and DR respectively, the DO's secret key SK_{do} and a keyword w_i. The algorithm proceeds as follows:

– picks $r_1 \xleftarrow{R} \mathbb{Z}_q^*$ randomly.
– computes $C_{1w_i} = r_1 P_1$ and $C_{2w_i} = r_1 P_2$.

– computes $C_{3w_i} = r_1 PK_{as} + r_1 PK_{ts} + h_1(w_i) SK_{do} PK_{dr} = r_1 a P_1 + r_1 b P_2 + h_1(w_i) cd P_3$.

It outputs the ciphertext $CT_{w_i} = (C_{1w_i}, C_{2w_i}, C_{3w_i})$ of w_i.

Trapdoor$(Para, PK_{as}, PK_{ts}, PK_{do}, SK_{dr}, w_i^{'})$: It takes as inputs $Para$, the public keys PK_{as}, PK_{ts}, PK_{do} of AS, TS and DO respectively, the DR's secret key SK_{dr} and $w_i^{'}$. The algorithm proceeds as follows:

– picks $r_2 \stackrel{R}{\longleftarrow} \mathbb{Z}_q^*$ randomly.
– computes $T_{1w_i^{'}} = r_2 P_1$ and $T_{2w_i^{'}} = r_2 P_2$.
– computes $T_{3w_i^{'}} = r_2 PK_{as} + r_2 PK_{ts} - h_1(w_i^{'}) SK_{dr} PK_{do} = r_2 a P_1 + r_2 b P_2 - h_1(w_i^{'}) dc P_3$.

Finally, it outputs the trapdoor $T_{w_i} = (T_{1w_i^{'}}, T_{2w_i^{'}}, T_{3w_i^{'}})$ of $w_i^{'}$.

Transition$(Para, SK_{as}, CT_{w_i}, T_{w_i^{'}})$: It takes as inputs the public parameters $Para$, the AS's secret key SK_{as}, the ciphertext CT_{w_i} of w_i and the trapdoor $T_{w_i^{'}}$ of $w_i^{'}$. The algorithm proceeds as follows:

– computes $(CT_{w_i} + T_{w_i^{'}}) = (C_{1w_i} + T_{1w_i^{'}}, C_{2w_i} + T_{2w_i^{'}}, C_{3w_i} + T_{3w_i^{'}})$ as follows:

$$\begin{aligned} C_{1w_i} + T_{1w_i^{'}} &= (r_1 P_1 + r_2 P_1) = (r_1 + r_2) P_1 \\ C_{2w_i} + T_{2w_i^{'}} &= (r_1 P_2 + r_2 P_2) = (r_1 + r_2) P_2 \\ C_{3w_i} + T_{3w_i^{'}} &= (r_1 a P_1 + r_1 b P_2 + h_1(w_i) cd P_3) \\ &\quad + (r_2 a P_1 + r_2 b P_2 - h_1(w_i^{'}) dc P_3) \\ &= (r_1 + r_2) a P_1 + (r_1 + r_2) b P_2 \\ &\quad + (h_1(w_i) - h_1(w_i^{'})) cd P_3 \end{aligned}$$

– chooses a random number $r_3 \in \mathbb{Z}_q^*$ and computes $ICT^1_{w_i w_i^{'}} = r_3(C_{2w_i} + T_{2w_i^{'}}) = r_3(r_1 + r_2) P_2$.
– computes $ICT^2_{w_i w_i^{'}} = r_3((C_{3w_i} + T_{3w_i^{'}}) - a(C_{1w_i} + T_{1w_i^{'}})) = r_3(r_1 + r_2) b P_2 + r_3(h_1(w_i) - h_1(w_i^{'})) cd P_3$

It outputs the intermediate ciphertext $ICT_{w_i w_i^{'}} = (ICT^1_{w_i w_i^{'}}, ICT^2_{w_i w_i^{'}})$.

Test$(Para, SK_{ts}, ICT_{w_i w_i^{'}})$: It takes as inputs the public parameters $Para$, the TS's secret key SK_{ts} and the intermediate ciphertext $ICT_{w_i w_i^{'}}$. It outputs 1 if the following equation holds:

$$SK_{ts} ICT^1_{w_i w_i^{'}} \stackrel{?}{=} ICT^2_{w_i w_i^{'}}$$

otherwise it outputs 0.

B. Soundness of DPAEKS

In this subsection, we first formalize the soundness of DPAEKS. Then we prove that our construction satisfies the soundness criterion under $Definition$ 4.

$Definition$ 4 ($Soundness$). For any two keywords (w_1, w_2), the definition of soundness is formally defined as follows:

$$\begin{aligned} Para &\leftarrow \textbf{Setup}(\lambda); \\ (PK_i, SK_i) &\leftarrow \textbf{KeyGen}(Para) \\ &\qquad where \quad i = as, ts, do, dr; \\ CT_{w_1} &\leftarrow \textbf{DPAEKS}(Para, PK_{as}, PK_{ts}, \\ &\qquad PK_{dr}, SK_{do}, w_1); \\ T_{w_2} &\leftarrow \textbf{Trapdoor}(Para, PK_{as}, PK_{ts}, \\ &\qquad PK_{do}, SK_{dr}, w_2); \\ ICT_{w_1 w_2} &\leftarrow \textbf{Transition}(Para, SK_{as}, CT_{w_1}, T_{w_2}); \\ if \quad w_1 = w_2, & \\ &\textbf{Test}(Para, SK_{ts}, ICT_{w_1 w_2}) = 1, \\ else & \\ &\textbf{Test}(Para, SK_{ts}, ICT_{w_1 w_2}) = 0 \end{aligned}$$

Theorem 1. DPAEKS satisfies the soundness criterion if and only if the hash function is collision resistant.

$Proof$. To simulate our scheme, we build a challenger C. C sets the public parameters $Para = \{q, \mathbb{G}_1, P_1, P_2, P_3, h_1\}$ and generates the public/secret key pairs $(PK_{as} = aP_1, SK_{as} = a)$, $(PK_{ts} = bP_2, SK_{ts} = b)$, $(PK_{do} = cP_3, SK_{do} = c)$ and $(PK_{dr} = dP_3, SK_{dr} = d)$ of AS, TS, DO and DR respectively. We assume the secure hash function h_1 is collision resistant. We assume the probabilistic polynomial time (PPT) adversary A selects two different keywords (w_1, w_2) to attack the soundness of our scheme, where $w_1 \neq w_2$.

C runs the DPAEKS algorithm to compute the ciphertext $CT_{w_1} = (C_{1w_1} = r_1 P_1, C_{2w_1} = r_1 P_2, C_{3w_1} = r_1 a P_1 + r_1 b P_2 + h_1(w_i) cd P_3)$ of keyword w_1, where $r_1 \stackrel{R}{\longleftarrow} \mathbb{Z}_q^*$. C executes the **Trapdoor** algorithm to compute the trapdoor $T_{w_2} = (T_{1w_2} = r_2 P_1, T_{2w_2} = r_2 P_2, T_{3w_2} = r_2 a P_1 + r_2 b P_2 - h_1(w_2) dc P_3)$ of w_2, where $r_2 \stackrel{R}{\longleftarrow} \mathbb{Z}_q^*$.

For the **Transition** algorithm, the process is as follows:

$$\begin{aligned} C_{1w_1} + T_{1w_2} &= (r_1 + r_2) P_1 \\ C_{2w_1} + T_{2w_2} &= (r_1 + r_2) P_2 \\ C_{3w_1} + T_{3w_2} &= (r_1 a P_1 + r_1 b P_2 + h_1(w_1) cd P_3) \\ &\quad + (r_2 a P_1 + r_2 b P_2 - h_1(w_2) dc P_3) \\ &= (r_1 + r_2) a P_1 + (r_1 + r_2) b P_2 \\ &\quad + (h_1(w_1) - h_1(w_2)) cd P_3 \\ ICT^1_{w_1 w_2} &= r_3(C_{2w_1} + T_{2w_2}) = r_3(r_1 + r_2) P_2 \\ ICT^2_{w_1 w_2} &= r_3((C_{3w_1} + T_{3w_2}) - a(C_{1w_1} + T_{1w_2})) \\ &= r_3(r_1 + r_2) b P_2 + r_3(h_1(w_1) - h_1(w_2)) cd P_3 \end{aligned}$$

where $r_3 \stackrel{R}{\longleftarrow} \mathbb{Z}_q^*$.

For the **Test** algorithm, the process is as follows:

$$\begin{aligned} SK_{ts} ICT^1_{w_1 w_2} &= b r_3 (r_1 + r_2) P_2 \\ ICT^2_{w_1 w_2} &= r_3(r_1 + r_2) b P_2 + r_3(h_1(w_1) - h_1(w_2)) cd P_3 \\ SK_{ts} ICT^1_{w_1 w_2} &\stackrel{?}{=} ICT^2_{w_1 w_2} \end{aligned}$$

Since $w_1 \neq w_2$ and the hash function h_1 is collision resistant, the inequalities $h_1(w_1) \neq h_1(w_2)$ and $SK_{ts} ICT^1_{w_1 w_2} \neq$

$ICT^2_{w_1w_2}$ holds. The test algorithm output 1 with probability $negl(\lambda)$. Therefore, if h_1 is collision resistant, our scheme satisfies the soundness criterion.

C. Security

Theorem 2. DPAEKS is IND-CKA secure in the random oracle model if the DDH assumption holds.

Depending on the following lemmas, we can prove that the theorem 2 is correct.

Lemma 1. For any PPT adversary A, $Adv^{IND-CKA}_{as,A}(\lambda)$ is negligible under $Definition$ 1.

$Proof$. We create three games as follows:

$Game$ 0: The challenger C performs everything in the IND-CKA game.

$Setup$. Given a security parameter λ, C outputs the public parameters $Para = \{q, \mathbb{G}_1, P_1, P_2, P_3, h_1\}$, and runs the **KeyGen**($Para$) algorithm to obtain four public/secret key pairs ($PK_{as} = aP_1, SK_{as} = a$), ($PK_{ts} = bP_2, SK_{ts} = b$), ($PK_{do} = cP_3, SK_{do} = c$) and ($PK_{dr} = dP_3, SK_{dr} = d$). It then gives $(PK_{as}, SK_{as}, PK_{ts}, PK_{do}, PK_{dr})$ to the adversary A.

$Phase$ 1. A issues the following queries adaptively.

Ciphertext Oracle $\mathcal{O}_c$: Upon receiving a keyword w_i, C first chooses $r_1 \xleftarrow{R} \mathbb{Z}^*_q$ randomly, and computes the ciphertext $C_{w_i} = \{C_{1w_i} = r_1P_1, C_{2w_i} = r_1P_2, C_{3w_i} = r_1PK_{as} + r_1PK_{ts} + h_1(w_i)SK_{do}PK_{dr}\}$. C returns C_{w_i} to A.

Trapdoor Oracle $\mathcal{O}_t$: Upon receiving a keyword w_i, C randomly chooses $r_2 \xleftarrow{R} \mathbb{Z}^*_q$, and computes the trapdoor $T_{w_i} = \{T_{1w_i} = r_2P_1, T_{2w_i} = r_2P_2, T_{3w_i} = r_2PK_{as} + r_2PK_{ts} - h_1(w_i)SK_{dr}PK_{do}\}$. C returns T_{w_i} to A.

Test Oracle $\mathcal{O}_{test}$: Upon receiving an intermediate ciphertext $ICT_{w_iw_i'} = \{ICT^1_{w_iw_i'}, ICT^2_{w_iw_i'}\}$, C runs the algorithm **Test**$(Para, SK_{ts}, ICT_{w_iw_i'})$ and returns the output to A.

$Challenge$. Once phase 1 is over, A adaptively selects two different keywords (w_0, w_1), which need to be challenged. Upon receiving w_0, w_1, C first selects $b \xleftarrow{R} \{0, 1\}$. Then C computes the ciphertext $CT^*_{w_b}$ of w_b as follows:

$$C^*_{1w_b} = r^*_1P_1$$
$$C^*_{2w_b} = r^*_1P_2$$
$$C^*_{3w_b} = r^*_1PK_{as} + r^*_1PK_{ts} + h_1(w_b)SK_{do}PK_{dr}$$

where $r^*_1 \xleftarrow{R} \mathbb{Z}^*_q$.

Finally, C sets $CT^*_{w_b} = (CT^*_{1w_b}, CT^*_{2w_b}, CT^*_{3w_b})$ as the keyword ciphertext and sends $CT^*_{w_b}$ to A.

$Phase$ 2. A can continue to issue $\mathcal{O}_c$, $\mathcal{O}_t$ and $\mathcal{O}_{test}$ as phase 1. The only restriction here is that neither w_0 nor w_1 could be submitted to the oracles.

$Guess$. Finally, A outputs a bit $b^{'} \in \{0, 1\}$ and wins the IND-CKA game if $b = b^{'}$.

If A's advantage in this game is $Adv^{Game\,0-CKA}_{as,A}(\lambda)$, then we have

$$Adv^{Game\,0-CKA}_{as,A}(\lambda) = Adv^{IND-CKA}_{as,A}(\lambda)$$

.

$Game$ 1. Let $Game$ 1 be the same as $Game$ 0, except that C computes $CT^{**}_{3w_b} = r^*_1PK_{as} + r^*_1PK_{ts} + h_1(w_b)r^{**}P_3$ instead of computing $CT^*_{3w_b} = r^*_1PK_{as} + r^*_1PK_{ts} + h_1(w_b)SK_{do}PK_{dr}$, where $r^{**} \xleftarrow{R} \mathbb{Z}^*_q$. The challenger sends the ciphertext $CT^{**}_{w_b} = (CT^*_{1w_b}, CT^*_{2w_b}, CT^{**}_{3w_b})$ to A. According to the **Claim** 1, we have:

$$|Adv^{Game\,1-CKA}_{as,A}(\lambda) - Adv^{Game\,0-CKA}_{as,A}(\lambda)| \leq Adv^{DDH}_{A}(\lambda)$$

where $Adv^{Game\,1-CKA}_{as,A}(\lambda)$ is the advantage of A in $Game$ 1 and $Adv^{DDH}_{A}(\lambda)$ is negligible if the DDH assumption holds.

Claim 1. If the DDH assumption holds, we have:

$$|Adv^{Game\,1-CKA}_{as,A}(\lambda) - Adv^{Game\,0-CKA}_{as,A}(\lambda)| \leq Adv^{DDH}_{A}(\lambda)$$

Proof. Given a four tuple (P_3, xP_3, yP_3, T), where $x, y \xleftarrow{R} \mathbb{Z}^*_q$ and $T \in \mathbb{G}_1$. The four tuple may be a DDH tuple, and we have $T = xyP_3$. Otherwise, $T \xleftarrow{R} \mathbb{G}_1$. In $Game$ 0, we assume that $PK_{as} = xP_3, PK_{ts} = yP_3$, then the $C^*_{3w_b} = r^P_1K_{as} + r^*_1PK_{ts} + h_1(w_b)xyP_3$. Additionally, we have $C^{**}_{3w_b} = r^*_1PK_{as} + r^*_1PK_{ts} + h_1(w_b)T$ in $Game$ 1. It is impossible to distinguish between $T \xleftarrow{R} \mathbb{G}_1$ and $T = xyP_3 \in \mathbb{G}_1$ if the DDH assumption holds. Thus, The inequation holds:

$$|Adv^{Game\,1-CKA}_{as,A}(\lambda) - Adv^{Game\,0-CKA}_{as,A}(\lambda)| \leq Adv^{DDH}_{A}(\lambda)$$

$Game$ 2. Let $Game$ 2 be the same game as $Game$ 1, except that the challenge selects $CT^{***}_{3w_b} \xleftarrow{R} \mathbb{G}_1$ instead of computing $CT^{**}_{3w_b} = r^*_1PK_{as} + r^*_1PK_{ts} + h_1(w_b)r^{**}P_3$. Due to $r^{**} \xleftarrow{R} \mathbb{Z}^*_q$ in $Game$ 1, the distribution $CT^{***}_{3w_b}$ and $CT^{**}_{3w_b}$ are indistinguishable in A's view. Thus, the equation holds:

$$Adv^{Game\,2-CKA}_{as,A}(\lambda) = Adv^{Game\,1-CKA}_{as,A}(\lambda)$$

Since the ciphertext $CT^{***}_{3w_b}$ is independent of the keywords (w_0, w_1) in $Game1$, A wins the $Game$ 2 with probability $\frac{1}{2}$. Therefore, the equation $Adv^{Game\,2-CKA}_{as,A}(\lambda) = 0$ holds.

Depending on the above games, the inequation holds:

$$|Adv^{Game\,2-CKA}_{as,A}(\lambda) - Adv^{IND-CKA}_{as,A}(\lambda)| \leq Adv^{DDH}_{A}(\lambda)$$

because $Adv^{Game\,2-CKA}_{as,A}(\lambda) = \frac{1}{2} - \frac{1}{2} = 0$ and $Adv^{DDH}_{A}(\lambda) \leq negl(\lambda)$, that is, $Adv^{IND-CKA}_{as,A}(\lambda)$ is negligible.

Lemma 2. For any PPT adversary A, $Adv^{IND-CKA}_{ts,A}(\lambda)$ is negligible.

The proof process of **Lemma 2** is similar to that of **Lemma 1**. The main difference is that the adversary possesses the private key of the test server ts. For simplicity, we omit the proof process of **Lemma 2**.

Theorem 3. DPAEKS is IND-KGA secure in the random oracle model if the DDH assumption holds.

Lemma 3. For any PPT adversary A, $Adv^{IND-KGA}_{as,A}(\lambda)$ is negligible under $Definition$ 2.

$Proof$. We create three games as follows:

$Game$ 0: C performs everything in the IND-KGA game.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2945714, IEEE Transactions on Cloud Computing

Setup. Given a security parameter λ, C outputs the system parameters $Para = \{q, \mathbb{G}_1, P_1, P_2, P_3, h_1\}$, and runs the $KeyGen(Para)$ to obtain key pairs ($PK_{as} = aP_1, SK_{as} = a$), ($PK_{ts} = bP_2, SK_{ts} = b$), ($PK_{do} = cP_3, SK_{do} = c$) and ($PK_{dr} = dP_3, SK_{dr} = d$). It then gives ($PK_{as}, PK_{ts}, SK_{as}, PK_{do}, PK_{dr}$) to the adversary A.

Phase 1. A can issue queries to oracles $\mathcal{O}_c, \mathcal{O}_t$ and $\mathcal{O}_{test}$ as **Lemma** 1.

Challenge. Once phase 1 is over, A selects a challenge keyword pair (w_0, w_1) and sends it to C. Upon receiving (w_0, w_1), C first picks $b \xleftarrow{R} \{0, 1\}$. Then C computes the trapdoor $T^*_{w_b} = (T_{1w_b}, T_{2w_b}, T_{3w_b})$ of w_b as follows:

$$T^*_{1w_b} = r^*_2 P_1$$
$$T^*_{2w_b} = r^*_2 P_2$$
$$T^*_{3w_b} = r^*_2 PK_{as} + r^*_2 PK_{ts} - h_1(w_b) SK_{dr} PK_{do}$$

where $r^*_2 \xleftarrow{R} \mathbb{Z}P^*_q$. Finally, C sends the trapdoor $T^*_{w_b}$ to A.

Phase 2. A can continue to issue queries to oracles $\mathcal{O}_c, \mathcal{O}_t$ and $\mathcal{O}_{test}$ as in phase 1. The restriction here is that neither w_0 nor w_1 could be submitted to the oracles.

Guess. Finally, A outputs a bit $b^{'} \in \{0, 1\}$ and wins the IND-KGA game if $b = b^{'}$.

If A's advantage in this game is $Adv^{Game\,0-KGA}_{as,A}(\lambda)$ and we have

$$Adv^{Game\,0-KGA}_{as,A}(\lambda) = Adv^{IND-KGA}_{as,A}(\lambda)$$

.

Game 1. Let *Game* 1 be the same as *Game* 0, except that C picks $T^{**}_{3w_b} = r^*_2 PK_{as} + r^*_2 PK_{ts} - h_1(w_b) r^{**} P_3$ instead of computing $T^*_{3w_b} = r^*_2 PK_{as} + r^*_2 PK_{ts} - h_1(w_b) SK_{do} PK_{dr}$, where $r^{**} \xleftarrow{R} \mathbb{Z}^*_q$. C sends the trapdoor $T^{**}_{w_b} = (T^*_{1w_b}, T^*_{2w_b}, T^{**}_{3w_b})$ to the adversary A. According to the **Claim** 1, we have

$$|Adv^{Game\,1-KGA}_{as,A}(\lambda) - Adv^{Game\,0-KGA}_{as,A}(\lambda)| \leq Adv^{DDH}_{A}(\lambda)$$

where $Adv^{Game\,1-KGA}_{as,A}(\lambda)$ is the advantage of A in *Game* 1.

Game 2. Let *Game* 2 be the same as *Game* 1, except that C chooses $T^{***}_{3w_b} \xleftarrow{R} \mathbb{G}_1$ instead of computing $T^{**}_{3w_b} = r^*_2 PK_{as} + r^*_2 PK_{ts} - h_1(w_b) r^{**} P_3$. Due to $r^{**} \xleftarrow{R} \mathbb{Z}^*_q$ in $T^{**}_{3w_b}$, the distribution $T^{***}_{3w_b}$ and $T^{**}_{3w_b}$ are indistinguishable in the adversary's view. Therefore, the equation holds

$$Adv^{Game\,2-KGA}_{as,A}(\lambda) = Adv^{Game\,1-KGA}_{as,A}(\lambda)$$

Since $T^{***}_{3w_b}$ is independent of the keywords (w_0, w_1), A can win the *Game* 2 with probability $\frac{1}{2}$. Therefore, the equation $Adv^{Game\,2-KGA}_{as,A}(\lambda) = \frac{1}{2} - \frac{1}{2} = 0$

Depending on the above games, we know that

$$|Adv^{Game\,2-KGA}_{as,A}(\lambda) - Adv^{IND-KGA}_{as,A}(\lambda)| \leq Adv^{DDH}_{A}(\lambda)$$

because $Adv^{Game\,2-KGA}_{as,A}(\lambda) = 0$ and $\leq Adv^{DDH}_{A}(\lambda) \leq negl(\lambda)$, that is, $Adv^{IND-KGA}_{as,A}(\lambda)$ is negligible.

Lemma 4. For any PPT adversary A, $Adv^{IND-KGA}_{ts,A}(\lambda)$ is negligible.

The proof process of **Lemma 4** is similar to that of **Lemma 3**, and the main difference is that the adversary obtains the private key SK_{ts} of TS instead of SK_{as}. For simplicity, we omit the proof process of **Lemma 4**.

Theorem 4 Our scheme is IND-ICGA secure in the random oracle model.

Lemma 5. For any PPT adversary A, $Adv^{IND-ICGA}_{ts,A}(\lambda)$ is negligible under *Definition* 3.

Proof. According to the description of our scheme, if $b_0 = b_1$ in the IND-ICGA game, the intermediate ciphertext $ICT_{w_{b_0} w_{b_1}} = (ICT^1_{w_{b_0} w_{b_1}}, ICT^2_{w_{b_0} w_{b_1}})$ will no contain the keywords information. As two keywords cancel each other when computing the intermediate ciphertext $ICT^2_{w_{b_0} w_{b_1}}$, the adversary has no advantage to guess the keywords by the given intermediate ciphertext. Thus, we only consider the case that $b_0 \neq b_1$.

Next, we create two games as follows:

Game 0: The challenger C performs everything in the IND-ICGA game.

Setup. Given a security parameter λ, C outputs the public parameters $Para = \{q, \mathbb{G}_1, P_1, P_2, P_3, h_1\}$, and runs the $KeyGen(Para)$ algorithm to obtain key pairs ($PK_{as} = aP_1, SK_{as} = a$), ($PK_{ts} = bP_2, SK_{ts} = b$), ($PK_{do} = cP_3, SK_{do} = c$) and ($PK_{dr} = dP_3, SK_{dr} = d$). It then gives ($PK_{as}, PK_{ts}, SK_{ts}, PK_{do}, PK_{dr}$) to the adversary A.

*Phase*1. The adversary A is allowed to issue queries to oracles $\mathcal{O}_c, \mathcal{O}_t$ and $\mathcal{O}_{tran}$ as **Lemma** 1.

Transition Oracle $\mathcal{O}_{tran}$: Upon receiving a ciphertext $CT_{w_i} = (C_{1w_i}, C_{2w_i}, C_{3w_i})$ of w_i and a trapdoor $T_{w^{'}_i} = (T_{1w^{'}_i}, T_{2w^{'}_i}, T_{3w^{'}_i})$ of $w^{'}_i$, C performs the following steps:

$$ICT^1_{w_i w^{'}_i} = r_3 (C_{2w_i} + T_{2w^{'}_i})$$
$$ICT^2_{w_i w^{'}_i} = r_3 (C_{3w_i} + T_{3w^{'}_i} - SK_{as}(C_{1w_i} + T_{1w^{'}_i}))$$

where $r_3 \xleftarrow{R} \mathbb{Z}^*_q$. C returns $(ICT^1_{w_i w^{'}_i}, ICT^2_{w_i w^{'}_i})$ to A.

Challenge. Once phase 1 is over, A adaptively selects three challenge keywords w_0, w_1, w_2 and sends them to C. Then C randomly selects $(b_0, b_1) \xleftarrow{R} \{0, 1, 2\}$ and generates the intermediate ciphertext $ICT_{w_{b_0} w_{b_1}}$.

$$\begin{aligned}
(C_{1w_{b_0}}, C_{2w_{b_0}}, C_{3w_{b_0}}) &= (r_1 P_1, r_1 P_2, r_1 PK_{as} + r_1 PK_{ts} \\
&\quad + h_1(w_{b_0}) SK_{do} PK_{dr}) \\
(T_{1w_{b_1}}, T_{2w_{b_1}}, T_{3w_{b_1}}) &= (r_2 P_1, r_2 P_2, r_2 PK_{as} + r_2 PK_{ts} \\
&\quad - h_1(w_{b_1}) SK_{dr} PK_{do}) \\
ICT^1_{w_{b_0} w_{b_1}} &= r^*_3 (C_{2w_{b_0}} + T_{2w_{b_1}}) \\
&= r^*_3 (r_1 + r_2) P_2 \\
ICT^2_{w_{b_0} w_{b_1}} &= r^*_3 (C_{3w_{b_0}} + T_{3w_{b_1}}) \\
&\quad - r^*_3 SK_{as} (C_{1w_{b_0}} + T_{1w_{b_1}}) \\
&= r^*_3 ((r_1 + r_2) PK_{ts}) + r^*_3 ((h_1(w_{b_0}) \\
&\quad - h_1(w_{b_1})) SK_{do} PK_{dr})
\end{aligned}$$

where $r^*_3 \xleftarrow{R} \mathbb{Z}^*_q$. C returns $(ICT^1_{w_{b_0} w_{b_1}}, ICT^2_{w_{b_0} w_{b_1}})$ to A.

Guess. Finally, A outputs $\{b^{'}_0, b^{'}_1\}$ as its guess and wins the IND-ICGA game if $\{b^{'}_0, b^{'}_1\} = \{b_0, b_1\}$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2945714, IEEE Transactions on Cloud Computing

If A's advantage in this game 0 is $Adv_{ts,A}^{Game\,0}(\lambda)$, the equation holds:

$$Adv_{ts,A}^{Game\,0}(\lambda) = Adv_{ts,A}^{IND-ICGA}(\lambda)$$

because $Game\,0$ is the same as IND-ICGA.

$Game$ 1. Assume that $Game$ 1 strictly follows the $Game$ 0, except that C picks $ICT^{2^*}_{w_{b_0}w_{b_1}} \xleftarrow{R} \mathbb{G}_1$ instead of computing $ICT^{2}_{w_{b_0}w_{b_1}} = r_3^*(C_{3w_{b_0}} + T_{3w_{b_1}}) - r_3^* SK_{as}(C_{1w_{b_0}} + T_{1w_{b_1}})$. since the r_3^* is randomly selected, the distribution of $ICT^{2}_{w_{b_0}w_{b_1}}$ and $ICT^{2^*}_{w_{b_0}w_{b_1}} \xleftarrow{R} \mathbb{G}_1$ is computationally indistinguishable in the A's view. So we can obtain the following:

$$|Adv_{ts,A}^{Game\,1-ICGA}(\lambda) - Adv_{ts,A}^{Game\,0-ICGA}(\lambda)| = 0$$

where $Adv_{ts,A}^{Game\,1-ICGA}(\lambda)$ denotes the advantage of A winning the game $Game$ 1.

In other words, the intermediate ciphertext $ICT_{w_{b_0}w_{b_1}} = (ICT^{1^*}_{w_{b_0}w_{b_1}}, ICT^{2*}_{w_{b_0}w_{b_1}})$ does not contain any information about its corresponding keyword. Therefore, the adversary only wins with probability $\frac{1}{3}$, meaning that the adversary randomly chooses two values from the three values $\{0, 1, 2\}$. Finally, we have:

$$|Adv_{ts,A}^{Game\,1}(\lambda) - Adv_{ts,A}^{IND-ICGA}(\lambda)| = 0$$

As $Adv_{ts,A}^{Game\,1}(\lambda) = \frac{1}{3} - \frac{1}{3} = 0$, we can say $Adv_{ts,A}^{IND-ICGA}(\lambda)$ is a negligible.

V. IMPLEMENTATION AND PERFROMANCE RESULTS

In this section, we first give a theoretical analysis of the performance of DPAEKS by comparing it with the existing related works [2], [5], [19], where the works [5] and [19] are recent research efforts that resulted in better performance to resist IKGA under the different frameworks (the single-server framework [5] and the dual-server framework [19]). Then, we implement a prototype of our scheme by using a popular cryptography library and evaluate its empirical performance.

A. Comparison

Computation Complexity: For simplicity, we define $T_{pairing}$ to denote the cost of a bilinear pairing $e : (\mathbb{G}_1, \mathbb{G}_2) \rightarrow \mathbb{G}_T$, T_{sm} to denote the cost of a scalar point multiplication operation $Q = rP \in \mathbb{G}_1$, T_H to denote the cost of a secure map-to-point hash function $H(\cdot) : \{0, 1\}^* \rightarrow \mathbb{G}_1$ and T_{pa} to denote the cost of a point addition operation $P_3 = P_1 + P_2 \in \mathbb{G}_1$. Some of the operations (e.g. the general hash operation or the integer arithmetic) are omitted because their time costs are negligible compared to the other operations. Table II presents the running time of the related operations, where the test elliptic-curve is a pairing-friendly curve[1] and we conducted the experiment on a personal computer[2]. As shown in Table II, the relationship of time costs of the above operations is $T_{pairing} \approx T_H > 2T_{sm} \gg T_{pa}$, thereby avoiding the use of the pairing operation or the map-to-point hash can be used to reduce the computational complexity of the SE schemes.

TABLE II. THE RUNNING TIME OF RELATED OPERATIONS

Operation	Running Time(millisecond)
$T_{pairing}$	6.594
T_H	6.487
T_{sm}	2.635
T_{pa}	0.014

Table III shows the number of the main operations that are required by these four schemes. Specifically, in order to generate a ciphertext, our scheme needs $5T_{sm}$ while the other three schemes need $1T_{pairing} + 2T_H + 2T_{sm}$, $1T_H + 4T_{sm}$ and $1T_H + 3T_{sm}$, respectively. For the trapdoor generation, although these computation-intensive operations $(T_{pairing}, T_H)$ are not used by our scheme, $5T_{sm}$ results in no advantage over other schemes. In the testing phase, our scheme is more efficient than other schemes, requiring only $4T_{sm} + 3T_{pa}$ without the pairing computation and map-to-point hash operation. Although the testing of Chen *et al.*'s scheme [19] does not involve these computation-intensive operations, the computation cost of their scheme is slightly higher than that of our scheme, requiring $7T_{sm}+3T_{pa}$. In addition, both ciphertext size and trapdoor size are very important indicator to evaluate the communication costs. For Boneh *et al.*'s scheme, Chen *et al.*'s scheme, Huang *et al.*'s scheme and our scheme, their communication costs of transferring a ciphertext, from the DO to the CS, are $2|\mathbb{G}_1|$, $3|\mathbb{G}_1|$, $2|\mathbb{G}_1|$ and $3|\mathbb{G}_1|$, while their communication costs of transferring a trapdoor, from the DR to the CS, are $|\mathbb{G}_1|$, $3|\mathbb{G}_1|$, $|\mathbb{G}_2|$ and $2|\mathbb{G}_1|$. In such settings, we have $|\mathbb{G}_1| = 512$ and $|\mathbb{G}_2| = 1536$. We observe that the communication costs of both transferring a ciphertext and a trapdoor in Boneh *et al.*'s scheme are less than that in the other three schemes, and our scheme is slightly better than Chen *et al.*'s scheme and Huang *et al.*'s scheme in terms of communication costs of transferring a trapdoor.

By combining the Table II and Table III, the Fig.3,4,5 respectively show the time costs of the **PEKS**, **Trapdoor** and **Test** algorithms with respect to the number of keywords. Firstly, we notice that the time costs of each algorithm for all schemes are linear with the number of keywords. Secondly, the time costs of both the **PEKS** algorithm and **Test** algorithm in our scheme are less than that in other schemes. The reason is that our scheme avoids the computation-intensive operations without introducing too many scalar point multiplication operations. In addition, our scheme will become more efficient than other schemes with the increasing number of keywords, which makes it more scalable in practical applications. Finally, the time cost of trapdoor generation in our scheme is slightly higher than that of Boneh *et al.*'s scheme, but significantly better than that of Huang *et al.*'s scheme and Chen *et al.*'s scheme.

Security Features: Table IV shows the features of each scheme, including mainly the IND-$Ciphertext$, IND-$Trapdoor$, $IKGA$, Aut, NS. We assume that IND-

[1]Barreto-Naehrig (BN) curve over a prime field of size ($\approx$ 256 bits)

[2]Intel Core i5 CPU 2.50GHz, 8.00GB RAM, a 250GB, Lenovo T410 running Windows 7

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2945714, IEEE Transactions on Cloud Computing

TABLE III. COMPUTATION AND COMMUNICATION COST

Scheme	Computation Costs			Communication Costs	
	PEKS *	Trapdoor	Test**	Size Per Ciphertext	Size Per Trapdoor
Boneh *et al.* [2]	$1T_{pairing} + 2T_H + 2T_{sm}$	$1T_H + 1T_{sm}$	$1T_{paring} + 1T_H$	$2\|\mathbb{G}_1\|$	$\|\mathbb{G}_1\|$
Chen *et al.* [19]	$1T_H + 4T_{sm}$	$1T_H + 4T_{sm}$	$7T_{sm} + 3T_{pa}$	$3\|\mathbb{G}_1\|$	$3\|\mathbb{G}_1\|$
Huang *et al.* [5]	$1T_H + 3T_{sm}$	$1T_{pairing} + 1T_H + 1T_{sm}$	$2T_{pairing}$	$2\|\mathbb{G}_1\|$	$\|\mathbb{G}_2\|$
Our	$5T_{sm}$	$5T_{sm}$	$4T_{sm} + 3T_{pa}$	$3\|\mathbb{G}_1\|$	$2\|\mathbb{G}_1\|$

* PEKS denotes the process of generating a keyword ciphertext.

** Test includes both the transition and the test process in the dual-server framework.

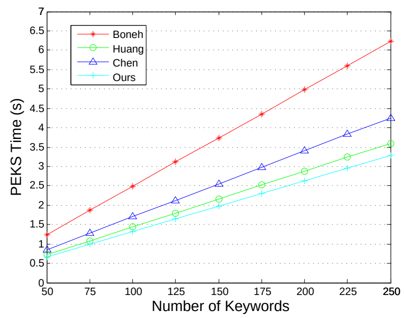

Fig. 3. Computation cost of PEKS phase

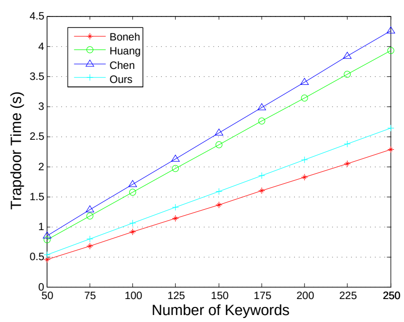

Fig. 4. Computation cost of trapdoor phase

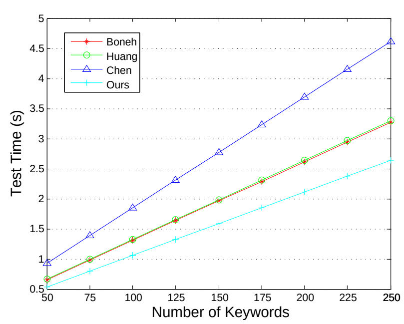

Fig. 5. Computation cost of test phase

TABLE IV. FEATURES.

Scheme	Features of Various Schemes				
	IND-Ciphertext	IND-Trapdoor	IKGA	Aut	NS
Boneh *et al.* [2]	√	×	×	√	1
Chen *et al.* [19]	√	√	√	×	2
Huang *et al.* [5]	×	×	√	√	1
Our	√	√	√	√	2

√ It denotes the scheme supports the corresponding feature.
× denotes the scheme fails in supporting the corresponding feature.

$Ciphertext$ means that the ciphertexts of keywords are indistinguishable, IND-$Trapdoor$ means that the trapdoors of keywords are indistinguishable, $IKGA$ means that the scheme can resist the IKGA, Aut means that only the authenticated user can search the encrypted data, and NS is that the number of servers used in a scheme. Compared with the single-server schemes [2], [5], our scheme achieves the ciphertext indistinguishability, trapdoor indistinguishability and resisting the inside keyword guessing attack by leveraging the dual-server framework. Compared with another similar dual-server scheme [19], our scheme supports the authentication without an additional authentication mechanism, which is used to prevent the unauthenticated user from accessing the data.

To sum up, our scheme has high computational efficiency, especially the **PEKS** and **Test** algorithms. This is mainly because the scheme avoids the use of computation-intensive operations. Meanwhile, the dual-server framework makes our scheme resistant to keyword guessing attacks launched by the outside and inside adversaries. It should be noted that resisting IKGA is achieved at the expense of communication overhead since the test is completed by interacting with both servers.

TABLE V. THE ELLIPTIC CURVES

Domain	Base Field Size (bit)	Security Level*
MNT	160**	80-bit
Secp256k1	256***	128-bit

* It is equal to the key bit-length of symmetric cipher (e.g. AES)
** 0xAD1FDFB99D7D159240B91083F3A869072D3AE145
*** 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFFC2F

B. Performance

To evaluate the performance of DPAEKS, we implemented our scheme and Chen *et al.*'s scheme [19] under different elliptic curves (as shown in Table V). We executed all experiment algorithms on the Windows platform and used the *MIRACL*

C++ library[3] to implement the related operations. We used the real Enron Email Dataset[4](about 423MB), which includes data from about 150 managers. We extracted about 7000 high frequency keywords (the length of the keyword is greater than 5, and the frequency of occurrence is greater than 20).

The configurations of the software and hardware are as follows: the client side (including DR and DO) is a personal computer with Intel core i5 CPU 2.40GHz, 4.00GB RAM, 500GB hard disk, Lenovo T440 running Windows 7. It generates the encrypted data and trapdoor of the keyword. The server side is a Windows 10 Dell desktop system with an Intel Core i7 CPU 3.4GHz, 8GB RAM and 1T hard disk. It is responsible for searching over encrypted data.

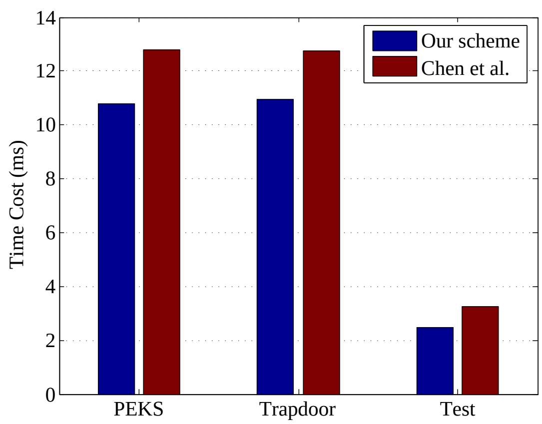

Fig. 6. The average time cost of every algorithm

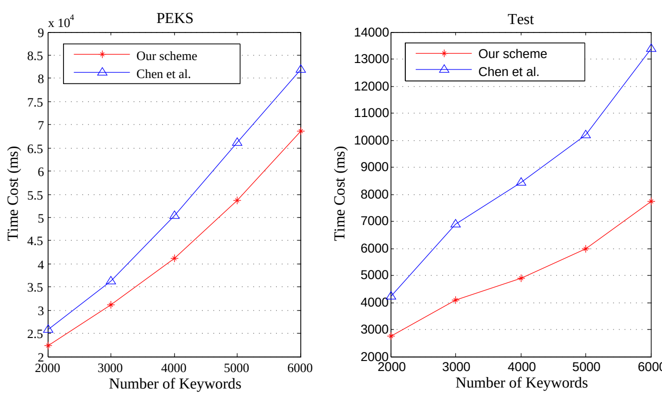

Fig. 7. The time cost of each algorithm with different number of keywords (MNT)

First, we choose the same elliptic curve as the Chen *et al.*'s scheme (MNT curve, which supports the bilinear pairing operation). Fig.6 shows the average time cost to generate one ciphertext, one trapdoor and complete the test process by Chen *et al.*'s scheme and our scheme. On the client side, our scheme only needs 10.7 ms, 10.9 ms to encrypt one keyword and generate one trapdoor respectively while Chen *et al.*'s scheme takes 12.8 ms, 12.7ms. Our scheme saves approximately 16% and 14% time cost to encrypt one keyword and generate one trapdoor respectively. On the server side, the average time cost to test one ciphertext in Chen *et al.*'s scheme is approximately 3.256 ms while our scheme only takes 2.478 ms. Moreover, when the number of keywords increases, our scheme saves more time costs compared to Chen *et al.*'s scheme for generating the encrypted dataset and finding a matching ciphertext, as shown in Fig. 7. Hence, our scheme is considerably more efficient in terms of computation cost to encrypt the dataset and search the matching ciphertext than Chen *et al.*'s scheme.

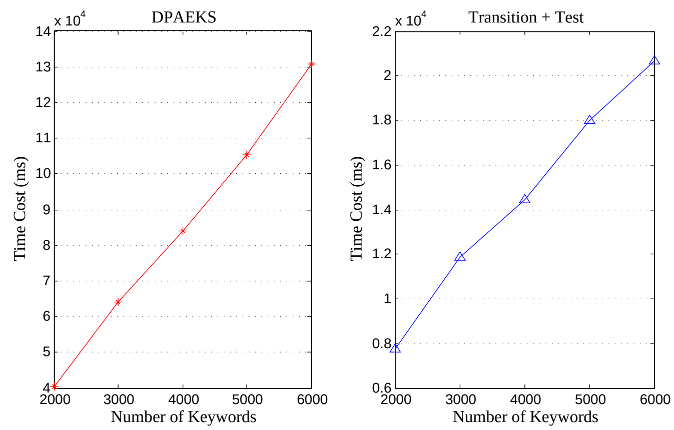

Fig. 8. The time cost of every algorithm with different number of keywords (Secp256K1)

To simulate different application environments, we also choose another elliptic curve with a higher security level, called Secp256k1 curve. Fig.8 shows the performance of our PEKS algorithm and test algorithm (including the transition process). According to the experimental results, we find that the average time cost to generate one ciphertext and test one ciphertext is approximately 17.01 ms and 2.9 ms respectively. In addition, like most existing PEKS schemes (e.g. [5], [19]), the time costs of the encryption algorithm and the test algorithm are both linear with the number of keywords.

VI. CONCLUSION

Combining protection against IKGA and efficiency is not trivial because the two properties are irreconcilable. In this paper, we have presented a new scheme called dual-server public-key authenticated encryption with keyword search (DPAEKS). The features of DPAEKS include: two non-colluding servers that are used to protect against IKGA and the data owner should be distributed with a pair of keys to authenticate the data. We developed a concrete construction of DPAEKS and proved its security. Finally, we implemented and evaluated the performance of the proposed scheme. The empirical results we obtained demonstrate that it is suitable for deployment in practical applications.

[3]https://www.miracl.com/
[4]http://www.cs.cmu.edu/~/enron/

ACKNOWLEDGMENT

We thank the anonymous reviewers and the editor for their valuable comments which helped us to improve the content and presentation of this paper.

REFERENCES

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

[2] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Eurocrypt*, vol. 3027. Springer, 2004, pp. 506–522.

[3] P. Xu, H. Jin, Q. Wu, and W. Wang, "Public-key encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack," *IEEE Transactions on computers*, vol. 62, no. 11, pp. 2266–2277, 2013.

[4] R. Chen, Y. Mu, G. Yang, F. Guo, and X. Wang, "A new general framework for secure public key encryption with keyword search," in *Australasian Conference on Information Security and Privacy*. Springer, 2015, pp. 59–76.

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

[6] D. Wang, N. Wang, P. Wang, and S. Qing, "Preserving privacy for free: Efficient and provably secure two-factor authentication scheme with user anonymity," *Information Sciences*, vol. 321, pp. 162–178, 2015.

[7] C.-h. Wang and T.-y. Tu, "Keyword search encryption scheme resistant against keyword-guessing attack by the untrusted server," *Journal of Shanghai Jiaotong University (Science)*, vol. 19, no. 4, pp. 440–442, 2014.

[8] J. Baek, R. Safavi-Naini, and W. Susilo, "Public key encryption with keyword search revisited," *Computational Science and Its Applications–ICCSA 2008*, pp. 1249–1259, 2008.

[9] H. S. Rhee, J. H. Park, W. Susilo, and D. H. Lee, "Improved searchable public key encryption with designated tester," in *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*. ACM, 2009, pp. 376–379.

[10] K. Emura, A. Miyaji, M. S. Rahman, and K. Omote, "Generic constructions of secure-channel free searchable encryption with adaptive security," *Security and communication networks*, vol. 8, no. 8, pp. 1547–1560, 2015.

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

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

[13] H. S. Rhee, W. Susilo, and H.-J. Kim, "Secure searchable public key encryption scheme against keyword guessing attacks," *IEICE Electronics Express*, vol. 6, no. 5, pp. 237–243, 2009.

[14] L. Fang, W. Susilo, C. Ge, and J. Wang, "Public key encryption with keyword search secure against keyword guessing attacks without random oracle," *Information Sciences*, vol. 238, pp. 221–241, 2013.

[15] L. Guo and W.-C. Yau, "Efficient secure-channel free public key encryption with keyword search for emrs in cloud storage," *Journal of medical systems*, vol. 39, no. 2, p. 11, 2015.

[16] A. Arriaga, Q. Tang, and P. Ryan, "Trapdoor privacy in asymmetric searchable encryption schemes," in *International Conference on Cryptology in Africa*. Springer, 2014, pp. 31–50.

[17] Y. Lu, G. Wang, and J. Li, "Keyword guessing attacks on a public key encryption with keyword search scheme without random oracle and its improvement," *Information Sciences*, vol. 479, pp. 270–276, 2019.

[18] I. R. Jeong, J. O. Kwon, D. Hong, and D. H. Lee, "Constructing peks schemes secure against keyword guessing attacks is possible?" *Computer communications*, vol. 32, no. 2, pp. 394–396, 2009.

[19] 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, 2016.

[20] 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, 2016.

[21] X. Huang, Y. Xiang, A. Chonka, J. Zhou, and R. H. Deng, "A generic framework for three-factor authentication: Preserving security and privacy in distributed systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 8, pp. 1390–1397, 2010.

[22] L. Wu, B. Chen, K.-K. R. Choo, and D. He, "Efficient and secure searchable encryption protocol for cloud-based internet of things," *Journal of Parallel and Distributed Computing*, vol. 111, pp. 152–161, 2018.

[23] X.-F. Wang, Y. Mu, R. Chen, and X.-S. Zhang, "Secure channel free id-based searchable encryption for peer-to-peer group," *Journal of Computer Science and Technology*, vol. 31, no. 5, pp. 1012–1027, 2016.

[24] Y. Yang and M. Ma, "Conjunctive keyword search with designated tester and timing enabled proxy re-encryption function for e-health clouds," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 4, pp. 746–759, 2016.

[25] 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, 2018.

[26] X. Song, C. Dong, D. Yuan, Q. Xu, and M. Zhao, "Forward private searchable symmetric encryption with optimized i/o efficiency," *IEEE Transactions on Dependable and Secure Computing*, 2018.

[27] G. Di Crescenzo and V. Saraswat, "Public key encryption with searchable keywords based on jacobi symbols," in *International Conference on Cryptology in India*. Springer, 2007, pp. 282–296.

[28] P. Xu, S. He, W. Wang, W. Susilo, and H. Jin, "Lightweight searchable public-key encryption for cloud-assisted wireless sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3712–3723, 2018.

[29] P. Xu, Q. Wu, W. Wang, W. Susilo, J. Domingo-Ferrer, and H. Jin, "Generating searchable public-key ciphertexts with hidden structures for fast keyword search," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 9, pp. 1993–2006, 2015.

[30] D. Wang, H. Cheng, P. Wang, X. Huang, and G. Jian, "Zipfs law in passwords," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2776–2791, 2017.

[31] D. Wang and P. Wang, "Two birds with one stone: Two-factor authentication with security beyond conventional bound," *IEEE transactions on dependable and secure computing*, vol. 15, no. 4, pp. 708–722, 2016.

Biwen Chen received his M.S. degree from Hubei University of Technology of China in 2016. He is currently pursing his Ph.D. degree at Computer School, Wuhan University, Wuhan, China. His main research interests include cryptography and information security, in particular, cryptographic protocols.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2945714, IEEE Transactions on Cloud Computing

Libing Wu was born in 1972. He received the B.S. and M.S. degrees in computer science from Central China Normal University, Wuhan, China, in 1994 and 2001, respectively. He received his Ph.D. degree in computer science from Wuhan University in 2006. He is currently a professor of the School of Cyber Science and Engineering, Wuhan University. He is a senior member of IEEE and CCF. His areas of research interests include distributed computing, trusted software and wireless sensor networks.

Sherali Zeadally is an Associate Professor in the College of Communication and Information at the University of Kentucky. He received the Bachelor and Doctorate degrees in computer science from the University of Cambridge, England, and the University of Buckingham, England, respectively. He is a Fellow of the British Computer Society and the Institution of Engineering Technology, England.

Debiao He received his Ph.D. degree in applied mathematics from School of Mathematics and Statistics, Wuhan University in 2009. He is currently a professor of the School of Cyber Science and Engineering, Wuhan University. His main research interests include cryptography and information security, in particular, cryptographic protocols.