Identity-certifying Authority-aided Identity-based Searchable Encryption Framework in Cloud System

Zi-Yuan Liu, Yi-Fan Tseng, Raylin Tso, Yu-Chi Chen, and Masahiro Mambo

***Abstract*—In the era of cloud computing, massive quantities of data are encrypted and uploaded to the cloud to realize a variety of applications and services while protecting user confidentiality. Accordingly, the formulation of methods for efficiently searching encrypted data has become a critical problem. Public-key encryption with keyword search is an efficient solution that allows the data owner to generate encrypted keywords for a given document while also allowing the data user to generate the corresponding trapdoor for searching. Huang and Li proposed a public-key authenticated encryption with keyword search (PAEKS) scheme to resist keyword guessing attacks, where the data owner not only encrypts keywords but also authenticates them.However, existing PAEKS-related schemes carry a trade-off between efficiency, storage cost, and security.In this paper, we introduce a novel framework, called identity-certifying authority-aided identity-based searchable encryption, which has the advantage of reducing storage space while remaining the efficiency and security.We formally define the system model and desired security requirements to represent attacks in a real scenario. In addition, we propose a provably secure scheme based on the gap bilinear Diffie–Hellman assumption and experimentally evaluate our scheme in terms of its performance and theoretical features against its state-of-the-art counterparts.**

***Index Terms*—identity-certifying authority, cloud system, identity-based encryption, keyword search.**

I. Introduction

With the maturation of cloud computing technology, enterprises have increasingly uploaded massive quantities of data to the cloud to reduce their storage and computing burden. For example, convenience store chains upload data from each branch to the cloud for analysis, and hospitals upload patient data to the cloud for management. In addition, since the introduction of the concept of Industry 4.0 by Lasi *et al.* [1], firms have begun integrating cloud systems into their data collection and production processes. However, privacy concerns remain. Data on shipments, patient records, and even factory inventory are highly sensitive, and companies may be reluctant to directly upload them to a cloud system that they cannot fully trust. Consequently, data are often encrypted before being uploaded to a cloud system in order to avoid information leakage, but such encrypted data pose a computational challenge for cloud systems.

Z.-Y. Liu, Y.-F. Tseng, and R. Tso are with the Department of Computer Science, National Chengchi University, Taipei 11605, Taiwan.
E-mail: {zyliu, yftesng, raylin}@cs.nccu.edu.tw
Yu-Chi Chen is with the Department of Computer Science and Engineering, Yuan Ze University, Taoyuan 32000, Taiwan
E-mail: wycchen@saturn.yzu.edu.tw
Masahiro Mambo is with the Institute of Science and Engineering, Kanazawa University, Kanazawa 920-1192, Japan.
E-mail: mambo@ec.t.kanazawa-u.ac.jp
Corresponding author: R. Tso

Symmetric searchable encryption (SSE), introduced by Song *et al.* [2], is one solution to the aforementioned problem. In SSE, a data owner (DO) can generate encrypted keywords for each encrypted file by using a symmetric key that is shared with the data user (DU) before their data are uploaded to the cloud system. Subsequently, the DU can generate a trapdoor for specified keywords and submit them to the cloud system to search for encrypted files that are related to these keywords. Because of these properties, SSE is well suited to cloud computing, and various SSE approaches have been proposed [3]–[6]. However, SSE is restricted to the key sharing problem of symmetric cryptosystems. Specifically, the DO and DU must agree on a shared key before encrypting keywords and generating trapdoors, respectively.

To further increase the range of application and reduce the communication overhead of negotiating keys, Boneh *et al.* [7] introduced a searchable encryption method in a public-key setting, called public key encryption with keyword search (PEKS). Instead of using a shared key as done in SSE, in PEKS, the DO encrypts keywords by using the DU's public key, and the DU generates corresponding trapdoors by using his or her secret key. After the pioneering work by Boneh *et al.,* PEKS immediately caught the attention of researchers, and many studies have applied PEKS to various applications [8]–[12]. However, in 2006, Byun *et al.* [13] observed that because the entropy of keywords is low, any malicious party, through a so-called keyword guessing attack, can randomly select keywords to generate the ciphertext and test whether the ciphertext is passable; thus, the malicious party can obtain the information associated with the keywords in the trapdoor. In particular, to resolve the keyword guessing attack launched by a malicious cloud server (CS) as part of a so-called *insider* keyword guessing attack (IKGA), Chen *et al.* [14]–[16] have first proposed solutions for dual-server setting, and their methods were improved upon by Tso *et al.* [17].

Huang and Li [18] recently introduced the concept of public authenticated encryption with keyword search (PAEKS) under a single-server setting, where the trapdoor works only for ciphertext that is authenticated by the DO using his or her secret key; therefore, a malicious CS cannot randomly generate ciphertext and further perform IKGA. Inspired by Huang and Li's work [18], scholars have proposed several PAEKS schemes [19]–[22]. However, because PAEKS schemes are based on public key settings and do not provide implicit authentication, they require a trusted public key infrastructure to bind public keys with the respective identities of entities through issuing certificates; nevertheless, this process incurs additional overhead. Several approaches are available for intuitively solving this problem in PAEKS:

1) Identity-based authenticated encryption with keyword search (IBAEKS) [23]: In IBAEKS schemes, the user can apply to a trusted key generation center (KGC) for the secret key of a specific identity; they can then use the identity as their public key. Thus, in IBAEKS, no additional steps are required to prove the validity of the public key.
2) Certificate-based authenticated encryption with keyword search (CBAEKS) [24], [25]: In CBAEKS schemes, the user can apply to a trusted certification authority for the certificate of a pair comprising a specific identity and public key. Unlike the certificates used in PAEKS schemes, the certificate in CBAEKS not only implicitly authenticates the validity of the identity and public key but also acts as a partial secret key.
3) Certificateless authenticated encryption with keyword search (CLAEKS) [26]–[28]: The solution of CLAEKS is similar to that of IBAEKS, except that the user generates a pair comprising public and secret values, in addition to obtaining the secret key of their identity from the KGC. The user finally uses the identity and public value as the full public key and uses the secret key and secret value as the full secret key.

Although the three aforementioned approaches solve the problem in PAEKS, each approach has its disadvantages in safety, storage, or efficiency. Specifically, in IBAEKS [23], because the KGC can gain access to any user's secret key, key escrow problems occur when the KGC is malicious. By contrast, although CBAEKS [24], [25] and CLAEKS [26]–[28] solve the key escrow problem, the users in these schemes require an additional generated certificate and a pair comprising a public and secret value. The larger key size in these schemes also entails larger ciphertext and trapdoor sizes. Consequently, users require more storage space to store keys, certificates, ciphertext, and trapdoors in CBAEKS and CLAEKS than they do in PAEKS and IBAEKS. Therefore, existing methods carry a three-way trade-off between efficiency, storage cost, and security.

Accordingly, to ameliorate this trade-off, we present a novel identity-certifying authority (ICA)-aided identity-based searchable encryption (IBSE) framework (hereafter referred to as ICA-IBSE). Our contributions are summarized as follows:

- To master the trade-off between efficiency and security, on the basis of Chow *et al.* [29] and Emura *et al.* [30], who have used an identity-certifying authority in identity-based encryption, we propose our ICA-IBSE scheme, which inherits the advantages of IBAEKS in terms of convenience and storage requirements and eliminates the disadvantage of IBAEKS regarding the key escrow problem.
- We define the system model and security requirements of the ICA-IBSE framework before applying it to a practical case. Moreover, we provide the security proofs to show that under the defined security models, our scheme is secure if the gap bilinear Diffie–Hellman (GBDH) assumption holds.

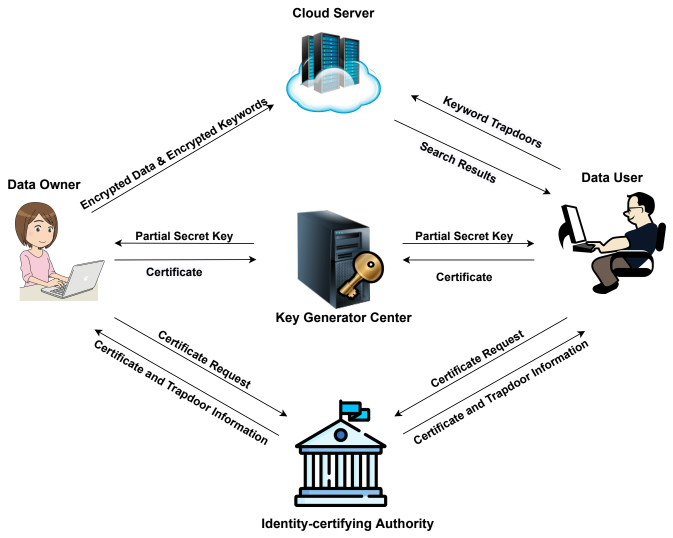

Fig. 1. The proposed ICA-IBSE framework

- We further provide a theoretical comparison and performance evaluation of our scheme against state-of-the-art schemes [20], [21], [23]–[26], [28]. The results indicate that our scheme is more efficient and that it performs better at reducing the storage requirement for the public key, ciphertext, and trapdoor.

II. Problem Formulation

A. System Model

The ICA-IBSE system model comprises five entities, namely the ICA, KGC, CS, DO, and DU, which are displayed in Fig. 1.

- ICA: This authority is responsible for validating the DO's and DU's identities and issuing trapdoor information and an identity certificate to them.
- KGC: By validating the correctness of the identity certificate, the KGC generates the DO's and DU's partial secret key without knowing any information about their identity.
- CS: The CS has sufficient storage and computing capacity and is mainly responsible for storing encrypted data along with the corresponding encrypted keywords and searching for data.
- DO: The DO first requests their certificate from the ICA and further generates their partial secret key by interacting with the KGC. Finally, the DO obtains their secret key by using trapdoor information. The DO can generate massive quantities of encrypted data along with the corresponding encrypted keywords, which are uploaded to the CS to reduce the storage requirement.
- DU: The DU generates their secret key as the DO does. Subsequently, the DU can issue trapdoors, generated by using their secret key, to the CS to retrieve the encrypted data associated with the specified keyword.

B. Adversary Model

In the following analysis, we consider adversary threats from different perspectives. We assume that the DO and DU are fully trusted, as is the case with most PEKS schemes. The DO and DU cannot collude with other parties to reveal their secret keys. In addition, we assume that the ICA and KGC cannot collude with each other to generate the DU's and DO's secret keys.

- The KGC and CS are honest but curious, which means that they will attempt to retrieve the sensitive information of the keywords from encrypted keywords and trapdoors.
- The ICA is malicious. In addition to attempting to obtain keyword information as the KGC and CS do, the ICA can generate a potentially malicious ICA key pair.
- The communication channels between the DO and DU and the cloud server are insecure, which means that all transmitted information is eavesdropped upon by any one party (*i.e.,* ICA, KGC, and a malicious outsider). However, the communication channels between the DO and DU and the ICA and KGC are secure. In other words, we assume that the communication channels are encrypted and authenticated (*e.g.,* by using TLS 1.3 [31]).

III. Definition and Security Models

A. ICA-IBSE Framework

A typical ICA-IBSE scheme comprises seven algorithms and a protocol, which are described as follows:

$\mathsf{Setup}(1^\lambda)$: The security parameter λ is taken as the input, and the system parameter pp is the output. Here, we assume that the keyword space $\mathcal{W}$ and the identity space $\mathcal{ID}$ are defined by the system parameter pp.

$\mathsf{ICA\text{-}Setup}(pp)$: The system parameter pp is taken as the input, and the public–secret key pair (pk_{ICA}, sk_{ICA}) of the ICA is the output.

$\mathsf{KGC\text{-}Setup}(pp)$: The system parameter pp is taken as the input, and the public–secret key pair (pk_{KGC}, sk_{KGC}) of the KGC is the output.

$\mathsf{ICA\text{-}Cert}(pp, pk_{ICA}, sk_{ICA}, ID)$: The system parameter pp, ICA's public key pk_{ICA}, ICA's secret key sk_{ICA}, and a user's identity $ID \in \mathcal{ID}$ are taken as the inputs, and a certificate $cert_{ID}$ and a piece of trapdoor information tf_{ID} are the outputs, which are sent to the ID through a secure channel.

$\langle\mathsf{User\text{-}Obtain\text{-}Key}(pp, pk_{KGC}, ID, cert_{ID}, tf_{ID}),$ $\mathsf{KGC\text{-}Issue\text{-}Key}(pp, pk_{KGC}, sk_{KGC}, pk_{ICA})\rangle$: This is an interactive key-issuing protocol between a user and the KGC that comprises two algorithms: User-Obtain-Key and KGC-Issue-Key. The user first generates the first-round message $\mathsf{M}_1 \leftarrow$ User-Obtain-Key and submits it to the KGC. Subsequently, the KGC returns a second-round message $\mathsf{M}_2 \leftarrow$ KGC-Issue-Key to the user. At the end of the protocol, the user can locally output a secret key sk_{ID} or $\perp$.

$\mathsf{Encrypt}(pp, sk_{DO}, DU, w)$: The system parameter pp, DO's secret key sk_{DO}, DU's identity DU, and a keyword $w \in \mathcal{W}$ are taken as the inputs, and a searchable ciphertext ct_w associated with keyword w is the output.

$\mathsf{Trapdoor}(pp, DO, sk_{DU}, w)$: The system parameter pp, DO's identity DO, DU's secret key sk_{DU}, and a keyword $w \in \mathcal{W}$ are taken as the inputs, and a trapdoor td_w associated with keyword w is the output.

$\mathsf{Test}(pp, ct_w, td_w)$: The system parameter pp, a searchable ciphertext ct_w, and a trapdoor td_w are taken as the inputs, and 1 is the output if ct_w is matched with td_w; otherwise, 0 is the output.

Definition 1 (Correctness and Consistency of ICA-IBSE)**.** *For all security parameters $\lambda \in \mathbb{N}$, all DOs $DO \in \mathcal{ID}$, all DUs $DU \in \mathcal{ID}$, and all keywords $w, w' \in \mathcal{W}$, ICA-IBSE is defined to be correct if, when $w = w'$, we have*

$$\Pr[\mathsf{Test}(pp, ct_w, td_{w'}) = 1] = 1$$

and ICA-IBSE is defined to be consistent if, when $w \neq w'$, we have

$$\Pr[\mathsf{Test}(pp, ct_w, td_{w'}) = 0] = 1 - \mathsf{negl}(\lambda),$$

where $pp \leftarrow \mathsf{Setup}(1^\lambda)$; $(pk_{ICA}, sk_{ICA}) \leftarrow \mathsf{ICA\text{-}Setup}(pp)$; $(pk_{KGC}, sk_{KGC}) \leftarrow \mathsf{KGC\text{-}Setup}(pp)$; $(cert_i, td_i) \leftarrow \mathsf{ICA\text{-}Cert}(pp, pk_{ICA}, sk_{ICA}, i)$; and $sk_i \leftarrow \langle\mathsf{User\text{-}Obtain\text{-}Key}(pp, pk_{KGC}, i, cert_i, td_i),$ $\mathsf{KGC\text{-}Issue\text{-}Key}(pp, pk_{KGC}, sk_{KGC}, pk_{ICA})\rangle$, for $i = \{DO, DU\}$, $ct_w = \mathsf{Encrypt}(pp, sk_{DO}, DU, w)$, and $td_{w'} \leftarrow \mathsf{Trapdoor}(pp, DO, sk_{DU}, w')$.

B. Security Models

To model the different aspects of the attacks described in Section II-B, we revise the security model in [32] and [21] to account for multichosen keyword attacks (MCKAs) and IKGAs. MCKAs, recently introduced by Qin *et al.* [21], ensure that no adversary can obtain any information from two tuples of encrypted keywords; IKGAs ensure that no insider adversary (*i.e.,* the CS) can obtain any information on keywords from the trapdoor, even when an insider can conduct tests.

Before presenting our security models, we first define the following oracles that are simulated by the challenger for the adversary:

Certificate Oracle $\mathcal{O}_{\mathsf{cert}}$: For any identity $ID \in \mathcal{ID}$, the challenger outputs $(cert_{ID}, tf_{ID}) \leftarrow \mathsf{ICA\text{-}Cert}(pp, pk_{ICA}, sk_{ICA}, ID)$ to the adversary.

Secret Key Oracle $\mathcal{O}_{\mathsf{sk}}$: For any first-round message M_1, the challenger runs $\mathsf{M}_2 \leftarrow \mathsf{KGC\text{-}Issue\text{-}Key}(pp, pk_{KGC}, sk_{KGC}, pk_{ICA})$ and returns M_2 to the adversary.

Ciphertext Oracle $\mathcal{O}_{\mathsf{ct}}$: For any keyword $w \in \mathcal{W}$, the challenger outputs $ct_w \leftarrow \mathsf{Encrypt}(pp, sk_{DO}, DU, w)$ to the adversary.

Trapdoor Oracle $\mathcal{O}_{\mathsf{td}}$: For any keyword $w \in \mathcal{W}$, the challenger outputs $td_w \leftarrow \mathsf{Trapdoor}(pp, DO, sk_{DU}, w)$ to the adversary.

Issue Key KGC Oracle $\mathcal{O}_{\mathsf{ik\text{-}KGC}}$: When the adversary makes this query, the challenger randomly chooses $ID \leftarrow \mathcal{ID}$ and computes $(cert_{ID}, tf_{ID}) \leftarrow \mathsf{ICA\text{-}Cert}(pp, pk_{ICA}, sk_{ICA}, ID)$. In addition, it runs $\mathsf{M}_1 \leftarrow \mathsf{User\text{-}Obtain\text{-}Key}(pp, pk_{KGC}, ID, cert_{ID}, tf_{ID})$, and returns M_1 to the adversary, stores ID to $IDList$, and updates $Q_{key} \leftarrow Q_{key}+1$. Here, Q_{key} is the number of instances $\mathcal{O}_{\text{ik-KGC}}$ is queried, and $IDList$ is a list of identities whose corresponding M_1 values have been obtained by the adversary through querying $\mathcal{O}_{\text{ik-KGC}}$.

Ciphertext KGC Oracle $\mathcal{O}_{\text{ct-KGC}}$: For any keyword $w \in \mathcal{W}$, DO index doi, and DU index dui, the challenger first checks whether $doi \in [Q_{key}]$ and $dui \in [Q_{key}]$. If not, the challenger forces the adversary to output a random bit $b' \in \{0,1\}$. Otherwise, the challenger retrieves the doi-th identity's secret key $sk_{IDList[doi]}$ and dui-th identity $IDList[dui]$ in $IDList$ and subsequently outputs $ct_w \leftarrow \mathsf{Encrypt}(pp, sk_{IDList[doi]}, IDList[dui], w)$ to the adversary.

Trapdoor KGC Oracle $\mathcal{O}_{\text{td-KGC}}$: For any keyword $w \in \mathcal{W}$, DO index doi, and DU index dui, the challenger first checks whether $doi \in [Q_{key}]$ and $dui \in [Q_{key}]$. If not, the challenger forces the adversary to output a random bit $b' \in \{0,1\}$. Otherwise, the challenger retrieves the doi-th identity $IDList[doi]$ and dui-th identity's secret key $sk_{IDList[dui]}$ and subsequently outputs $td_w \leftarrow \mathsf{Trapdoor}(pp, IDList[doi], sk_{IDList[dui]}, w)$ to the adversary.

The CS can directly obtain the entirety of ciphertext and all trapdoors from the DO and DU individually; hence, intuitively, the CS has greater attack capability relative to eavesdroppers on the channel. Therefore, we only consider attacks from the CS and not from parties other than the CS, ICA, or KGC. We formulate six games (MCKA-CS, MCKA-KGC, MCKA-ICA, IKGA-CS, IKGA-KGC, and IKGA-ICA) to model the MCKA or IKGA from the CS, KGC, or ICA. In this paper, MCKA and IKGA games are represented using blue solid and red dotted lines, respectively.

First, in the MCKA-CS and IKGA-CS games, the adversary is restricted to issuing queries to $\mathcal{O}_{\text{cert}}$ on some challenged identities $DO, DU \in \mathcal{ID}$. Furthermore, the adversary cannot issue queries to oracles $\mathcal{O}_{\text{ct}}$ on the challenged keywords $w_{b,i} \in \mathcal{W}$ and cannot issue queries to oracles $\mathcal{O}_{\text{td}}$ on the challenged keywords $w_b \in \mathcal{W}$ for the challenged identities $DO, DU \in \mathcal{ID}$, where $b \in \{0,1\}$ and $i \in \{1, \cdots, n\}$. We consider that ICA-IBSE is MCKA-CS secure if the advantage

$$\mathsf{Adv}^{\mathsf{MCKA\text{-}CS}}_{\mathit{ICA\text{-}IBSE}}(\lambda) = |\Pr[b = b'] - 1/2|$$

is negligible for any adversary, and we consider that ICA-IBSE is IKGA-CS secure if the advantage

$$\mathsf{Adv}^{\mathsf{IKGA\text{-}CS}}_{\mathit{ICA\text{-}IBSE}}(\lambda) = |\Pr[b = b'] - 1/2|$$

is negligible for any adversary.

MCKA-CS / IKGA-CS Game

```
$pp \leftarrow \mathsf{Setup}(1^\lambda)$; random $b \in \{0,1\}$;
$(pk_{ICA}, sk_{ICA}) \leftarrow \mathsf{ICA\text{-}Setup}(pp)$; $(pk_{KGC}, sk_{KGC}) \leftarrow \mathsf{KGC\text{-}Setup}(pp)$;
$\tilde{w_0} = (w_{0,1}, \cdots, w_{0,n}), \tilde{w_1} = (w_{1,1}, \cdots, w_{1,n}), DO, DU \leftarrow \mathcal{A}^{\mathcal{O}_{\text{cert}}(\cdot),\mathcal{O}_{\text{sk}}(\cdot),\mathcal{O}_{\text{ct}}(\cdot),\mathcal{O}_{\text{td}}(\cdot)}(pp, pk_{ICA}, pk_{KGC})$;
$\tilde{ct}^* = (ct_1^*, \cdots, ct_n^*)$, where $ct_i^* \leftarrow \mathsf{Encrypt}(pp, sk_{DO}, DU, w_{b,i})$, for $i = 1, \cdots, n$;
$w_0, w_1, DO, DU \leftarrow \mathcal{A}^{\mathcal{O}_{\text{cert}}(\cdot),\mathcal{O}_{\text{sk}}(\cdot),\mathcal{O}_{\text{ct}}(\cdot),\mathcal{O}_{\text{td}}(\cdot)}(pp, pk_{ICA}, pk_{KGC})$;
$td^* \leftarrow \mathsf{Trapdoor}(pp, DO, sk_{DU}, w_b)$;
$b' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{cert}}(\cdot),\mathcal{O}_{\text{sk}}(\cdot),\mathcal{O}_{\text{ct}}(\cdot),\mathcal{O}_{\text{td}}(\cdot)}(pp, pk_{ICA}, pk_{KGC})$;
```

In MCKA-ICA and IKGA-ICA games, we model a malicious ICA that can generate a potentially malicious ICA key pair (pk_{ICA}, sk_{ICA}). However, to model the state of affairs where the ICA cannot interact with the KGC, we restrict the ICA to access Secret Key Oracle $\mathcal{O}_{\text{sk}}$. In addition, the adversary is restricted to issue queries to oracles $\mathcal{O}_{\text{ct}}$ on challenged keywords $w_{b,i} \in \mathcal{W}$ and to issue queries to oracles $\mathcal{O}_{\text{td}}$ on challenged keywords $w_b \in \mathcal{W}$ for some challenged identities $DO, DU \in \mathcal{ID}$, where $b \in \{0,1\}$ and $i \in \{1, \cdots, n\}$. We consider that ICA-IBSE is MCKA-ICA secure if the advantage

$$\mathsf{Adv}^{\mathsf{MCKA\text{-}ICA}}_{\mathit{ICA\text{-}IBSE}}(\lambda) = |\Pr[b = b'] - 1/2|$$

is negligible for any adversary, and we consider that ICA-IBSE is IKGA-ICA secure if the advantage

$$\mathsf{Adv}^{\mathsf{IKGA\text{-}ICA}}_{\mathit{ICA\text{-}IBSE}}(\lambda) = |\Pr[b = b'] - 1/2|$$

is negligible for any adversary.

MCKA-ICA / IKGA-ICA Game

```
$pp \leftarrow \mathsf{Setup}(1^\lambda)$; random $b \in \{0,1\}$;
$(pk_{ICA}, sk_{ICA}) \leftarrow \mathsf{ICA\text{-}Setup}(pp)$; $(pk_{KGC}, sk_{KGC}) \leftarrow \mathsf{KGC\text{-}Setup}(pp)$;
$\tilde{w_0} = (w_{0,1}, \cdots w_{0,n}), \tilde{w_1} = (w_{1,1}, \cdots, w_{1,n}), DO, DU \leftarrow \mathcal{A}^{\mathcal{O}_{\text{ct}}(\cdot),\mathcal{O}_{\text{td}}(\cdot)}(pp, pk_{KGC})$;
$\tilde{ct}^* = (ct_1^*, \cdots, ct_n^*)$, where $ct_i^* \leftarrow \mathsf{Encrypt}(pp, sk_{DO}, DU, w_{b,i})$, for $i = 1, \cdots, n$;
$w_0, w_1, DO, DU \leftarrow \mathcal{A}^{\mathcal{O}_{\text{ct}}(\cdot),\mathcal{O}_{\text{td}}(\cdot)}(pp, pk_{KGC})$;
$td^* \leftarrow \mathsf{Trapdoor}(pp, DO, sk_{DU}, w_b)$;
$b' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{ct}}(\cdot),\mathcal{O}_{\text{td}}(\cdot)}(pp, pk_{KGC})$;
```

Finally, the MCKA-KGC and IKGA-KGC games differ from the preceding games in terms of their setups. Specifically, the adversary is given KGC's secret key. Moreover, we use Q_{key} to count the number of KGC queries to Issue Key KGC Oracle $\mathcal{O}_{\text{ik-KGC}}$, and we use $IDList$ to store the corresponding identity of the secret key that is returned by this oracle. To simulate the state of affairs where the adversary does not know the identity of the user, after the adversary completes the first query phase, the adversary outputs two arbitrary indices $\alpha, \beta \in [Q_{key}]$ instead of two identities $DO, DU \in \mathcal{ID}$. Subsequently, by using the indices, the challenger chooses $IDList[\alpha]$ and $IDList[\beta]$ from $IDList$ as the DO and DU, respectively. Furthermore, the adversary is restricted to issuing queries to $\mathcal{O}_{\text{ct-KGC}}$ on challenged keywords $w_{b,i} \in \mathcal{W}$ and to issuing queries to $\mathcal{O}_{\text{td-KGC}}$ on challenged keywords $w_b \in \mathcal{W}$ for the challenged indices (α, β), where $b \in \{0,1\}$ and $i \in \{1, \cdots, n\}$. We consider that ICA-IBSE is MCKA-KGC secure if the advantage

$$\mathsf{Adv}^{\mathsf{MCKA\text{-}KGC}}_{\mathit{ICA\text{-}IBSE}}(\lambda) = |\Pr[b = b'] - 1/2|$$

is negligible for any adversary, and we consider that ICA-IBSE is IKGA-KGC secure if the advantage

$$\mathsf{Adv}^{\mathsf{IKGA\text{-}KGC}}_{\mathit{ICA\text{-}IBSE}}(\lambda) = |\Pr[b = b'] - 1/2|$$

is negligible for any adversary.

MCKA-KGC / IKGA-KGC Game

$pp \leftarrow \mathsf{Setup}(1^\lambda); IDList = \emptyset; Q_{key} := 1;$ random $b \in \{0,1\}$;
$(pk_{ICA}, sk_{ICA}) \leftarrow \mathsf{ICA\text{-}Setup}(pp); (pk_{KGC}, sk_{KGC}) \leftarrow \mathsf{KGC\text{-}Setup}(pp)$;
$\tilde{w_0} = (w_{0,1}, \cdots, w_{0,n}), \tilde{w_1} = (w_{1,1}, \cdots, w_{1,n}), \alpha, \beta \leftarrow$
$\mathcal{A}^{\mathcal{O}_{\mathsf{ik\text{-}KGC}}(\cdot), \mathcal{O}_{\mathsf{ct\text{-}KGC}}(\cdot), \mathcal{O}_{\mathsf{td\text{-}KGC}}(\cdot)}(pp, pk_{ICA}, pk_{KGC}, sk_{KGC})$ for $\alpha, \beta \in [Q_{key}]$;
$\tilde{ct}^* = (ct_1^*, \cdots, ct_n^*)$, where $ct_i^* \leftarrow \mathsf{Encrypt}(pp, sk_{IDList[\alpha]}, IDList[\beta], w_{b,i})$,
for $i = 1, \cdots, n$;
$w_0, w_1, \alpha, \beta \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{ik\text{-}KGC}}(\cdot), \mathcal{O}_{\mathsf{ct\text{-}KGC}}(\cdot), \mathcal{O}_{\mathsf{td\text{-}KGC}}(\cdot)}(pp, pk_{ICA}, pk_{KGC}, sk_{KGC})$
for $\alpha, \beta \in [Q_{key}]$;
$td^* \leftarrow \mathsf{Trapdoor}(pp, IDList[\alpha], sk_{IDList[\beta]}, w_b)$;
$b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{ik\text{-}KGC}}(\cdot), \mathcal{O}_{\mathsf{ct\text{-}KGC}}(\cdot), \mathcal{O}_{\mathsf{td\text{-}KGC}}(\cdot)}(pp, pk_{ICA}, pk_{KGC}, sk_{KGC})$;

IV. Identity-certifying Authority-aided Identity-based Searchable Encryption Framework

In this section, we first introduce the requisite preliminaries (including some background on the pairing and signature) before proposing a concrete construction. We also analyze the correctness and consistency of the proposed scheme.

A. Preliminaries

Symmetric Bilinear Groups. Let q be a λ-bit prime, and let $\mathbb{G}_1$ and $\mathbb{G}_T$ be two cyclic groups of the same prime order q, where $g \in \mathbb{G}_1$ is a generator. In addition, let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ be a bilinear pairing (a map). We consider that the tuple $(q, g, \mathbb{G}_1, \mathbb{G}_T, \hat{e})$ is a symmetric bilinear group if the following properties are satisfied:

- Bilinearlity: for all $g_1, g_2 \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_q$, $\hat{e}(g_1^a, g_2^b) = \hat{e}(g_1, g_2)^{ab}$;
- Nondegeneracy: $g_T := \hat{e}(g, g)$ is a generator of $\mathbb{G}_T$ (*i.e.*, $g_T \neq 1$ holds);
- Computability: $\hat{e}$ is efficiently computable.

Digital Signature Scheme. A digital signature scheme Sig with the message space $\mathcal{M}$ comprises three algorithms:

$\mathsf{Sig.KeyGen}(1^\lambda)$: The security parameter λ is taken as the input, and a verification key $\mathsf{vk}_{\mathsf{Sig}}$ and signing key $\mathsf{sk}_{\mathsf{Sig}}$ are the outputs.

$\mathsf{Sig.Sign}(\mathsf{sk}_{\mathsf{Sig}}, m)$: The signing key $\mathsf{sk}_{\mathsf{Sig}}$ and a message $m \in \mathcal{M}$ are take as the inputs, and a signature σ_{sig} is the output.

$\mathsf{Sig.Verify}(\mathsf{vk}_{\mathsf{Sig}}, m, \sigma_{\mathsf{Sig}})$: The verification key $\mathsf{vk}_{\mathsf{Sig}}$, a message $m \in \mathcal{M}$, and a signature σ_{Sig} are taken as the inputs, and 1 is the output if the signature is valid; otherwise, $\perp$ is the output.

A signature scheme Sig is considered to be correct if for all λ, all messages $m \in \mathcal{M}$, all $(\mathsf{vk}_{\mathsf{Sig}}, \mathsf{sk}_{\mathsf{Sig}}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$, and all $\sigma_{\mathsf{Sig}} \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}_{\mathsf{Sig}}, m)$, the following holds:

$$\Pr[\mathsf{Sig.Verify}(\mathsf{vk}_{\mathsf{Sig}}, m, \sigma_{\mathsf{Sig}}) = 1] = 1.$$

In addition, we consider that Sig is secure against an existential unforgeability under an adaptive chosen message attack (EU-CMA) if, for any adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathsf{Sig}}^{\mathsf{EU\text{-}CMA}}(\lambda) = \Pr[\mathsf{Sig.Verify}(\mathsf{vk}_{\mathsf{Sig}}, m^*, \sigma_{\mathsf{Sig}}^*) = 1]$ is negligible. Here, the adversary $\mathcal{A}$ is allowed to query the signing oracle $\mathcal{O}_{\mathsf{sign}}(\cdot)$ on any message $m \neq m^* \in \mathcal{M}$ and allowed to obtain the corresponding signature σ_{Sig}.

EU-CMA Game

$(\mathsf{vk}_{\mathsf{Sig}}, \mathsf{sk}_{\mathsf{Sig}}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$;
$(m^*, \sigma_{\mathsf{Sig}}^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{sign}}(\cdot)}(\mathsf{vk}_{\mathsf{Sig}})$;

B. Our Construction

Let the identity space $\mathcal{ID}$ of the ICA-IBSE scheme be $\mathcal{ID} = \mathbb{Z}_q$ and the keyword space $\mathcal{W}$ of the ICA-IBSE scheme be $\mathcal{W} = \{0,1\}^n$ for some n. In addition, let Sig : $(\mathsf{Sig.KeyGen}, \mathsf{Sig.KeyGen}, \mathsf{Sig.Verify})$ be an EU-CMA-secure digital signature with message space $\mathcal{M} = \{0,1\}^m$ for some m.

$\mathsf{Setup}(1^\lambda)$: This algorithm chooses two multiplicative cyclic groups $\mathbb{G}_1, \mathbb{G}_T$ with prime order q; a generator $g \in \mathbb{G}_1$; a pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$; and three cryptographic hash functions $H : \{0,1\}^* \to \mathbb{G}_1$, $h_1 : \{0,1\}^* \times \{0,1\}^* \times \mathbb{G}_T \times \{0,1\}^* \to \mathbb{Z}_q^*$, and $h_2 : \mathbb{Z}_q^* \times \mathbb{G}_1 \to \mathbb{Z}_q^*$. It sets the system parameter to be

$$pp := \{1^\lambda, \mathbb{G}_1, \mathbb{G}_T, \hat{e}, q, g, H, h_1, h_2\}.$$

$\mathsf{ICA\text{-}Setup}(pp)$: This algorithm runs $(\mathsf{vk}_{\mathsf{Sig}}, \mathsf{sk}_{\mathsf{Sig}}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$. It then outputs

$$pk_{ICA} := \mathsf{vk}_{\mathsf{Sig}}; sk_{ICA} := \mathsf{sk}_{\mathsf{Sig}}.$$

$\mathsf{KGC\text{-}Setup}(pp)$: The KGC picks $x \leftarrow \mathbb{Z}_q^*$ and computes $Y = g^x$. It then outputs

$$pk_{KGC} := Y; sk_{KGC} := x.$$

$\mathsf{ICA\text{-}Cert}(pp, pk_{ICA}, sk_{ICA}, ID)$: The ICA computes $u_{ID} = H(ID)$, picks $y_{ID,1} \leftarrow \mathbb{Z}_q^*$, and computes $u_{ID,1} = g^{y_{ID,1}}$. In addition, it computes $u_{ID,2} = u_{ID} u_{ID,1} \in \mathbb{G}_1$ and $\sigma_{\mathsf{Sig}} \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}_{\mathsf{Sig}}, u_{ID,2})$. Finally, it outputs

$$cert_{ID} := (u_{ID,2}, \sigma_{\mathsf{Sig}}); tf_{ID} := y_{ID,1}.$$

$\langle \mathsf{User\text{-}Obtain\text{-}Key}(pp, pk_{KGC}, ID, cert_{ID}, tf_{ID}),$ $\mathsf{KGC\text{-}Issue\text{-}Key}(pp, pk_{KGC}, sk_{KGC}, pk_{ICA}) \rangle$: The user and the KGC run the following steps:

1) The user sets $\mathsf{M}_1 = cert_{ID} = (u_{ID,2}, \sigma_{\mathsf{Sig}})$ and sends M_1 to the KGC.
2) After receiving M_1, the KGC verifies the correctness of σ_{Sig}. If $\mathsf{Sig.Verify}(\mathsf{vk.Sig}, u_{ID,2}, \sigma_{Sig}) = \perp$, the KGC sets $\mathsf{M}_2 = \perp$. Otherwise, it computes $\mathsf{M}_2 = y_{ID,2} = u_{ID,2}^x$. Finally, it returns M_2 to the user.
3) If $\mathsf{M}_2 = \perp$, the user outputs $\perp$. Otherwise, it computes $e_{ID} = y_{ID,2} \cdot Y^{-y_{ID,1}}$ and outputs
$$sk_{ID} := e_{ID}.$$

$\mathsf{Encrypt}(pp, sk_{DO}, DU, w)$: The data owner randomly selects $r \leftarrow \mathbb{Z}_q^*$, and computes $c_1 = g^r, c_2 = g^{rh_2(h_1(DO,DU,k,w),c_1)}$, where $k = \hat{e}(sk_{DO}, H(DU))$. Subsequently, it outputs

$$ct_w := (c_1, c_2).$$

$\mathsf{Trapdoor}(pp, DO, sk_{DU}, w)$: The DU outputs

$$td_w := h_1(DO, DU, k, w),$$

where $k = \hat{e}(H(DO), sk_{DU})$.

$\mathsf{Test}(pp, ct_w, td_w)$: The CS checks whether $c_2 = c_1^{h_2(td_w, c_1)}$. It returns 1 if the equation is satisfied and returns 0 if otherwise.

C. Correctness and Consistency of ICA-IBSE

We analyze the correctness and consistency of the proposed ICA-IBSE scheme as follows.

1) For $i = \{DO, DU\}$, we have

$$\begin{aligned} sk_i &= y_{i,2} \cdot Y^{-y_{i,1}} \\ &= u_{i,2}^x \cdot Y^{-y_{i,1}} \\ &= u_i^x \cdot u_{i,1}^x \cdot Y^{-y_{i,1}} \\ &= H(i)^x \cdot Y^{y_{i,1}} \cdot Y^{-y_{i,1}} \\ &= H(i)^x. \end{aligned}$$

2) Considering the Test algorithm for a ciphertext $ct_w = (c_1, c_2)$ and a trapdoor $td_{w'}$, we have

$$\begin{aligned} g^{rh_2(h_1(DO,DU,k,w),c_1)} = c_2 &= c_1^{h_2(td_{w'},c_1)} \\ &= g^{rh_2(td_{w'},c_1)} \\ &= g^{rh_2(h_1(DO,DU,k',w'),c_1)} \end{aligned}$$

3) In addition, we have

$$\begin{aligned} k = \hat{e}(sk_{DO}, H(DU)) &= \hat{e}(H(DO)^x, H(DU)) \\ = \hat{e}(H(DO), H(DU)^x) &= \hat{e}(H(DO), sk_{DU}) \\ = k'. \end{aligned}$$

Because $k = k'$, when $w = w'$, we have $h_1(DO, DU, k, w) = h_1(DO, DU, k', w')$ and $c_2 = c_1^{h_2(td_{w'},c_1)}$; therefore, correctness is satisfied. Conversely, when $w \neq w'$, because the probability that $h_1(DO, DU, k, w) = h_1(DO, DU, k', w')$ is negligibly low, we have $c_2 \neq c_1^{h_2(td_{w'},c_1)}$; therefore, consistency is also satisfied.

V. Security Analysis of ICA-IBSE

In this section, we demonstrate that our scheme is secure against various forms of attacks.

Gap Bilinear Diffie–Hellman Assumption [33], [34]. Given a symmetric bilinear group tuple $\Phi = (q, g, \mathbb{G}_1, \mathbb{G}_T, \hat{e})$, we consider that the GBDH assumption holds if, for any probabilistic polynomial-time algorithm (PPT) adversary $\mathcal{A}$, the advantage (defined as follows) is negligible:

$$\begin{aligned} &\mathsf{Adv}_\Phi^{\mathsf{GBDH}}(\mathcal{A}, q_{\mathsf{DBDH}}) := \\ &\Pr[T = \hat{e}(g,g)^{abc} \mid a,b,c \in \mathbb{Z}_q; T \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{DBDH}}}(\Phi, g^a, g^b, g^c)], \end{aligned}$$

where $\mathcal{O}_{\mathsf{DBDH}}$ denotes a decision bilinear Diffie–Hellman oracle that takes (g^a, g^b, g^c, T) as its input and outputs 1 if $\hat{e}(g,g)^{abc} = T$ and 0 if otherwise; q_{DBDH} denotes the maximum number of queries by $\mathcal{A}$ to $\mathcal{O}_{\mathsf{DBDH}}$.

Theorem 1. *The proposed construction is MCKA-CS secure if the underlying signature scheme* Sig *is EU-CMA secure under the hard GBDH assumption.*

Proof. Suppose that some PPT algorithm $\mathcal{A}$ can break the MCKA-CS security of the proposed scheme. If so, then the following proof demonstrates that some other algorithm $\mathcal{B}$ can use $\mathcal{A}$ to solve the GBDH assumption.

Before the beginning of the game, $\mathcal{B}$ is given a GBDH instance (g, g^a, g^b, g^c), where $a, b, c \in \mathbb{Z}_q^*$ are random choices.

Initialization. $\mathcal{B}$ first generates the system parameter $pp = \{1^\lambda, \mathbb{G}_1, \mathbb{G}_T, \hat{e}, q, g, H, h_1, h_2\}$ according to the scheme. Subsequently, it chooses $\ell_1, \ell_2 \leq q_H$ randomly as the indices of the challenged identities for the DO and DU, respectively. Here, q_H is the maximum number of queries that could be queried by the H-oracle for different identities. $\mathcal{B}$ also runs $(pk_{ICA}, sk_{ICA}) \leftarrow \mathsf{ICA\text{-}Setup}(pp)$ and sets $pk_{KGC} = g^a$. Furthermore, $\mathcal{B}$ sets initials on four lists (cert-list, H-list, h_1-list, and h_2-list) and randomly chooses a bit $b \in \{0, 1\}$. Finally, $\mathcal{B}$ returns (pp, pk_{ICA}, pk_{KGC}) to $\mathcal{A}$.

Phase 1. In this phase, $\mathcal{A}$ is allowed to query the following oracles adaptively at polynomially many instances.

- H-oracle: On the ith nonrepeated query $ID_i \in \mathcal{ID}$, $\mathcal{B}$ first searches the H-list for the entry $(i, ID_i, \mu_{ID_i}, u_{ID_i})$. If no such entry exists, then $\mathcal{B}$ executes the following under the following conditions:
  - If $i \notin \{\ell_1, \ell_2\}$: $\mathcal{B}$ randomly chooses $\mu_{ID_i} \in \mathbb{Z}_q^*$, computes $u_{ID_i} = g^{\mu_{ID_i}}$, adds $(i, ID_i, \mu_{ID_i}, u_{ID_i})$ to H-list, and returns u_{ID_i} to $\mathcal{A}$.
  - if $i = \ell_1$: $\mathcal{B}$ sets $u_{ID_i} = g^b$, adds $(i, ID_i, \perp, u_{ID_i})$ to H-list, and returns u_{ID_i} to $\mathcal{A}$.
  - if $i = \ell_2$: $\mathcal{B}$ sets $u_{ID_i} = g^c$, adds $(i, ID_i, \perp, u_{ID_i})$ to H-list, and returns u_{ID_i} to $\mathcal{A}$.
- h_1-oracle: On the query (ID_i, ID_j, k, w), $\mathcal{B}$ searches h_1-list for the entry $((ID_i, ID_j, k, w), h)$ and returns h. Note that we assume that $((ID_i, ID_j, k, w), h)$ is identical to $((ID_j, ID_i, k, w), h)$ for h_1-list. If no such entry exists, $\mathcal{B}$ runs the following steps:
  - retrieve u_{ID_i} and u_{ID_j} by calling $H(ID_i)$ and $H(ID_j)$, respectively.
  - check if $\mathcal{O}_{\mathsf{DBDH}}(pk_{KGC}, u_{ID_i}, u_{ID_j}, k) = 1$.
  - if $\{i, j\} = \{\ell_1, \ell_2\}$ and the DBDH oracle returns 1, return k as the answer to the GBDH problem and abort.
  - randomly choose $h \in \mathbb{Z}_q^*$, add $(ID_i, ID_j, k, w), h)$ to h_1-list, and return h to $\mathcal{A}$.
- h_2-oracle: On the query (h, c_1), $\mathcal{B}$ searches h_2-list for the entry $((h, c_1), \tilde{h})$ and returns $\tilde{h}$. If no such entry exists, $\mathcal{B}$ randomly chooses $\tilde{h} \in \mathbb{Z}_q^*$, adds $((h, c_1), \tilde{h})$ to h_2-list, and returns $\tilde{h}$ to $\mathcal{A}$.
- $\mathcal{O}_{\mathsf{cert}}(\cdot)$: when $\mathcal{A}$ queries for a certificate corresponding to ID, $\mathcal{B}$ first goes through H-list for a tuple $(*, ID, *, *)$. If no such tuple is found, $\mathcal{B}$ calls $H(ID)$ first and obtains $(i, ID, \mu_{ID}, u_{ID})$. Subsequently, $\mathcal{B}$ samples $y_{ID,1} \leftarrow \mathbb{Z}_q^*$ and computes $u_{ID,1} = g^{y_{ID,1}}$, $u_{ID,2} = u_{ID}u_{ID,1}$, and $\sigma_{\mathsf{Sig}} \leftarrow \mathsf{Sig.Sign}(sk_{ICA}, u_{ID,2})$. $\mathcal{B}$ also sets $cert_{ID} = (u_{ID,2}, \sigma_{\mathsf{Sig}})$ and $tf_{ID} = y_{ID,1}$ and adds $(cert_{ID}, tf_{ID}, ID)$ to cert-list. Finally, $\mathcal{B}$ returns $(cert_{ID}, tf_{ID})$ to $\mathcal{A}$.
- $\mathcal{O}_{\mathsf{sk}}(\cdot)$: when $\mathcal{A}$ queries for a secret key with a first-round message M_1, $\mathcal{B}$ parses $\mathsf{M}_1 = (u_{ID,2}, \sigma_{\mathsf{Sig}})$. $\mathcal{B}$ returns $\perp$ if $\mathsf{Sig.Verify}(pk_{ICA}, u_{ID,2}, \sigma_{\mathsf{Sig}}) = \perp$. Otherwise, $\mathcal{B}$ extracts $tf_{ID} = y_{ID,1}$ from $(cert_{ID} = (u_{ID,2}, \sigma_{\mathsf{Sig}}), tf_{ID} = y_{ID,1}, ID) \in$ cert-list. Here, we note that because Sig is EU-CMA secure, if the verification passes, it necessarily exists in cert-list. Subsequently, $\mathcal{B}$ goes through H-list for the tuple $(i, ID, \mu_{ID}, *)$.

- if $i \notin \{\ell_1, \ell_2\}$: $\mathcal{B}$ computes $y_{ID,2} = pk_{KGC}^{y_{ID,1}+\mu_{ID}}$. Subsequently, $\mathcal{B}$ sets $\mathsf{M}_2 = y_{ID,2}$ and returns it to $\mathcal{A}$.
- if $i \in \{\ell_1, \ell_2\}$: $\mathcal{B}$ aborts the game and outputs a random element in $\mathbb{G}_T$.

- $\mathcal{O}_{\mathsf{ct}}(\cdot)$: when $\mathcal{A}$ queries for a ciphertext with (ID_i, ID_j, w), $\mathcal{B}$ executes the following steps:
  - retrieve $(i, ID_i, \mu_{ID_i}, u_{ID_i})$ and $(j, ID_j, \mu_{ID_j}, u_{ID_j})$ from H-list for ID_i and ID_j, respectively.
  - randomly select $r \in \mathbb{Z}_q^*$.
  - if $\{i, j\} = \{\ell_1, \ell_2\}$: search h_1-list for the tuple $((ID_i, ID_j, \perp, w), h)$; if no such tuple exists, randomly choose $h \in \mathbb{Z}_q^*$ and add $((ID_i, ID_j, \perp, w), h)$ to h_1-list.
  - if otherwise (*i.e.,* $i \notin \{\ell_1, \ell_2\}$ or $j \notin \{\ell_1, \ell_2\}$): either compute $k = \hat{e}(pk_{KGC}^{\mu_{ID_i}}, u_{ID_j})$ if $i \notin \{\ell_1, \ell_2\}$ or compute $k = \hat{e}(u_{ID_i}, pk_{KGC}^{\mu_{ID_j}})$ if $j \notin \{\ell_1, \ell_2\}$. Note that if $i \notin \{\ell_1, \ell_2\}$ and $j \notin \{\ell_1, \ell_2\}$, $\mathcal{B}$ can randomly set $k = \hat{e}(pk_{KGC}^{\mu_{ID_i}}, u_{ID_j})$ or $k = \hat{e}(u_{ID_i}, pk_{KGC}^{\mu_{ID_j}})$.
  - search h_1-list for the tuple $((ID_i, ID_j, k, w), h)$. If no such tuple exists, randomly choose $h \in \mathbb{Z}_q^*$ and add $((ID_i, ID_j, k, w), h)$ to h_1-list.
  - compute $c_1 = g^r$ and $c_2 = g^{r\tilde{h}}$, where $\tilde{h}$ is retrieved from h_2-list (*i.e.,* $\tilde{h} = h_2(h, c_1)$).
  - return $ct_w = (c_1, c_2)$.
- $\mathcal{O}_{\mathsf{td}}(\cdot)$: when $\mathcal{A}$ queries for a trapdoor with (ID_i, ID_j, w), $\mathcal{B}$ executes the following steps:
  - retrieve $(i, ID_i, \mu_{ID_i}, u_{ID_i})$ and $(j, ID_j, \mu_{ID_j}, u_{ID_j})$ from H-list for ID_i and ID_j, respectively.
  - randomly select $r \in \mathbb{Z}_q^*$.
  - if $\{i, j\} = \{\ell_1, \ell_2\}$: search h_1-list for the tuple $((ID_i, ID_j, \perp, w), h)$. If no such tuple exists, randomly choose $h \in \mathbb{Z}_q^*$ and add $((ID_i, ID_j, \perp, w), h)$ to h_1-list.
  - if otherwise (*i.e.,* $i \notin \{\ell_1, \ell_2\}$ or $j \notin \{\ell_1, \ell_2\}$), either compute $k = \hat{e}(pk_{KGC}^{\mu_{ID_i}}, u_{ID_j})$ if $i \notin \{\ell_1, \ell_2\}$ or compute $k = \hat{e}(u_{ID_i}, pk_{KGC}^{\mu_{ID_j}})$ if $j \notin \{\ell_1, \ell_2\}$. Note that if $i \notin \{\ell_1, \ell_2\}$ and $j \notin \{\ell_1, \ell_2\}$, $\mathcal{B}$ can randomly set $k = \hat{e}(pk_{KGC}^{\mu_{ID_i}}, u_{ID_j})$ or $k = \hat{e}(u_{ID_i}, pk_{KGC}^{\mu_{ID_j}})$.
  - search h_1-list for the tuple $((ID_i, ID_j, k, w), h)$. If no such tuple exists, randomly choose $h \in \mathbb{Z}_q^*$ and add $((ID_i, ID_j, k, w), h)$ to h_1-list.
  - return $td_w = h$.

Challenge. At the end of **Phase 1**, $\mathcal{A}$ outputs the challenged tuple $(\tilde{w_0} = \{w_{0,1}, \cdots, w_{0,n}\}, \tilde{w_1} = \{w_{1,1}, \cdots, w_{1,n}\}, DO, DU)$ and $\mathcal{B}$ executes the following steps:

- obtain $(i, DO, \perp, u_{DO})$ and $(j, DU, \perp, u_{DU})$ by calling $H(DO)$ and $H(DU)$, respectively. If $\{i, j\} \neq \{\ell_1, \ell_2\}$, abort the game.
- for $i = 1, \cdots, n$, execute the following steps. First, randomly choose $r_i \in \mathbb{Z}_q^*$. Second, search h_1-list for the tuple $((DO, DU, \perp, w_{b,i}), h_i)$; if no such entry is found, randomly choose $h_i \in \mathbb{Z}_q^*$ and add $((DO, DU, \perp, w_{b,i}), h_i)$ to h_1-list. Third, compute $ct_i^* = (c_{1,i}^*, c_{2,i}^*)$, where $c_{1,i}^* = g^{r_i}$, $c_{2,i}^* = g^{r_i\tilde{h_i}}$, and $\tilde{h_i}$ is retrieved from h_2-list (*i.e.,* $\tilde{h_i} = h_2(h_i, c_{1,i}^*)$).
- return challenged ciphertext $\tilde{ct^*} = (ct_1^*, \cdots, ct_n^*)$.

Phase 2. In this phase, $\mathcal{A}$ can keep the query oracles identical to those in **Phase 1**.

Guess. Finally, $\mathcal{A}$ outputs $b' \in \{0, 1\}$ as its guess. $\mathcal{B}$ searches h_1-list for k^* such that $\mathcal{O}_{\mathsf{DBDH}}(g^a, g^b, g^c, k^*) = 1$ and returns k^* as answer.

Analyze. Because $\mathcal{B}$ follows the proposed scheme, with the exception that the hash functions are modeled by random oracles, its simulation is identical to that of the real scheme. Because ℓ_1 and ℓ_2 are independent of $\mathcal{A}$'s perspective, the probability that $\mathcal{B}$ does not abort the game (*i.e.,* $\{i, j\} \notin \{\ell_1, \ell_2\}$ in querying h_1-oracle and $\{DO, DU\} = \{ID_{\ell_1}, ID_{\ell_2}\}$ in **Challenge**) is $\frac{2}{q_H}$, where q_{h_1} is the maximum number of queries that could be made to the h_1 oracle for different inputs. Furthermore, because h_1 is modeled as a random oracle, the adversary's advantage is negligible unless $(DO, DU, k^*, w_{b,i}, h)$ appears on h_1-list such that $k^* = \hat{e}(g, g)^{abc}$. If this tuple appears on h_1-list, then $\mathcal{B}$ is necessarily able to solve the GBDH problem. Therefore, if there exists some $\mathcal{A}$ that can break the MCKA-CS-secure scheme at a nonnegligible advantage ϵ, then there exists some $\mathcal{B}$ that can break the GBDH problem at a nonnegligible advantage $\epsilon' \geq \epsilon \cdot \frac{2}{q_H}$. □

Theorem 2. *The proposed scheme is IKGA-CS secure if the underlying signature scheme* Sig *is EU-CMA secure under the hard GBDH assumption.*

Proof. The proof is similar to the proof of Theorem 1, except for the **Challenge** phase. Thus, we describe only the proof for the **Challenge** phase.

Challenge. At the end of **Phase 1**, $\mathcal{A}$ outputs a challenged tuple (w_0, w_1, DO, DU), and $\mathcal{B}$ executes the following steps:

- obtain $(i, DO, \perp, u_{DO})$ and $(j, DU, \perp, u_{DU})$ by calling $H(DO)$ and $H(DU)$, respectively. If $\{i, j\} \neq \{\ell_1, \ell_2\}$, abort the game.
- search h_1-list for the tuple $((DO, DU, \perp, w_b), h)$. If no such tuple is found, randomly choose $h \in \mathbb{Z}_q^*$ and add $((DO, DU, \perp, w_b), h)$ to h_1-list.
- return challenged trapdoor $td^* = h$.

□

Theorem 3. *The proposed scheme is MCKA-ICA and IKGA-ICA secure if it is MCKA-CS and IKGA-CS secure, respectively.*

Proof. This proof is intuitive. Because the adversary in MCKA-ICA and IKGA-ICA is a weaker variant of that in MCKA-CS and IKGA-CS, respectively, Theorem 1 and Theorem 2 entail Theorem 3. Note that because $\mathcal{B}$ need not reply to the certificate query for $\mathcal{A}$, the signature scheme Sig need not be EU-CMA secure. □

Theorem 4. *The proposed scheme is MCKA-KGC secure under the hard GBDH assumption.*

Proof. Suppose that some PPT algorithm $\mathcal{A}$ can break the MCKA-KGC security of the proposed scheme. If so, then the following proof demonstrates that some other algorithm $\mathcal{B}$ can use $\mathcal{A}$ to solve the GBDH problem.

Before the beginning of the game, $\mathcal{B}$ is given a GBDH instance (g, g^a, g^b, g^c), where $a, b, c \in \mathbb{Z}_q^*$ are random choices.
Initialization. $\mathcal{B}$ first generates the system parameter $pp = \{1^\lambda, \mathbb{G}_1, \mathbb{G}_T, \hat{e}, q, g, H, h_1, h_2\}$ according to the scheme. Subsequently, it sets $Q_{key} := 1$ and sets the initials of an empty list IDList to count the number of $\mathcal{A}$ queries to $\mathcal{O}_{\text{ik-KGC}}$ and store the corresponding identity of the secret key returned by this oracle. $\mathcal{B}$ also runs $(pk_{ICA}, sk_{ICA}) \leftarrow \text{ICA-Setup}(pp)$ and $(pk_{KGC}, sk_{KGC}) \leftarrow \text{KGC-Setup}(pp)$ and chooses two arbitrary indices $\ell_1, \ell_2 \leq \text{Max}_{Q_{key}}$, where $\text{Max}_{Q_{key}}$ is the maximum number of queries that can be queried by the $\mathcal{O}_{\text{ik-KGC}}$ oracle. In addition, $\mathcal{B}$ sets the initials for three additional lists (H-list, h_1-list, and h_2-list) and randomly chooses a bit $b \in \{0, 1\}$. Finally, $\mathcal{B}$ returns $(pp, pk_{ICA}, pk_{KGC}, sk_{KGC})$ to $\mathcal{A}$.
Phase 1. In this phase, $\mathcal{A}$ is allowed to query the following oracles adaptively at polynomially many instances.

- H-oracle: On the ith nonrepeated query ID_i, $\mathcal{B}$ first searches H-list for the entry $(i, ID_i, \mu_{ID_i}, u_{ID_i})$. If no such entry exists, then $\mathcal{B}$ randomly chooses $\mu_{ID_i} \in \mathbb{Z}_q^*$, computes $u_{ID_i} = g^{\mu_{ID_i}}$, adds $(i, ID_i, \mu_{ID_i}, u_{ID_i})$ to H-list, and returns u_{ID_i} to $\mathcal{A}$.
- h_1-oracle: On the query (ID_i, ID_j, k, w), $\mathcal{B}$ first searches h_1-list for the entry $((ID_i, ID_j, k, w), h)$ and returns h. Notably, we assume that $((ID_i, ID_j, k, w), h)$ is identical to $((ID_j, ID_i, k, w), h)$ for h_1-list. If no such entry exists, $\mathcal{B}$ executes the following steps:
  - retrieve u_{ID_i} and u_{ID_j} by calling $H(ID_i)$ and $H(ID_j)$, respectively.
  - check if $\mathcal{O}_{\text{DBDH}}(pk_{KGC}, u_{ID_i}, u_{ID_j}, k) = 1$.
  - if $\{i, j\} = \{\ell_1, \ell_2\}$ and the DBDH oracle returns 1, return k as the answer to the GBDH problem and abort.
  - randomly choose $h \in \mathbb{Z}_q^*$, add $(ID_i, ID_j, k, w), h)$ to h_1-list, and return h to $\mathcal{A}$.
- h_2-oracle: On the query (h, c_1), $\mathcal{B}$ searches h_2-list for the entry $((h, c_1), \tilde{h})$ and returns $\tilde{h}$. If no such entry exists, then $\mathcal{B}$ randomly chooses $\tilde{h} \in \mathbb{Z}_q^*$, adds $((h, c_1), \tilde{h})$ to h_2-list, and returns $\tilde{h}$ to $\mathcal{A}$.
- $\mathcal{O}_{\text{ik-KGC}}(\cdot)$: When $\mathcal{A}$ issues a issue key query, $\mathcal{B}$ first samples $ID \leftarrow \mathcal{ID}$ such that ID does not exist in H-list. Subsequently, $\mathcal{B}$ executes the following steps under the following conditions:
  - if $Q_{key} \notin \{\ell_1, \ell_2\}$: $\mathcal{B}$ obtains u_{ID} by querying $H(ID)$.
  - if $Q_{key} = \ell_1$: $\mathcal{B}$ sets $u_{ID} = g^{ab}$ and adds $(ID, \perp, u_{ID})$ to H-list.
  - if $Q_{key} = \ell_2$: $\mathcal{B}$ sets $u_{ID} = g^c$ and adds $(ID, \perp, u_{ID})$ to H-list.

  In addition, $\mathcal{B}$ randomly chooses $y_{ID,1} \leftarrow \mathbb{Z}_q^*$, computes $u_{ID,1} = g^{y_{ID,1}}$, and computes $u_{ID,2} = u_{ID} u_{ID,1} \in \mathbb{G}$ and $\sigma_{\text{Sig}} \leftarrow \text{Sig.Sign}(sk_{ICA}, u_{ID,2})$. Finally, $\mathcal{B}$ returns $\mathsf{M}_1 = (u_{ID,2}, \sigma_{\text{Sig}})$ to $\mathcal{A}$, sets $\mathit{IDList}[Q_{key}] = ID$, and updates $Q_{key} = Q_{key} + 1$.
- $\mathcal{O}_{\text{ct}}(\cdot)$: when $\mathcal{A}$ queries for a ciphertext with (i, j, w), where $i, j \in [Q_{key}]$, $\mathcal{B}$ executes the following steps:
  - retrieve $(\mathit{IDList}[i], \mu_{\mathit{IDList}[i]}, u_{\mathit{IDList}[i]})$ and $(\mathit{IDList}[j], \mu_{\mathit{IDList}[j]}, u_{\mathit{IDList}[j]})$ from H-list for $\mathit{IDList}[i]$ and $\mathit{IDList}[i]$, respectively.
  - randomly select $r \in \mathbb{Z}_q^*$.
  - if $\{i, j\} = \{\ell_1, \ell_2\}$: search h_1-list for the tuple $((\mathit{IDList}[i], \mathit{IDList}[j], \perp, w), h)$. If no such tuple exists, randomly choose $h \in \mathbb{Z}_q^*$ and add $((\mathit{IDList}[i], \mathit{IDList}[j], \perp, w), h)$ to h_1-list.
  - if otherwise (*i.e.,* $i \notin \{\ell_1, \ell_2\}$ or $j \notin \{\ell_1, \ell_2\}$), either compute $k = \hat{e}(pk_{KGC}^{\mu_{\mathit{IDList}[i]}}, u_{\mathit{IDList}[j]})$ if $i \notin \{\ell_1, \ell_2\}$ or compute $k = \hat{e}(u_{\mathit{IDList}[i]}, pk_{KGC}^{\mu_{\mathit{IDList}[j]}})$ if $j \notin \{\ell_1, \ell_2\}$. Note that if $i \notin \{\ell_1, \ell_2\}$ and $j \notin \{\ell_1, \ell_2\}$, $\mathcal{B}$ can randomly set $k = \hat{e}(pk_{KGC}^{\mu_{\mathit{IDList}[i]}}, u_{\mathit{IDList}[j]})$ or $k = \hat{e}(u_{\mathit{IDList}[i]}, pk_{KGC}^{\mu_{\mathit{IDList}[j]}})$.
  - search h_1-list for the tuple $((\mathit{IDList}[i], \mathit{IDList}[j], k, w), h)$. If no such tuple exists, randomly choose $h \in \mathbb{Z}_q^*$ and add $((\mathit{IDList}[i], \mathit{IDList}[j], k, w), h)$ to h_1-list.
  - compute $c_1 = g^r$ and $c_2 = g^{r\tilde{h}}$, where $\tilde{h}$ is retrieved from h_2-list (*i.e.,* $h_2(h, c_1)$).
  - return $ct_w = (c_1, c_2)$.
- $\mathcal{O}_{\text{td}}(\cdot)$: when $\mathcal{A}$ queries for a trapdoor with (i, j, w), where $i, j \in [Q_{key}]$, $\mathcal{B}$ executes the following steps:
  - retrieve $(\mathit{IDList}[i], \mu_{\mathit{IDList}[i]}, u_{\mathit{IDList}[i]})$ and $(\mathit{IDList}[j], \mu_{\mathit{IDList}[j]}, u_{\mathit{IDList}[j]})$ from H-list for $\mathit{IDList}[i]$ and $\mathit{IDList}[i]$, respectively.
  - if $\{i, j\} = \{\ell_1, \ell_2\}$: search h_1-list for the tuple $((\mathit{IDList}[i], \mathit{IDList}[j], \perp, w), h)$. If no such tuple exists, randomly choose $h \in \mathbb{Z}_q^*$ and add $((\mathit{IDList}[i], \mathit{IDList}[j], \perp, w), h)$ to h_1-list.
  - if otherwise (*i.e.,* $i \notin \{\ell_1, \ell_2\}$ or $j \notin \{\ell_1, \ell_2\}$), either compute $k = \hat{e}(pk_{KGC}^{\mu_{\mathit{IDList}[i]}}, u_{\mathit{IDList}[j]})$ if $i \notin \{\ell_1, \ell_2\}$ or compute $k = \hat{e}(u_{\mathit{IDList}[i]}, pk_{KGC}^{\mu_{\mathit{IDList}[j]}})$ if $j \notin \{\ell_1, \ell_2\}$. Note that if $i \notin \{\ell_1, \ell_2\}$ and $j \notin \{\ell_1, \ell_2\}$, $\mathcal{B}$ can randomly set $k = \hat{e}(pk_{KGC}^{\mu_{\mathit{IDList}[i]}}, u_{\mathit{IDList}[j]})$ or $k = \hat{e}(u_{\mathit{IDList}[i]}, pk_{KGC}^{\mu_{\mathit{IDList}[j]}})$.
  - search h_1-list for the tuple $((\mathit{IDList}[i], \mathit{IDList}[j], k, w), h)$. If no such tuple exists, randomly choose $h \in \mathbb{Z}_q^*$ and add $((\mathit{IDList}[i], \mathit{IDList}[j], k, w), h)$ to h_1-list.
  - return $td_w = h$.

Challenge. At the end of **Phase 1**, $\mathcal{A}$ outputs the challenged tuple $(\tilde{w_0} = \{w_{0,1}, \cdots, w_{0,n}\}, \tilde{w_1} = \{w_{1,1}, \cdots, w_{1,n}\}, \alpha, \beta)$, and $\mathcal{B}$ executes the following steps:

- obtain $(\mathit{IDList}[\alpha], \perp, u_{\mathit{IDList}[\alpha]})$ and $(\mathit{IDList}[\beta], \perp, u_{\mathit{IDList}[\beta]})$ by calling $H(\mathit{IDList}[\alpha])$ and $H(\mathit{IDList}[\beta])$, respectively. If $\{\alpha, \beta\} \neq \{\ell_1, \ell_2\}$, abort the game.
- for $i = 1, \cdots, n$, perform the following steps. First, randomly choose $r_i \in \mathbb{Z}_q^*$. Second, search h_1-list for the tuple $((\mathit{IDList}[\alpha], \mathit{IDList}[\beta], \perp, w_{b,i}), h_i)$. If no such tuple is found, randomly choose $h_i \in \mathbb{Z}_q^*$ and add $((\mathit{IDList}[\alpha], \mathit{IDList}[\beta], \perp, w_{b,i}), h_i)$ to h_1-list. Third, compute $ct_i^* = (c_{1,i}^*, c_{2,i}^*)$, where $c_{1,i}^* = g^{r_i}$, $c_{2,i}^* = g^{r_i \tilde{h}_i}$ and $\tilde{h}_i$ is retrieved from h_2-list (*i.e.,* $\tilde{h}_i = h_2(h_i, c_{1,i}^*)$).
- return the challenged ciphertext $\tilde{ct}^* = (ct_1^*, \cdots, ct_n^*)$.

Phase 2. In this phase, $\mathcal{A}$ can keep the query oracles identical to those in **Phase 1**.

Guess. Finally, $\mathcal{A}$ outputs $b' \in \{0,1\}$ as its guess. $\mathcal{B}$ searches h_1-list for k^* such that $\mathcal{O}_{\mathsf{DBDH}}(g^a, g^b, g^c, (k^*)^{x^{-1}}) = 1$ and returns $(k^*)^{x^{-1}}$ as the answer.

Analysis. Because $\mathcal{B}$ follows the proposed scheme, except that the hash functions are modeled by random oracles, its simulation is identical to that of the real scheme. Because ℓ_1 and ℓ_2 are independent of $\mathcal{A}$'s perspective, the probability that $\mathcal{B}$ does not abort the game ($\{\alpha,\beta\} = \{IDList[\ell_1], IDList[\ell_2]\}$ in **Challenge**) is $\frac{2}{\mathsf{Max}_{Q_{key}}}$. Furthermore, because h_1 is modeled as a random oracle, the adversary's advantage is negligible, unless $(IDList[\alpha], IDList[\beta], k^*, w_{b,i}, h)$ appears in h_1-list such that $(k^*)^{x^{-1}} = \hat{e}(g,g)^{abcx(x^{-1})} = \hat{e}(g,g)^{abc}$. If this tuple appears in h_1-list, then $\mathcal{B}$ is necessarily able to solve the GBDH problem. Therefore, if there exists such an $\mathcal{A}$ that can break the MCKA-KGC-secure scheme at a nonnegligible advantage ϵ, then there exists some $\mathcal{B}$ that can break the GBDH problem at a nonnegligible advantage $\epsilon' \geq \epsilon \cdot \frac{2}{\mathsf{Max}_{Q_{key}}}$. □

Theorem 5. *The proposed scheme is IKGA-KGC secure under the hard GBDH assumption.*

Proof. The proof is similar to the proof of Theorem 4, except for the **Challenge** phase. Therefore, only the proof for the **Challenge** phase is presented.

Challenge. At the end of **Phase 1**, $\mathcal{A}$ outputs the challenged tuple $(w_0, w_1, \alpha, \beta)$ and $\mathcal{B}$ executes the following steps:

- obtain $(IDList[\alpha], \perp, u_{IDList[\alpha]})$ and $(IDList[\beta], \perp, u_{IDList[\beta]})$ by calling $H(IDList[\alpha])$ and $H(IDList[\beta])$, respectively. If $\{\alpha,\beta\} \neq \{\ell_1,\ell_2\}$, $\mathcal{B}$ aborts the game.
- search h_1-list for the tuple $((IDList[\alpha], IDList[\beta], \perp, w_b), h)$. If no such tuple is found, randomly choose $h \in \mathbb{Z}_q^*$ and add $((IDList[\alpha], IDList[\beta], \perp, w_b), h)$ to h_1-list.
- return the challenged trapdoor $td^* = h$.

□

VI. Theoretical Comparison and Performance Evaluation

In this section, we detail the theoretical comparison of our scheme with other state-of-the-art schemes, specifically the PAEKS schemes CWZH19 [20] and QCHLZ20 [21], the IBAEKS scheme LHSYS19 [23], the CBAEKS schemes LLZ19 [25] and LLW21 [24], and the CLAEKS schemes HMZKL19 [28] and PSE20 [26]. The features of these schemes are listed in Table I. We also evaluate the performance of our proposed scheme against that of the LHSYS19 [23], QCHLZ20 [21], and LLW21 [24] schemes.

A. Theoretical Comparison

We compare the schemes with respect to their communication cost and computational cost. The comparison results are presented in Table II. For communication cost, we use $|\mathbb{Z}_q|$, $|\mathbb{G}_1|$, and $|\mathbb{G}_T|$ to denote the bit lengths of group elements in $\mathbb{Z}_q$, $\mathbb{G}_1$, and $\mathbb{G}_T$, respectively. In addition, we use $|ID|$ and $|h|$ to denote the bit lengths of a user's identity and output of the hash function, respectively. Note that in the CBAEKS

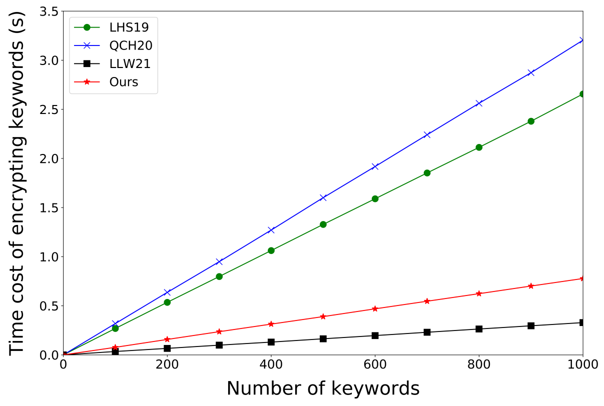

Fig. 2. Computation cost of encrypting keywords

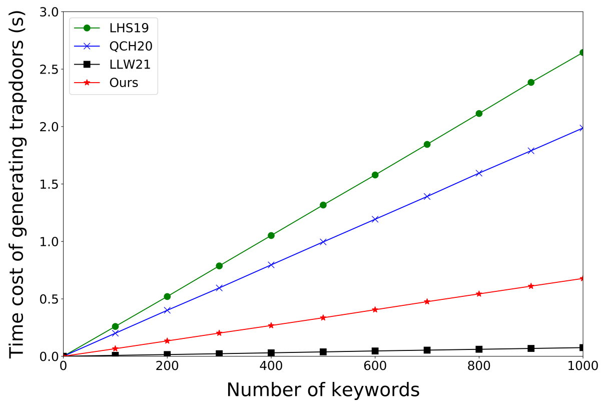

Fig. 3. Computation cost of generating trapdoor

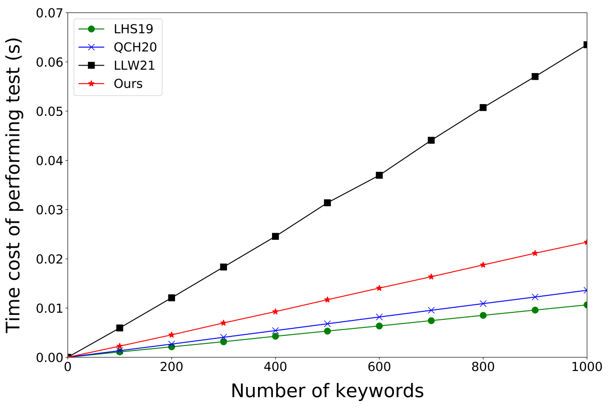

Fig. 4. Computation cost of performing test

TABLE I
FEATURES OF COMPARED SCHEMES

Schemes	**Type**	**Certificateless**	**Implicit Authentication**	**No Key Escrow**	**No Key Distribution**	**No Secure Channel**[1]
HMZKL18 [28]	CLAEKS	Yes	Yes	Yes	No	No
LHSYS19 [23]	IBAEKS	Yes	Yes	No	No	No
CWZH19 [20]	PAEKS	No	No	Yes	No	Yes
LLZ19 [25]	CBAEKS	No	Yes	Yes	Yes	Yes
PSE20 [26]	CLAEKS	Yes	Yes	Yes	No	No
QCH20 [21]	PAEKS	No	No	Yes	No	Yes
LLW21 [24]	CBAEKS	No	Yes	Yes	Yes	Yes
Ours	ICA-IBSE	Yes[2]	Yes	Yes	No	No

[1]channel between KGC and DO/DU.
[2]certificates in ICA-IBSE are used to generate secret keys only once; by contrast, certificates in PAEKS are continually used for authentication, and those in CBAEKS are continually used for encrypting or generating the Trapdoor.

TABLE II
COMMUNICATION AND COMPUTATIONAL COST OF COMPARED SCHEMES

Schemes	**Communication Cost**				**Computation Cost**		
	Public Key	Secret Key	Ciphertext	Trapdoor	KeywordEnc	TrapdoorGen	Test
HMZKL18 [28]	$2\|\mathbb{G}_1\| + \|ID\|$	$2\|\mathbb{Z}_q\|$	$2\|\mathbb{G}_1\|$	$\|\mathbb{G}_T\|$	$5T_E + T_H + 2T_h$	$T_P + 3T_E + T_H + 2T_h$	$T_P + T_E$
LHSYS19 [23]	$\|ID\|$	$\|\mathbb{G}_1\|$	$2\|\mathbb{G}_1\| + \|\mathbb{G}_T\|$	$2\|\mathbb{G}_1\|$	$2T_P + 3T_E + 2T_H$	$T_P + 2T_E + 2T_H$	$2T_P + 2T_E$
CWZH19 [20]	$\|\mathbb{G}_1\|$	$\|\mathbb{Z}_q\|$	$3\|\mathbb{G}_1\|$	$2\|\mathbb{G}_1\|$	$5T_E + T_h$	$5T_E + T_h$	T_E
LLZ19 [25]	$2\|\mathbb{G}_1\|$	$\|\mathbb{Z}_q\|$	$3\|\mathbb{G}_1\| + \|h\|$	$\|\mathbb{G}_1\|$	$2T_P + 3T_E + T_H + 4T_h$	$T_P + 3T_E + 2T_H + 3T_h$	$T_P + T_h$
PSE20 [26]	$2\|\mathbb{G}_1\| + \|ID\|$	$\|\mathbb{G}_1\| + \|\mathbb{Z}_q\|$	$2\|\mathbb{G}_1\|$	$\|h\|$	$T_P + 3T_E + T_H$	$T_P + T_E + T_H$	T_E
QCHLZ20 [21]	$\|\mathbb{G}_1\|$	$\|\mathbb{Z}_q\|$	$\|\mathbb{G}_1\| + \|h\|$	$\|\mathbb{G}_1\|$	$T_P + 3T_E + T_H + T_h$	$2T_E + T_H$	$T_P + T_h$
LLW21 [24]	$3\|\mathbb{G}_1\|$	$2\|\mathbb{Z}_q\|$	$\|\mathbb{G}_1\| + 2\|\mathbb{Z}_q\| + \|h\|$	$\|\mathbb{G}_1\| + \|\mathbb{Z}_q\|$	$5T_E + 4T_h$	$2T_E + 2T_h$	$2T_E + 2T_h$
Ours	$\|ID\|$	$\|\mathbb{G}_1\|$	$2\|\mathbb{G}_1\|$	$\|h\|$	$T_P + 2T_E + T_H + 2T_h$	$T_P + T_H + T_h$	$T_E + T_h$

$|\mathbb{G}_1|, |\mathbb{G}_T|, |\mathbb{Z}_q|$, $|ID|$, and $|h|$ denote the bit length of the elements of $\mathbb{G}_1, \mathbb{G}_T, \mathbb{Z}_q$, identity ID, and output of hash function, respectively.
T_P, T_E, T_H, and T_h denote the time costs of pairing, modular exponential, hash-to-point, and hash operations, respectively.

TABLE III
BIT LENGTH OF ELEMENTS AND RUNNING TIME OF OPERATIONS

Bit-length (bit)				**Running time (ms)**			
$\|\mathbb{G}_1\|$	$\|\mathbb{G}_T\|$	$\|\mathbb{Z}_q\|$	$\|h\|$	T_H	T_h	T_P	T_E
512	1024	160	256	1.858	0.014	0.002	0.000016

TABLE IV
EXPERIMENTAL PLATFORM

Description	**Data**
CPU	AMD Ryzen 5-2600 3.4GHz
CPU processor number	6
Operation system	Ubuntu 18.04
Linux kernel version	5.3.0-59-generic
Random access memory	16.3GB
Solid state disk	232.9GB

schemes [24], [25], the size of the public key is the sum of the sizes of the public key and certificate. For computational cost, we use the symbols *KeywordEnc* and *TrapdoorGen* to denote the cost of encryption and trapdoor generation per keyword, respectively. We also use the symbol *Test* to denote the cost of performing a test of whether a ciphertext is matched with a trapdoor. In this theoretical comparison, we consider only four time-consuming operations, namely bilinear pairing, modular exponential, hash-to-point, and hash operations, which are denoted as T_P, T_E, T_H, and T_h, respectively.

B. Performance Evaluation

To evaluate the performance of our scheme, we fully implement our proposed scheme and the LHSYS19 [23], QCHLZ20 [21], and LLW21 [24] schemes. The source codes are available at `https://github.com/zyliu-crypto/ICA-IBSE`. We conduct the experiment in the environment described in Table IV. Specifically, we use the SHA3-256 library[3] for the general cryptographic hash function, and we use the PBC library[4] for operations over groups. In particular, we adopt Type-A pairing with a 160-bit group order, 512-bit group element for $\mathbb{G}_1$, and 1024-bit group element for $\mathbb{G}_T$. Table III details the time cost per operation and the space taken up by different elements.

The results are presented in Figs. 2, 3, and 4, indicating that our scheme efficiently encrypts keywords and generates trapdoors. These results demonstrate that our scheme, although slower than the LLW21 scheme [24], is faster than the LHSYS19 [23] and QCHLZ20 [21] schemes in encrypting keywords and generating trapdoors. However, although our scheme is theoretically faster than the QCHLZ20 scheme [21] in *Test* (*i.e.,* $T_E + T_h < T_p + T_h$ in Table II), during the implementation, we must perform additional operations to convert the output of the hash function to some element in $\mathbb{Z}_q^*$, which takes time. Therefore, our scheme is faster than the LLW21 scheme [24] only in terms of *Test*.

[3]https://github.com/brainhub/SHA3IUF
[4]https://crypto.stanford.edu/pbc/

VII. Conclusion

In this paper, we present a novel ICA-IBSE scheme that masters the trade-off between efficiency (in terms of low storage requirement) and convenience (which IBAEKS achieves by avoiding the key escrow problem). A concrete framework is presented, and security proofs are provided, which demonstrate that the ICA-IBSE scheme can resist MCKAs and IKGAs under random oracles. Moreover, we experimentally verify that our scheme not only reduces storage requirements but is also practicable relative to its state-of-the-art counterparts.

Acknowledgment

This research was supported by the Ministry of Science and Technology, Taiwan (ROC), under Project Numbers MOST 108-2218-E-004-001-, 108-2218-E-004-002-MY2, and 109-2218-E-011-007-.

References

[1] H. Lasi, P. Fettke, H. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," *Bus. Inf. Syst. Eng.*, vol. 6, no. 4, pp. 239–242, 2014. [Online]. Available: https://doi.org/10.1007/s12599-014-0334-4

[2] D. X. Song, D. A. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *2000 IEEE Symposium on Security and Privacy, Berkeley, California, USA, May 14-17, 2000*. IEEE Computer Society, 2000, pp. 44–55. [Online]. Available: https://doi.org/10.1109/SECPRI.2000.848445

[3] M. Zhang, Y. Chen, and J. Huang, "SE-PPFM: A searchable encryption scheme supporting privacy-preserving fuzzy multikeyword in cloud systems," *IEEE Syst. J.*, 2020. [Online]. Available: https://doi.org/10.1109/JSYST.2020.2997932

[4] Z. Fu, X. Wu, C. Guan, X. Sun, and K. Ren, "Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 12, pp. 2706–2716, 2016. [Online]. Available: https://doi.org/10.1109/TIFS.2016.2596138

[5] Z. Fu, F. Huang, K. Ren, J. Weng, and C. Wang, "Privacy-preserving smart semantic search based on conceptual graphs over encrypted outsourced data," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 8, pp. 1874–1884, 2017. [Online]. Available: https://doi.org/10.1109/TIFS.2017.2692728

[6] C. Guo, W. Liu, X. Liu, and Y. Zhang, "Secure similarity search over encrypted non-uniform datasets," *IEEE Trans. Cloud Comput.*, 2020. [Online]. Available: https://doi.org/10.1109/TCC.2020.3000233

[7] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, ser. Lecture Notes in Computer Science, C. Cachin and J. Camenisch, Eds., vol. 3027. Springer, 2004, pp. 506–522. [Online]. Available: https://doi.org/10.1007/978-3-540-24676-3_30

[8] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi, "Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions," in *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, ser. Lecture Notes in Computer Science, V. Shoup, Ed., vol. 3621. Springer, 2005, pp. 205–222. [Online]. Available: https://doi.org/10.1007/11535218_13

[9] L. Xu, X. Yuan, R. Steinfeld, C. Wang, and C. Xu, "Multi-writer searchable encryption: An lwe-based realization and implementation," in *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security, AsiaCCS 2019, Auckland, New Zealand, July 09-12, 2019*, S. D. Galbraith, G. Russello, W. Susilo, D. Gollmann, E. Kirda, and Z. Liang, Eds. ACM, 2019, pp. 122–133. [Online]. Available: https://doi.org/10.1145/3321705.3329814

[10] R. Behnia, M. O. Ozmen, and A. A. Yavuz, "Lattice-based public key searchable encryption from experimental perspectives," *IEEE Trans. Dependable Secur. Comput.*, vol. 17, no. 6, pp. 1269–1282, 2020. [Online]. Available: https://doi.org/10.1109/TDSC.2018.2867462

[11] Y. Miao, J. Ma, X. Liu, J. Weng, H. Li, and H. Li, "Lightweight fine-grained search over encrypted data in fog computing," *IEEE Trans. Serv. Comput.*, vol. 12, no. 5, pp. 772–785, 2019. [Online]. Available: https://doi.org/10.1109/TSC.2018.2823309

[12] Y. Miao, J. Ma, X. Liu, X. Li, Q. Jiang, and J. Zhang, "Attribute-based keyword search over hierarchical data in cloud computing," *IEEE Trans. Serv. Comput.*, vol. 13, no. 6, pp. 985–998, 2020. [Online]. Available: https://doi.org/10.1109/TSC.2017.2757467

[13] J. W. Byun, H. S. Rhee, H. Park, and D. H. Lee, "Off-line keyword guessing attacks on recent keyword search schemes over encrypted data," in *Secure Data Management, Third VLDB Workshop, SDM 2006, Seoul, Korea, September 10-11, 2006, Proceedings*, ser. Lecture Notes in Computer Science, W. Jonker and M. Petkovic, Eds., vol. 4165. Springer, 2006, pp. 75–83. [Online]. Available: https://doi.org/10.1007/11844662_6

[14] R. Chen, Y. Mu, G. Yang, F. Guo, and X. Wang, "A new general framework for secure public key encryption with keyword search," in *Information Security and Privacy - 20th Australasian Conference, ACISP 2015, Brisbane, QLD, Australia, June 29 - July 1, 2015, Proceedings*, ser. Lecture Notes in Computer Science, E. Foo and D. Stebila, Eds., vol. 9144. Springer, 2015, pp. 59–76. [Online]. Available: https://doi.org/10.1007/978-3-319-19962-7_4

[15] ——, "Dual-server public-key encryption with keyword search for secure cloud storage," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 4, pp. 789–798, 2016. [Online]. Available: https://doi.org/10.1109/TIFS.2015.2510822

[16] R. Chen, Y. Mu, G. Yang, F. Guo, X. Huang, X. Wang, and Y. Wang, "Server-aided public key encryption with keyword search," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 12, pp. 2833–2842, 2016. [Online]. Available: https://doi.org/10.1109/TIFS.2016.2599293

[17] R. Tso, K. Huang, Y. Chen, S. M. M. Rahman, and T. Wu, "Generic construction of dual-server public key encryption with keyword search on cloud computing," *IEEE Access*, vol. 8, pp. 152 551–152 564, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.3017745

[18] Q. Huang and H. Li, "An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks," *Inf. Sci.*, vol. 403, pp. 1–14, 2017. [Online]. Available: https://doi.org/10.1016/j.ins.2017.03.038

[19] Y. Zhang, C. Xu, J. Ni, H. Li, and X. S. Shen, "Blockchain-assisted public-key encryption with keyword search against keyword guessing attacks for cloud storage," *IEEE Trans. Cloud Comput.*, 2019. [Online]. Available: https://doi.org/10.1109/TCC.2019.2923222

[20] B. Chen, L. Wu, S. Zeadally, and D. He, "Dual-server public-key authenticated encryption with keyword search," *IEEE Trans. Cloud Comput.*, 2019. [Online]. Available: https://doi.org/10.1109/TCC.2019.2945714

[21] B. Qin, Y. Chen, Q. Huang, X. Liu, and D. Zheng, "Public-key authenticated encryption with keyword search revisited: Security model and constructions," *Inf. Sci.*, vol. 516, pp. 515–528, 2020. [Online]. Available: https://doi.org/10.1016/j.ins.2019.12.063

[22] Z. Liu, Y. Tseng, R. Tso, and M. Mambo, "Quantum-resistant public-key authenticated encryption with keyword search for industrial internet of things," *IACR Cryptol. ePrint Arch.*, 2020. [Online]. Available: https://eprint.iacr.org/2020/955

[23] H. Li, Q. Huang, J. Shen, G. Yang, and W. Susilo, "Designated-server identity-based authenticated encryption with keyword search for encrypted emails," *Inf. Sci.*, vol. 481, pp. 330–343, 2019. [Online]. Available: https://doi.org/10.1016/j.ins.2019.01.004

[24] Y. Lu, J. Li, and F. Wang, "Pairing-free certificate-based searchable encryption supporting privacy-preserving keyword search function for iiots," *IEEE Trans. Ind. Informatics*, vol. 17, no. 4, pp. 2696–2706, 2021. [Online]. Available: https://doi.org/10.1109/TII.2020.3006474

[25] Y. Lu, J. Li, and Y. Zhang, "Secure channel free certificate-based searchable encryption withstanding outside and inside keyword guessing attacks," *IEEE Trans. Serv. Comput.*, 2019. [Online]. Available: https://doi.org/10.1109/TSC.2019.2910113

[26] N. Pakniat, D. Shiraly, and Z. Eslami, "Certificateless authenticated encryption with keyword search: Enhanced security model and a concrete construction for industrial iot," *J. Inf. Secur. Appl.*, vol. 53, p. 102525, 2020. [Online]. Available: https://doi.org/10.1016/j.jisa.2020.102525

[27] X. Liu, H. Li, G. Yang, W. Susilo, J. Tonien, and Q. Huang, "Towards enhanced security for certificateless public-key authenticated encryption with keyword search," in *Provable Security - 13th International Conference, ProvSec 2019, Cairns, QLD, Australia, October 1-4, 2019, Proceedings*, ser. Lecture Notes in Computer Science, R. Steinfeld and

T. H. Yuen, Eds., vol. 11821. Springer, 2019, pp. 113–129. [Online]. Available: https://doi.org/10.1007/978-3-030-31919-9_7

[28] 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 Trans. Ind. Informatics*, vol. 14, no. 8, pp. 3618–3627, 2018. [Online]. Available: https://doi.org/10.1109/TII.2017.2771382

[29] S. S. M. Chow, "Removing escrow from identity-based encryption," in *Public Key Cryptography - PKC 2009, 12th International Conference on Practice and Theory in Public Key Cryptography, Irvine, CA, USA, March 18-20, 2009. Proceedings*, ser. Lecture Notes in Computer Science, S. Jarecki and G. Tsudik, Eds., vol. 5443. Springer, 2009, pp. 256–276. [Online]. Available: https://doi.org/10.1007/978-3-642-00468-1_15

[30] K. Emura, S. Katsumata, and Y. Watanabe, "Identity-based encryption with security against the KGC: A formal model and its instantiation from lattices," in *Computer Security - ESORICS 2019 - 24th European Symposium on Research in Computer Security, Luxembourg, September 23-27, 2019, Proceedings, Part II*, ser. Lecture Notes in Computer Science, K. Sako, S. A. Schneider, and P. Y. A. Ryan, Eds., vol. 11736. Springer, 2019, pp. 113–133. [Online]. Available: https://doi.org/10.1007/978-3-030-29962-0_6

[31] E. Rescorla, "The transport layer security (TLS) protocol version 1.3," *RFC*, vol. 8446, pp. 1–160, 2018. [Online]. Available: https://doi.org/10.17487/RFC8446

[32] K. Emura, S. Katsumata, and Y. Watanabe, "Identity-based encryption with security against the KGC: A formal model and its instantiation from lattices," in *Computer Security - ESORICS 2019 - 24th European Symposium on Research in Computer Security, Luxembourg, September 23-27, 2019, Proceedings, Part II*, ser. Lecture Notes in Computer Science, K. Sako, S. A. Schneider, and P. Y. A. Ryan, Eds., vol. 11736. Springer, 2019, pp. 113–133. [Online]. Available: https://doi.org/10.1007/978-3-030-29962-0_6

[33] T. Okamoto and D. Pointcheval, "The gap-problems: A new class of problems for the security of cryptographic schemes," in *Public Key Cryptography, 4th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2001, Cheju Island, Korea, February 13-15, 2001, Proceedings*, ser. Lecture Notes in Computer Science, K. Kim, Ed., vol. 1992. Springer, 2001, pp. 104–118. [Online]. Available: https://doi.org/10.1007/3-540-44586-2_8

[34] A. Joux and K. Nguyen, "Separating decision diffie-hellman from computational diffie-hellman in cryptographic groups," *J. Cryptol.*, vol. 16, no. 4, pp. 239–247, 2003. [Online]. Available: https://doi.org/10.1007/s00145-003-0052-4