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

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*—cloud systems, identity-based searchable encryption, identity-certifying authority, 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 on shipments, patient records, and even factory inventory are highly sensitive, and companies may be reluctant to directly upload them to cloud systems that they cannot fully trust. Consequently, data are often encrypted before being uploaded to cloud systems in order to avoid information leakage, but such encrypted data pose a computational challenge for cloud systems.

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 systems. Subsequently, the DU can generate a trapdoor for specified keywords and submit them to the cloud systems 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], [4], [5], [6], [7]. 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.* [8] 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 [9], [10], [11], [12], [13], [14], [15], [16]. However, in 2006, Byun *et al.* [17] observed that because the entropy of keywords is low, any malicious party, through a so-called keyword guessing attacks, 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 attacks launched by a malicious cloud server (CS) as part of a so-called *insider* keyword guessing attacks (IKGA), Chen *et al.* [18], [19], [20] have first proposed solutions for dual-server setting, and their methods were improved upon by Tso *et al.* [21].

Huang and Li [22] recently introduced the concept of public-key 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 [22], scholars have proposed several PAEKS schemes [23], [24], [25], [26], [27], [28],

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
Y.-C. Chen is with the Department of Computer Science and Engineering, Yuan Ze University, Taoyuan 32003, Taiwan
E-mail: wycchen@saturn.yzu.edu.tw
M. Mambo is with the Institute of Science and Engineering, Kanazawa University, Kanazawa 920-1192, Japan.
E-mail: mambo@ec.t.kanazawa-u.ac.jp
The authors thank the anonymous reviewers for their insightful suggestions on this work. This research was supported by the Ministry of Science and Technology, Taiwan (ROC), under project numbers MOST 108-2218-E-004-002-MY2, MOST 109-2628-E-155-001-MY3, MOST 109-2221-E-004-011-MY3, MOST 109-3111-8-004-001-, MOST 110-2218-E-004-001-MBK, and MOST 110-2221-E-004-003-. (Corresponding author: R. Tso)

[29]. 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) [30]: In IBAEKS schemes, the user can apply to a trusted key generation center (KGC) for the secret key of a specific identity; he/she 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) [31], [32]: In CBAEKS schemes, the user can apply to a trusted certificate authority for the certificate of a pair comprising a specific identity and public key. Unlike the certificates used in PAEKS schemes, the certificates 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) [33], [34], [35], [36]: 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 his/her 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 [30], because the KGC can gain access to any user's secret key, key escrow problems occur when the KGC is malicious. In addition, although CBAEKS [31], [32] and CLAEKS [33], [34], [35], [36] solve the key escrow problem, the users in these schemes require an additional generated certificate or a public–secret value pair to encrypt keywords or generate trapdoors. Consequently, users require more storage space to store certificates and keys. It becomes less convenient for the users in CBAEKS and CLAEKS schemes than in IBAEKS schemes, because the users have to use additional information instead of merely using their identities to encrypt keywords and generate trapdoors. 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.* [37] and Emura *et al.* [38], who have used an ICA in identity-based encryption, we propose our ICA-IBSE scheme, which inherits the advantages of IBAEKS in terms of convenience and storage requirements as well as 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.
- We further provide a theoretical comparison and performance evaluation of our scheme against state-of-the-art schemes [24], [25], [30], [31], [32], [33], [35]. 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. Preliminaries

In this section, we introduce the requisite background, including notations, digital signature, symmetric bilinear groups, and gap bilinear Diffie–Hellman assumption.

A. Notations

For simplicity and readability, TABLE I describes the notations used throughout the paper.

TABLE I
Notations and Descriptions

Notations	Descriptions
λ	Security parameter
pp	Public parameter
$\mathcal{ID}, \mathcal{W}, \mathcal{M}$	Identity space, Keyword space, Message space
q	A big prime number
$[a, b]$	Set $\{a, a+1, \cdots, b\}$ (omit a if $a = 1$)
$\mathbb{Z}_q^*$	Integers in $[1, q-1]$ which are relatively prime to q
$\mathbb{G}_1, \mathbb{G}_T$	Bilinear groups with order q
$\mathbb{G}_{ec}$	Elliptic curve group with order q
g	A generator of $\mathbb{G}_1$
$\hat{e}$	Bilinear pairing
H, h_1, h_2	Cryptographic hash functions
IKGA	Insider keyword guessing attacks
MCKA	Multichosen keyword attacks
ICA, KGC	Identity-certifying authority, Key generation center
CS, DO, DU	Cloud server, Data owner, Data user
ID, DO, DU	The identity of ID, DO, and DU, respectively
pk_{ICA}, sk_{ICA}	The public key and secret key of ICA
pk_{KGC}, sk_{KGC}	The public key and secret key of KGC
sk_{ID}	The secret key of identity ID
$cert_{ID}, tf_{ID}$	The certificate and trapdoor of identity ID
w	A keyword
ct_w	A ciphertext associated with keyword w
td_w	A trapdoor associated with keyword w
$\perp$	A null symbol
$\mathcal{A}, \mathcal{B}$	The polynomial time adversary and challenger
$\mathcal{A}^{\mathcal{O}(\cdot)}(x)$	$\mathcal{A}$ with input x can access the black-box oracle $\mathcal{O}$
$y \leftarrow F(x)$	Storing the output of an algorithm F with input x to the variable y
$s \xleftarrow{\$} S$	Sampling an element s from set S uniformly at random
$\|K\|$	Bit length of elements of K
$\mathsf{negl}(\lambda)$	An arbitrary function f is negligible in λ, where $f(\lambda) = o(\lambda^{-c})$ for every fixed constant c
Sig	Digital signature scheme
$\mathsf{vk}_{\mathsf{Sig}}, \mathsf{sk}_{\mathsf{Sig}}$	Verification key and Signing key of Sig
m, σ_{Sig}	Message, Signature
Q_{key}	The amount of instances obtained by querying $\mathsf{Issue\ Key\ KGC\ Oracle}$
$IDList$	Storing the corresponding identity of the secret key that is returned by $\mathsf{Issue\ Key\ KGC\ Oracle}$
$(*)$	Any element in a tuple

B. Digital Signature

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 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 equation $\Pr[\mathsf{Sig.Verify}(\mathsf{vk}_{\mathsf{Sig}}, m, \sigma_{\mathsf{Sig}}) = 1] = 1$ holds. In addition, we consider that Sig achieves existential unforgeability against adaptive chosen message attacks (EU-CMA) if, for any $\mathcal{A}$, the advantage of $\mathcal{A}$ wins the EU-CMA game $\mathsf{Adv}^{\mathsf{EU\text{-}CMA}}_{\mathsf{Sig},\mathcal{A}}(\lambda) := \Pr[\mathsf{Sig.Verify}(\mathsf{vk}_{\mathsf{Sig}}, m^*, \sigma^*_{\mathsf{Sig}}) = 1]$ 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
$(\mathsf{vk}_{\mathsf{Sig}}, \mathsf{sk}_{\mathsf{Sig}}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$; $(m^*, \sigma^*_{\mathsf{Sig}}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}(\cdot)}(\mathsf{vk}_{\mathsf{Sig}})$;

C. 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.

D. Gap Bilinear Diffie–Hellman Assumption [39], [40]

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:

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

where q_{DBDH} denotes the maximum number of queries by $\mathcal{A}$ to $\mathcal{O}_{\mathsf{DBDH}}$, and $\mathcal{O}_{\mathsf{DBDH}}$ 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 otherwise.

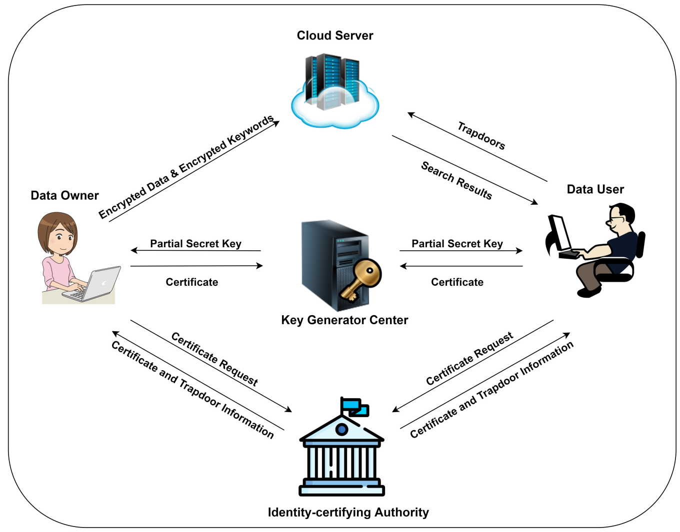


Fig. 1. The proposed ICA-IBSE framework

III. Description of Entities in ICA-IBSE

The ICA-IBSE comprises five entities, namely the Identity-certifying Authority (ICA), Key Generation Center (KGC), Cloud Server (CS), Data Owner (DO), and Data User (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 keys without knowing any information about their identities.
- 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 his/her certificate from the ICA and further generates his/her partial secret key by interacting with the KGC. Finally, the DO obtains his/her 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 his/her secret key as the DO does. Subsequently, the DU can issue trapdoors, generated by using his/her secret key, to the CS to retrieve the encrypted data associated with the specified keyword.

In the following analysis, we consider adversary threats from different perspectives.

- The DO and DU are fully trusted, as all of the case in PAEKS schemes, because DO and DU know the secret information (*i.e.,* keywords). Therefore, the DO and DU is also unable to collude with other parties to reveal their secret keys, or the keyword privacy can be directly compromised by other parties.
- The ICA and KGC cannot collude with each other, because it is impossible for KGC to issue any secret

key to a user whose identity cannot be directly or indirectly identified. Therefore, to enable KGC to generate secret keys, there must be some party (*e.g.,* ICA) that can privately authenticate the user's identity. Then, the user can indirectly validate his/her identity to the KGC. However, if ICA and KGC are collude with each other, they can adaptively generate any user's secret keys. Therefore, we need to assume that ICA and KGC cannot collude. Although this restriction has to be added, it is still in accordance with the real scenarios. For example, supposing ICA is a government-issued certificate party that sends a certificate showing the user is authenticated, with this certificate, the service provider (*e.g.,* KGC) can provide the secret key to the user only by verifying the certificate without further knowing the user's true identity, so the service provider is unable to generate the secret key itself. In general, there are usually government regulators to ensure that the government is not colluding with service providers.

- 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 (*e.g.,* 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 [41]).

IV. Definition and Security Models of ICA-IBSE

A. Definition

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 $\mathcal{ID}$ of ICA-IBSE are defined by the system parameter pp.

$\mathsf{ICA\text{-}Setup}(pp)$: This algorithm is executed by ICA that takes the system parameter pp as input, and outputs the public–secret key pair (pk_{ICA}, sk_{ICA}) of the ICA.

$\mathsf{KGC\text{-}Setup}(pp)$: This algorithm is executed by KGC that takes the system parameter pp as input, and outputs the public–secret key pair (pk_{KGC}, sk_{KGC}) of the KGC.

$\mathsf{ICA\text{-}Cert}(pp, pk_{ICA}, sk_{ICA}, ID)$: This algorithm is executed by ICA that takes the system parameter pp, ICA's public key pk_{ICA}, ICA's secret key sk_{ICA}, and a user's identity $ID \in \mathcal{ID}$ as inputs, and outputs a certificate $cert_{ID}$ and a trapdoor information tf_{ID}, which are sent to the user 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 $\mathsf{M}_1 \leftarrow \mathsf{User\text{-}Obtain\text{-}Key}(pp, pk_{KGC}, ID, cert_{ID}, tf_{ID})$ and submits it to the KGC. Subsequently, the KGC returns a second-round message $\mathsf{M}_2 \leftarrow \mathsf{KGC\text{-}Issue\text{-}Key}(pp, pk_{KGC}, sk_{KGC}, pk_{ICA})$ to the user. At the end of the protocol, the user can locally output a secret key sk_{ID} or $\perp$.

$\mathsf{Encrypt}(pp, sk_{DO}, DU, w)$: This algorithm is executed by DO that takes the system parameter pp, DO's secret key sk_{DO}, DU's identity DU, and a keyword $w \in \mathcal{W}$ as inputs, and outputs a searchable ciphertext ct_w associated with keyword w is the output.

$\mathsf{Trapdoor}(pp, DO, sk_{DU}, w)$: This algorithm is executed by DU that takes the system parameter pp, DO's identity DO, DU's secret key sk_{DU}, and a keyword $w \in \mathcal{W}$ as inputs, and outputs a trapdoor td_w associated with keyword w.

$\mathsf{Test}(pp, ct_w, td_w)$: This algorithm is executed by CS that takes the system parameter pp, a searchable ciphertext ct_w, and a trapdoor td_w 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 III, we revise the security model in [25] and [38] to account for multichosen keyword attacks (MCKA) and IKGA in the ICA-IBSE framework. Here, MCKA, recently introduced by Qin *et al.* [25], ensures that no adversary can obtain any information on keywords from two tuples of encrypted keywords; IKGA ensures that no insider adversary (*e.g.,* the CS) can obtain any information on keywords from the trapdoor, even when an insider can conduct tests.

In ICA-IBSE framework, the CS can directly obtain the entirety of ciphertext and all the trapdoors from DO and DU; hence, intuitively, the CS has greater attack capability relative to eavesdroppers on the channel. In addition, ICA and KGC are not fully trusted party in this framework. Therefore, we not only define the attacks from CS (*i.e.,* MCKA-CS and IKGA-CS), but further define additionally security games to model the attacks from ICA and KGC (*i.e.,* MCKA-ICA, IKGA-ICA, MCKA-KGC, and IKGA-KGC) for the real scenarios.

Informally, in these security models, the prefix (MCKA or IKGA) means that the models are related to the MCKA security or IKGA security, while the suffix (ICA, CS, and KGC) means that which malicious party is being modeled. In more detail, for malicious ICA, since ICA cannot collude to KGC in ICA-IBSE framework, the malicious ICA in MCKA-ICA and IKGA-ICA models is unable to obtain any identities' secret key by querying oracles. As for malicious CS, it is allowed to query user's partial secret key (*i.e.,* M_2) in MCKA-CS and IKGA-CS models. As for malicious KGC, to model the ability of KGC that can generate any user's partial secret key, malicious KGC is given the KGC's secret key in MCKA-KGC and IKGA-KGC models. In addition, the malicious KGC is allowed to query user's certificate (*i.e.,* M_1) and further generates any user's secret key. However, since KGC cannot obtain the identity information of the user in ICA-IBSE, the malicious KGC also cannot obtain the identity information corresponding to the certificate when querying the oracles.

Before presenting the formal security models, we first define the following oracles that are simulated by the challenger $\mathcal{B}$ for the adversary $\mathcal{A}$:

- Certificate Oracle $\mathcal{O}_{\mathsf{cert}}$: For any identity $ID \in \mathcal{ID}$, $\mathcal{B}$ outputs $(cert_{ID}, tf_{ID}) \leftarrow \mathsf{ICA\text{-}Cert}(pp, pk_{ICA}, sk_{ICA}, ID)$ to $\mathcal{A}$.
- Secret Key Oracle $\mathcal{O}_{\mathsf{sk}}$: For any first-round message M_1, $\mathcal{B}$ runs $\mathsf{M}_2 \leftarrow \mathsf{KGC\text{-}Issue\text{-}Key}(pp, pk_{KGC}, sk_{KGC}, pk_{ICA})$ and returns M_2 to $\mathcal{A}$.
- Ciphertext Oracle $\mathcal{O}_{\mathsf{ct}}$: For any keyword $w \in \mathcal{W}$, $\mathcal{B}$ outputs $ct_w \leftarrow \mathsf{Encrypt}(pp, sk_{DO}, DU, w)$ to $\mathcal{A}$.
- Trapdoor Oracle $\mathcal{O}_{\mathsf{td}}$: For any keyword $w \in \mathcal{W}$, $\mathcal{B}$ outputs $td_w \leftarrow \mathsf{Trapdoor}(pp, DO, sk_{DU}, w)$ to $\mathcal{A}$.
- Issue Key KGC Oracle $\mathcal{O}_{\mathsf{ik\text{-}KGC}}$: When $\mathcal{A}$ makes this query, $\mathcal{B}$ randomly chooses $ID \xleftarrow{\$} \mathcal{ID}$ and computes $(cert_{ID}, tf_{ID}) \leftarrow \mathsf{ICA\text{-}Cert}(pp, pk_{ICA}, sk_{ICA}, ID)$. In addition, $\mathcal{B}$ runs $\mathsf{M}_1 \leftarrow \mathsf{User\text{-}Obtain\text{-}Key}(pp, pk_{KGC}, ID, cert_{ID}, tf_{ID})$, and returns M_1 to $\mathcal{A}$. Furthermore, $\mathcal{B}$ stores ID to $IDList$, and updates $Q_{key} \leftarrow Q_{key} + 1$. Here, Q_{key} is the amount of instances obtained by querying $\mathcal{O}_{\mathsf{ik\text{-}KGC}}$, and $IDList$ is a list of identities whose corresponding M_1 values have been obtained by the adversary through querying $\mathcal{O}_{\mathsf{ik\text{-}KGC}}$.
- Ciphertext KGC Oracle $\mathcal{O}_{\mathsf{ct\text{-}KGC}}$: For any keyword $w \in \mathcal{W}$, DO index doi, and DU index dui, $\mathcal{B}$ first checks whether $doi \in [Q_{key}]$ and $dui \in [Q_{key}]$. If not, $\mathcal{B}$ forces $\mathcal{A}$ to output a random bit $b' \in \{0,1\}$. Otherwise, $\mathcal{B}$ 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 $\mathcal{A}$.
- Trapdoor KGC Oracle $\mathcal{O}_{\mathsf{td\text{-}KGC}}$: For any keyword $w \in \mathcal{W}$, DO index doi, and DU index dui, $\mathcal{B}$ first checks whether $doi \in [Q_{key}]$ and $dui \in [Q_{key}]$. If not, $\mathcal{B}$ forces $\mathcal{A}$ to output a random bit $b' \in \{0,1\}$. Otherwise, $\mathcal{B}$ 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 $\mathcal{A}$.

The following we give a formal description of these security models. In particular, 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 unable to issue queries to $\mathcal{O}_{\mathsf{cert}}$ on the challenged identities $DO, DU \in \mathcal{ID}$. Furthermore, the adversary cannot issue queries to oracles $\mathcal{O}_{\mathsf{ct}}$ on the challenged keywords $w_{b,i} \in \mathcal{W}$ and cannot issue queries to oracles $\mathcal{O}_{\mathsf{td}}$ 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

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

is negligible for any $\mathcal{A}$, and we consider that ICA-IBSE is IKGA-CS secure if the advantage

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

is negligible for any $\mathcal{A}$.

MCKA-CS / IKGA-CS Game

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

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, the adversary cannot access to $\mathcal{O}_{\mathsf{sk}}$. In addition, the adversary is also unable to issue queries to oracles $\mathcal{O}_{\mathsf{ct}}$ on challenged keywords $w_{b,i} \in \mathcal{W}$ and oracles $\mathcal{O}_{\mathsf{td}}$ on 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-ICA secure if the advantage

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

is negligible for any $\mathcal{A}$, and we consider that ICA-IBSE is IKGA-ICA secure if the advantage

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

is negligible for any $\mathcal{A}$.

MCKA-ICA / IKGA-ICA Game

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

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 amount of instances obtained by querying Issue Key KGC Oracle $\mathcal{O}_{\text{ik-KGC}}$, and 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 cannot issue 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

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

is negligible for any $\mathcal{A}$, and we consider that ICA-IBSE is IKGA-KGC secure if the advantage

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

is negligible for any $\mathcal{A}$.

MCKA-KGC / IKGA-KGC Game

$$pp \leftarrow \mathsf{Setup}(1^\lambda); IDList = \emptyset; Q_{key} := 1;\ b \xleftarrow{\$} \{0,1\};$$
$$(pk_{ICA}, sk_{ICA}) \leftarrow \mathsf{ICA\text{-}Setup}(pp); (pk_{KGC}, sk_{KGC}) \leftarrow \mathsf{KGC\text{-}Setup}(pp);$$
$$(\tilde{w_0} = (w_{0,1}, \cdots, w_{0,n}), \tilde{w_1} = (w_{1,1}, \cdots, w_{1,n}), \alpha, \beta) \leftarrow$$
$$\mathcal{A}^{\mathcal{O}_{\text{ik-KGC}}(\cdot), \mathcal{O}_{\text{ct-KGC}}(\cdot), \mathcal{O}_{\text{td-KGC}}(\cdot)}(pp, pk_{ICA}, pk_{KGC}, sk_{KGC});$$
$$\tilde{ct}^* = (ct^*_1, \cdots, ct^*_n),$$
$$\text{where } ct^*_i \leftarrow \mathsf{Encrypt}(pp, sk_{IDList[\alpha]}, IDList[\beta], w_{b,i}),$$
$$\text{for } i = 1, \cdots, n;$$
$$b' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{ik-KGC}}(\cdot), \mathcal{O}_{\text{ct-KGC}}(\cdot), \mathcal{O}_{\text{td-KGC}}(\cdot)}(pp, pk_{ICA}, pk_{KGC}, sk_{KGC}, \tilde{ct}^*);$$
$$(w_0, w_1, \alpha, \beta) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{ik-KGC}}(\cdot), \mathcal{O}_{\text{ct-KGC}}(\cdot), \mathcal{O}_{\text{td-KGC}}(\cdot)}(pp, pk_{ICA}, pk_{KGC}, sk_{KGC})$$
$$td^* \leftarrow \mathsf{Trapdoor}(pp, IDList[\alpha], sk_{IDList[\beta]}, w_b);$$
$$b' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{ik-KGC}}(\cdot), \mathcal{O}_{\text{ct-KGC}}(\cdot), \mathcal{O}_{\text{td-KGC}}(\cdot)}(pp, pk_{ICA}, pk_{KGC}, sk_{KGC}, td^*);$$

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

In this section, we first propose a concrete scheme and then analyze the correctness and consistency of the proposed scheme.

A. Our Construction

Let $\mathcal{ID} = \mathbb{Z}_q^*$ and $\mathcal{W} = \{0,1\}^n$ for some n be the identity space and keyword space of the ICA-IBSE scheme, respectively. Let $DO \in \mathbb{Z}_q^*$ and $DU \in \mathbb{Z}_q^*$ be the identities of the DO and DU, respectively. 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 cyclic groups $\mathbb{G}_1, \mathbb{G}_T$ with a large prime order q; a generator $g \in \mathbb{G}_1$; a pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$; and three cryptographic hash functions $H : \mathbb{Z}_q^* \rightarrow \mathbb{G}_1$, $h_1 : \mathbb{Z}_q^* \times \mathbb{Z}_q^* \times \mathbb{G}_T \times \{0,1\}^n \rightarrow \mathbb{Z}_q^*$, and $h_2 : \mathbb{Z}_q^* \times \mathbb{G}_1 \rightarrow \mathbb{Z}_q^*$. It sets the system parameter to be

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

$\mathsf{ICA\text{-}Setup}(pp)$: The ICA runs $(\mathsf{vk_{Sig}}, \mathsf{sk_{Sig}}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$. It then outputs

$$pk_{ICA} := \mathsf{vk_{Sig}}; sk_{ICA} := \mathsf{sk_{Sig}}.$$

$\mathsf{KGC\text{-}Setup}(pp)$: The KGC picks $x \xleftarrow{\$} \mathbb{Z}_q^*$ 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 $y_{ID,1} \xleftarrow{\$} \mathbb{Z}_q^*$, and computes $u_{ID,1} = g^{y_{ID,1}}$. In addition, it computes $u_{ID,2} \in \mathbb{G}_1$ as $u_{ID,2} = u_{ID} \cdot u_{ID,1}$ and $\sigma_{\mathsf{Sig}} \leftarrow \mathsf{Sig.Sign}(\mathsf{sk_{Sig}}, u_{ID,2})$. 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 $\mathsf{M}_2 := y_{ID,2} = u^x_{ID,2}$. 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 DO randomly selects $r \xleftarrow{\$} \mathbb{Z}_q^*$, 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 otherwise.

B. Correctness and Consistency of ICA-IBSE

The following we analyze the correctness and consistency of the proposed ICA-IBSE scheme:

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

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

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.

VI. Security Analysis of ICA-IBSE

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

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 $pp = \{\lambda, \mathbb{G}_1, \mathbb{G}_T, \hat{e}, q, g, H, h_1, h_2\}$ according to the scheme. Subsequently, $\mathcal{B}$ 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 query to 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, $\mathcal{B}$ initials on four lists (cert-list, H-list, h_1-list, and h_2-list) and randomly chooses a bit $b \xleftarrow{\$} \{0, 1\}$. 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 steps under the following conditions:
  - If $i \notin \{\ell_1, \ell_2\}$: $\mathcal{B}$ randomly chooses $\mu_{ID_i} \xleftarrow{\$} \mathbb{Z}_q^*$, 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 $h \xleftarrow{\$} \mathbb{Z}_q^*$, 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 $\tilde{h} \xleftarrow{\$} \mathbb{Z}_q^*$, 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 $y_{ID,1} \xleftarrow{\$} \mathbb{Z}_q^*$ and computes $u_{ID,1} = g^{y_{ID,1}}$, $u_{ID,2} = u_{ID} \cdot u_{ID,1}$, 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 $r \xleftarrow{\$} \mathbb{Z}_q^*$.
  - 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 $h \xleftarrow{\$} \mathbb{Z}_q^*$ and add $((ID_i, ID_j, \perp, w), h)$ to h_1-list.
  - 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 $h \xleftarrow{\$} \mathbb{Z}_q^*$ 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 $r \xleftarrow{\$} \mathbb{Z}_q^*$.
  - 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 $h \xleftarrow{\$} \mathbb{Z}_q^*$ and add $((ID_i, ID_j, \perp, w), h)$ to h_1-list.
  - 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 $h \xleftarrow{\$} \mathbb{Z}_q^*$ 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 $r_i \xleftarrow{\$} \mathbb{Z}_q^*$. Second, search h_1-list for the tuple $((DO, DU, \perp, w_{b,i}), h_i)$; if no such entry is found, randomly choose $h_i \xleftarrow{\$} \mathbb{Z}_q^*$ 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 $c_{1,i}^* = g^{r_i}$, $c_{2,i}^* = g^{r_i \tilde{h_i}}$, where $\tilde{h_i}$ 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.

Analysis. 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 is the maximum number of queries that could be made to the H oracle for different inputs. Furthermore, because h_1 is modeled as a random oracle, $\mathcal{A}$'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 with a nonnegligible advantage ϵ, then there exists some $\mathcal{B}$ that can break the GBDH problem with 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 $h \xleftarrow{\$} \mathbb{Z}_q^*$ 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. Since malicious ICA cannot collude with the KGC, it cannot obtain any secret key information about the user, even if it has the ability to generate a potentially malicious ICA key pair. However, malicious CS has the ability to obtain the secret key information of any user except for the secret key information of the challenged identities by querying the secret key oracles. Therefore, malicious ICA can be viewed as a weaker variant of malicious CS. Consequently, Theorem 1 and Theorem 2 entail Theorem 3. Note that because $\mathcal{B}$ does not need to reply to the certificate query for $\mathcal{A}$, the signature scheme Sig does not need to 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 $pp = \{\lambda, \mathbb{G}_1, \mathbb{G}_T, \hat{e}, q, g, H, h_1, h_2\}$ according to the scheme. Subsequently, it sets $Q_{key} := 1$ to count the number of $\mathcal{A}$ queries to $\mathcal{O}_{\mathsf{ik\text{-}KGC}}$ and initials of an empty list $IDList$ to store the corresponding identity of the secret key returned by this oracle. $\mathcal{B}$ also runs $(pk_{ICA}, sk_{ICA}) \leftarrow \mathsf{ICA\text{-}Setup}(pp)$ and $(pk_{KGC}, sk_{KGC}) \leftarrow \mathsf{KGC\text{-}Setup}(pp)$ and chooses two arbitrary indices $\ell_1, \ell_2 \leq \mathsf{Max}_{Q_{key}}$, where $\mathsf{Max}_{Q_{key}}$ is the maximum number of queries that can query to the $\mathcal{O}_{\mathsf{ik\text{-}KGC}}$ 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 $b \xleftarrow{\$} \{0,1\}$. 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.

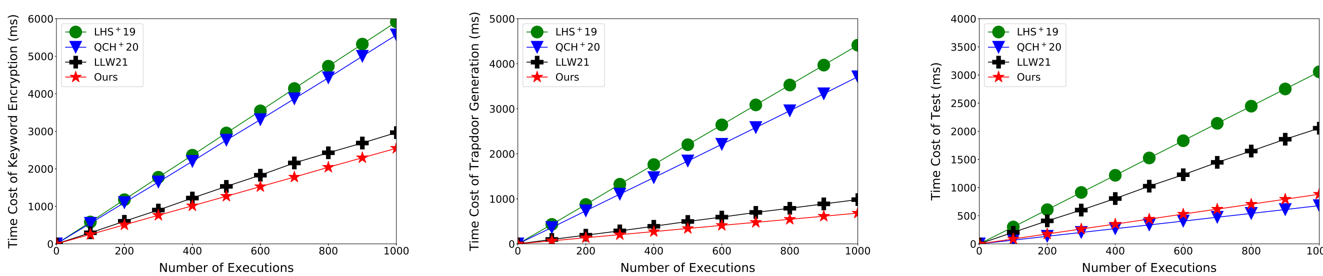
TABLE II
FEATURES OF COMPARED SCHEMES

Schemes	Type	Certificateless	Implicit Authentication	No Key Escrow	No Key Distribution	No Secure Channel[1]
HMZ+18 [35]	CLAEKS	Yes	Yes	Yes	No	No
LHS+19 [30]	IBAEKS	Yes	Yes	No	No	No
CWZ+19 [24]	PAEKS	No	No	Yes	No	Yes
LLZ19 [32]	CBAEKS	No	Yes	Yes	Yes	Yes
PSE20 [33]	CLAEKS	Yes	Yes	Yes	No	No
QCH+20 [25]	PAEKS	No	No	Yes	No	Yes
LLW21 [31]	CBAEKS	No	Yes	Yes	Yes	Yes
Ours	ICA-IBSE	No[2]	Yes	Yes	No	No

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

TABLE III
COMPUTATIONAL COST OF COMPARED SCHEMES

Schemes	KeywordEnc	TrapdoorGen	Test
HMZ+18 [35]	$5T_{em} + T_{htp} + 2T_h + 3T_{pm}$	$T_{bp} + 3T_{em} + T_{htp} + 2T_h + 2T_{pm}$	$2T_{bp} + 2T_{em} + T_h + 2T_{pm}$
LHS+19 [30]	$2T_{bp} + 3T_{em} + 2T_{htp}$	$T_{bp} + 2T_{em} + 2T_{htp}$	$2T_{bp} + 2T_{em} + T_{pm}$
CWZ+19 [24]	$5T_{sm} + 2T_{pa} + T_h$	$5T_{sm} + 2T_{pa} + T_h$	$4T_{sm} + 5T_{pa}$
LLZ19 [32]	$2T_{bp} + 3T_{em} + T_{htp} + 4T_h + T_{pm}$	$T_{bp} + 3T_{em} + 2T_{htp} + 3T_h + T_{pm}$	$T_{bp} + T_h$
PSE20 [33]	$T_{bp} + 3T_{em} + T_{htp}$	$T_{bp} + T_{em} + T_{htp} + T_h$	$T_{em} + T_h$
QCH+20 [25]	$T_{bp} + 3T_{em} + T_{htp} + T_h$	$2T_{em} + T_{htp}$	$T_{bp} + T_h$
LLW21 [31]	$5T_{sm} + 4T_h$	$2T_{sm} + 2T_{pa} + 2T_h$	$2T_{sm} + 2T_h$
Ours	$T_{bp} + 2T_{em} + T_{htp} + 2T_h$	$T_{bp} + T_{htp} + T_h$	$T_{em} + T_h$



(a) Encrypting keywords
(b) Generating trapdoors
(c) Performing tests

Fig. 2. Comparison of the efficiency of our scheme with LHS+19, QCH+20, and LLW21 schemes.

TABLE IV
COMMUNICATION COST OF COMPARED SCHEMES

Schemes	Public Key	Secret Key	Ciphertext	Trapdoor
HMZ+18 [35]	$2\|\mathbb{G}_1\| + \|ID\|$	$2\|\mathbb{Z}_q^*\|$	$2\|\mathbb{G}_1\|$	$\|\mathbb{G}_T\|$
LHS+19 [30]	$\|ID\|$	$\|\mathbb{G}_1\|$	$2\|\mathbb{G}_1\| + \|\mathbb{G}_T\|$	$2\|\mathbb{G}_1\|$
CWZ+19 [24]	$\|\mathbb{G}_{ec}\|$	$\|\mathbb{Z}_q^*\|$	$3\|\mathbb{G}_{ec}\|$	$2\|\mathbb{G}_{ec}\|$
LLZ19 [32]	$2\|\mathbb{G}_1\|$	$\|\mathbb{Z}_q^*\|$	$3\|\mathbb{G}_1\| + \|h\|$	$\|\mathbb{G}_1\|$
PSE20 [33]	$\|\mathbb{G}_1\| + \|ID\|$	$\|\mathbb{G}_1\| + \|\mathbb{Z}_q^*\|$	$2\|\mathbb{G}_1\|$	$\|h\|$
QCH+20 [25]	$\|\mathbb{G}_1\|$	$\|\mathbb{Z}_q^*\|$	$\|\mathbb{G}_1\| + \|h\|$	$\|\mathbb{G}_1\|$
LLW21 [31]	$3\|\mathbb{G}_{ec}\|$	$2\|\mathbb{Z}_q^*\|$	$\|\mathbb{G}_{ec}\| + 2\|\mathbb{Z}_q^*\| + \|h\|$	$\|\mathbb{G}_{ec}\| + \|\mathbb{Z}_q^*\|$
Ours	$\|ID\|$	$\|\mathbb{G}_1\|$	$2\|\mathbb{G}_1\|$	$\|h\|$

TABLE V
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

- 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 $\mu_{ID_i} \xleftarrow{\$} \mathbb{Z}_q^*$, 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 $\mathcal{O}_{\mathsf{DBDH}}(pk_{KGC}, u_{ID_i}, u_{ID_j}, k) = 1$.

TABLE VI
BIT LENGTH OF ELEMENTS AND RUNNING TIME OF OPERATIONS (80-BIT SECURITY)

Bit-length (bit)					**Running time (ms)**							
$\lvert\mathbb{Z}_q^*\rvert$	$\lvert\mathbb{G}_1\rvert$	$\lvert\mathbb{G}_T\rvert$	$\lvert\mathbb{G}_{ec}\rvert$	$\lvert h\rvert$	T_{htp}	T_h	T_{bp}	T_{em}	T_{pm}	T_{mi}	T_{pa}	T_{sm}
160	512	1024	320	256	1.929	0.009	0.654	0.854	0.002	0.001	0.002	0.281

- 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 $h \xleftarrow{\$} \mathbb{Z}_q^*$, 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 $\tilde{h} \xleftarrow{\$} \mathbb{Z}_q^*$, adds $((h,c_1),\tilde{h})$ to h_2-list, and returns $\tilde{h}$ to $\mathcal{A}$.
- $\mathcal{O}_{\mathsf{ik\text{-}KGC}}(\cdot)$: When $\mathcal{A}$ issues a issue key query, $\mathcal{B}$ first samples $ID \xleftarrow{\$} \mathcal{ID}$ 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 $y_{ID,1} \xleftarrow{\$} \mathbb{Z}_q^*$, computes $u_{ID,1}=g^{y_{ID,1}}$, and computes $u_{ID,2}\in\mathbb{G}$ as $u_{ID,2}=u_{ID}\cdot u_{ID,1}$ and $\sigma_{\mathsf{Sig}} \leftarrow \mathsf{Sig.Sign}(sk_{ICA}, u_{ID,2})$. Finally, $\mathcal{B}$ returns $\mathsf{M}_1=(u_{ID,2},\sigma_{\mathsf{Sig}})$ to $\mathcal{A}$, sets $IDList[Q_{key}]=ID$, and updates $Q_{key}=Q_{key}+1$.
- $\mathcal{O}_{\mathsf{ct\text{-}KGC}}(\cdot)$: when $\mathcal{A}$ queries for a ciphertext with (i,j,w), where $i,j\in[Q_{key}]$, $\mathcal{B}$ executes the following steps:
  - retrieve $(IDList[i],\mu_{IDList[i]},u_{IDList[i]})$ and $(IDList[j],\mu_{IDList[j]},u_{IDList[j]})$ from H-list for $IDList[i]$ and $IDList[i]$, respectively.
  - randomly choose $r \xleftarrow{\$} \mathbb{Z}_q^*$.
  - if $\{i,j\}=\{\ell_1,\ell_2\}$: search h_1-list for the tuple $((IDList[i], IDList[j],\perp,w),h)$. If no such tuple exists, randomly choose $h \xleftarrow{\$} \mathbb{Z}_q^*$ and add $((IDList[i], IDList[j],\perp,w),h)$ to h_1-list.
  - otherwise (*i.e.,* $i\notin\{\ell_1,\ell_2\}$ or $j\notin\{\ell_1,\ell_2\}$), either compute $k=\hat{e}(pk_{KGC}^{\mu_{IDList[i]}},u_{IDList[j]})$ if $i\notin\{\ell_1,\ell_2\}$ or compute $k=\hat{e}(u_{IDList[i]},pk_{KGC}^{\mu_{IDList[j]}})$ 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 $k=\hat{e}(pk_{KGC}^{\mu_{IDList[i]}},u_{IDList[j]})$ or $k=\hat{e}(u_{IDList[i]},pk_{KGC}^{\mu_{IDList[j]}})$.
  - search h_1-list for the tuple $((IDList[i], IDList[j],k,w),h)$. If no such tuple exists, randomly choose $h \xleftarrow{\$} \mathbb{Z}_q^*$ and add $((IDList[i], IDList[j],k,w),h)$ 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 $ct_w=(c_1,c_2)$.
- $\mathcal{O}_{\mathsf{td\text{-}KGC}}(\cdot)$: when $\mathcal{A}$ queries for a trapdoor with (i,j,w), where $i,j\in[Q_{key}]$, $\mathcal{B}$ executes the following steps:
  - retrieve $(IDList[i],\mu_{IDList[i]},u_{IDList[i]})$ and $(IDList[j],\mu_{IDList[j]},u_{IDList[j]})$ from H-list for $IDList[i]$ and $IDList[i]$, respectively.
  - if $\{i,j\}=\{\ell_1,\ell_2\}$: search h_1-list for the tuple $((IDList[i], IDList[j],\perp,w),h)$. If no such tuple exists, randomly choose $h \xleftarrow{\$} \mathbb{Z}_q^*$ and add $((IDList[i], IDList[j],\perp,w),h)$ to h_1-list.
  - otherwise (*i.e.,* $i\notin\{\ell_1,\ell_2\}$ or $j\notin\{\ell_1,\ell_2\}$), either compute $k=\hat{e}(pk_{KGC}^{\mu_{IDList[i]}},u_{IDList[j]})$ if $i\notin\{\ell_1,\ell_2\}$ or compute $k=\hat{e}(u_{IDList[i]},pk_{KGC}^{\mu_{IDList[j]}})$ 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 $k=\hat{e}(pk_{KGC}^{\mu_{IDList[i]}},u_{IDList[j]})$ or $k=\hat{e}(u_{IDList[i]},pk_{KGC}^{\mu_{IDList[j]}})$.
  - search h_1-list for the tuple $((IDList[i], IDList[j],k,w),h)$. If no such tuple exists, randomly choose $h \xleftarrow{\$} \mathbb{Z}_q^*$ and add $((IDList[i], IDList[j],k,w),h)$ 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 $(IDList[\alpha],\perp,u_{IDList[\alpha]})$ and $(IDList[\beta],\perp,u_{IDList[\beta]})$ by calling $H(IDList[\alpha])$ and $H(IDList[\beta])$, respectively. If $\{\alpha,\beta\}\neq\{\ell_1,\ell_2\}$, abort the game.
- for $i=1,\cdots,n$, perform the following steps. First, randomly choose $r_i \xleftarrow{\$} \mathbb{Z}_q^*$. Second, search h_1-list for the tuple $((IDList[\alpha], IDList[\beta],\perp,w_{b,i}),h_i)$. If no such tuple is found, randomly choose $h_i \xleftarrow{\$} \mathbb{Z}_q^*$ and add $((IDList[\alpha], IDList[\beta],\perp,w_{b,i}),h_i)$ 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 with a nonnegligible advantage ϵ, then there exists some $\mathcal{B}$ that can break the GBDH problem with 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 $(\mathit{IDList}[\alpha], \perp, u_{\mathit{IDList}[\alpha]})$ and $(\mathit{IDList}[\beta], \perp, u_{\mathit{IDList}[\beta]})$ by calling $H(\mathit{IDList}[\alpha])$ and $H(\mathit{IDList}[\beta])$, respectively. If $\{\alpha, \beta\} \neq \{\ell_1, \ell_2\}$, $\mathcal{B}$ aborts the game.
- search h_1-list for the tuple $((\mathit{IDList}[\alpha], \mathit{IDList}[\beta], \perp, w_b), h)$. If no such tuple is found, randomly choose $h \xleftarrow{\$} \mathbb{Z}_q^*$ and add $((\mathit{IDList}[\alpha], \mathit{IDList}[\beta], \perp, w_b), h)$ to h_1-list.
- return the challenged trapdoor $td^* = h$.

□

VII. 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: CWZ+19 [24] and QCH+20 [25], the IBAEKS scheme: LHS+19 [30], the CBAEKS schemes: LLZ19 [32] and LLW21 [31], and the CLAEKS schemes: HMZKL19 [35] and PSE20 [33]. The features of these schemes are listed in TABLE II. We also evaluate the performance of our proposed scheme against that of the LHS+19 [30], QCH+20 [25], and LLW21 [31] schemes.

A. Theoretical Comparison

We compare the schemes with respect to their communication cost and computational cost. The comparison results are presented in TABLE III and IV. For communication cost, we use $|\mathbb{Z}_q^*|$, $|\mathbb{G}_1|$, $|\mathbb{G}_T|$, and $|\mathbb{G}_{ec}|$ to denote the bit lengths of element of the $\mathbb{Z}_q^*$, bilinear group $\mathbb{G}_1$, bilinear target group $\mathbb{G}_T$, and elliptic curve group $\mathbb{G}_{ec}$, 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 schemes [31], [32], 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 eight time-consuming operations that are primarily used in these schemes, namely hash-to-point function (T_{htp}), hash (T_h), bilinear pairing (T_{bp}), modular exponential over bilinear group $\mathbb{G}_1$ (T_{em}), point multiplication over the bilinear group $\mathbb{G}_1$ (T_{pm}), modular inverse over $\mathbb{Z}_q^*$ (T_{mi}), point addition over the elliptic curve group $\mathbb{G}_{ec}$ (T_{pa}), and scalar multiplication over the elliptic curve group $\mathbb{G}_{ec}$ (T_{sm}).

B. Performance Evaluation

In order to make a more specific comparison, we first experiment to compare the time cost of each operation and the space required by each element, where the time cost of each operation is obtained by the average of 1000 times. Furthermore, to evaluate the performance of our scheme, we fully implement our proposed scheme, LHS+19 [30], QCH+20 [25], and LLW21 [31] schemes. The source codes of comparison and implementations are available at `https://github.com/zyliu-crypto/ICA-IBSE`. We conduct the experiment in the environment described in TABLE V. Specifically, we use the SHA3-256 library[3] for the general cryptographic hash function, and we use the PBC library[4] and MIRACL library[5] for operations over bilinear groups (T_{htp}, T_{bp}, T_{em}, and T_{pm}) and elliptic curve group (T_{mi}, T_{pa}, and T_{sm}), respectively. For achieving the same security level (*i.e.,* 80-bit), 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 $\mathbb{G}_T$ for bilinear pairing, and adopt the security parameter secp160r1 recommended by Standards for Efficient Cryptography Group [42] for elliptic curve group $\mathbb{G}_{ec}$.

The result of the cost of each operation and space required by each element is listed in Table. VI. In addition, the evaluation results of the concrete implementations are presented in Fig. 2 which shows that while our scheme is slower than QCH+20 scheme [25] in performing test, overall comparisons, our scheme can effectively encrypt keywords, generate trapdoors and perform tests. In addition, although LLW21 scheme [31] is a pairing-free scheme and is faster than our scheme in theoretical comparison, it needs to use a lot of hash functions, resulting in the need to constantly switch `epoint` type and `big` type in MIRACL library and `char` type in C language back and forth in practice, thus increasing execution time.

VIII. Conclusion

In this paper, we present a novel ICA-IBSE scheme that masters the trade-off between efficiency (in terms of low storage requirement), convenience, and security. A concrete framework is presented, and security proofs are provided, which demonstrate that the ICA-IBSE scheme can resist MCKA and IKGA 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.

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.

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

[2] D. X. Song, D. A. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *IEEE S&P 00*. IEEE Computer Society, 2000, pp. 44–55.

[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.

[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.

[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.

[6] C. Guo, W. Liu, X. Liu, and Y. Zhang, "Secure similarity search over encrypted non-uniform datasets," *IEEE Trans. Cloud Comput.*, 2020.

[7] P. Tian, C. Guo, K.-K. R. Choo, Y. Liu, L. Li, and L. Yao, "EAFS: An efficient, accurate, and forward secure searchable encryption scheme supporting range search," *IEEE Syst. J.*, pp. 1–11, 2021.

[8] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *EUROCRYPT 2004*, C. Cachin and J. Camenisch, Eds., vol. 3027. Springer, 2004, pp. 506–522.

[9] 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 *CRYPTO 2005*, V. Shoup, Ed., vol. 3621. Springer, 2005, pp. 205–222.

[10] L. Xu, X. Yuan, R. Steinfeld, C. Wang, and C. Xu, "Multi-writer searchable encryption: An LWE-based realization and implementation," in *AsiaCCS 2019*, S. D. Galbraith, G. Russello, W. Susilo, D. Gollmann, E. Kirda, and Z. Liang, Eds. ACM, 2019, pp. 122–133.

[11] 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.

[12] 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.

[13] 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.

[14] M. Ma, D. He, N. Kumar, K. R. Choo, and J. Chen, "Certificateless searchable public key encryption scheme for industrial Internet of things," *IEEE Trans. Ind. Informatics*, vol. 14, no. 2, pp. 759–767, 2018.

[15] K. Zhang, J. Long, X. Wang, H.-N. Dai, K. Liang, and M. Imran, "Lightweight searchable encryption protocol for industrial Internet of things," *IEEE Trans. Ind. Informatics*, vol. 17, no. 6, pp. 4248–4259, 2021.

[16] X. Liu, G. Yang, W. Susilo, J. Tonien, X. Liu, and J. Shen, "Privacy-preserving multi-keyword searchable encryption for distributed systems," *IEEE Trans. Parallel Distributed Syst.*, vol. 32, no. 3, pp. 561–574, 2021.

[17] 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 *SDM 2006*, W. Jonker and M. Petkovic, Eds., vol. 4165. Springer, 2006, pp. 75–83.

[18] R. Chen, Y. Mu, G. Yang, F. Guo, and X. Wang, "A new general framework for secure public key encryption with keyword search," in *ACISP 2015*, E. Foo and D. Stebila, Eds., vol. 9144. Springer, 2015, pp. 59–76.

[19] ——, "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.

[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 Trans. Inf. Forensics Secur.*, vol. 11, no. 12, pp. 2833–2842, 2016.

[21] 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.

[22] 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.

[23] 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.

[24] B. Chen, L. Wu, S. Zeadally, and D. He, "Dual-server public-key authenticated encryption with keyword search," *IEEE Trans. Cloud Comput.*, 2019.

[25] 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.

[26] Z.-Y. Liu, Y.-F. Tseng, R. Tso, M. Mambo, and Y.-C. Chen, "Public-key authenticated encryption with keyword search: A generic construction and its quantum-resistant instantiation," Cryptology ePrint Archive, Report 2020/955, 2020, https://ia.cr/2020/955.

[27] L. Wu, B. Chen, S. Zeadally, and D. He, "An efficient and secure searchable public key encryption scheme with privacy protection for cloud storage," *Soft Comput.*, vol. 22, no. 23, pp. 7685–7696, 2018.

[28] X. Pan and F. Li, "Public-key authenticated encryption with keyword search achieving both multi-ciphertext and multi-trapdoor indistinguishability," *J. Syst. Archit.*, vol. 115, p. 102075, 2021.

[29] Y. Lu and J. Li, "Lightweight public key authenticated encryption with keyword search against adaptively-chosen-targets adversaries for mobile devices," *IEEE Trans. Mob. Comput.*, pp. 1–1, 2021.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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 *ProvSec 2019*, R. Steinfeld and T. H. Yuen, Eds., vol. 11821. Springer, 2019, pp. 113–129.

[35] 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.

[36] Y. Lu, J. Li, and Y. Zhang, "Privacy-preserving and pairing-free multirecipient certificateless encryption with keyword search for cloud-assisted IIoT," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 2553–2562, 2020.

[37] S. S. M. Chow, "Removing escrow from identity-based encryption," in *PKC 2009*, S. Jarecki and G. Tsudik, Eds., vol. 5443. Springer, 2009, pp. 256–276.

[38] K. Emura, S. Katsumata, and Y. Watanabe, "Identity-based encryption with security against the KGC: A formal model and its instantiation from lattices," in *ESORICS 2019*, K. Sako, S. A. Schneider, and P. Y. A. Ryan, Eds., vol. 11736. Springer, 2019, pp. 113–133.

[39] T. Okamoto and D. Pointcheval, "The gap-problems: A new class of problems for the security of cryptographic schemes," in *PKC 2001*, K. Kim, Ed., vol. 1992. Springer, 2001, pp. 104–118.

[40] 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.

[41] E. Rescorla, "The transport layer security (TLS) protocol version 1.3," *RFC*, vol. 8446, pp. 1–160, 2018.

[42] M. Qu, "SEC 2: Recommended elliptic curve domain parameters," *Certicom Res., Mississauga, ON, Canada, Tech. Rep. SEC2-Ver-0.6*, 1999. [Online]. Available: https://www.secg.org/sec2-v2.pdf