Jiang P, Mu Y, Guo F *et al.* Private keyword-search for database systems against insider attacks. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 32(3): 599–617 May 2017. DOI 10.1007/s11390-017-1745-8

Private Keyword-Search for Database Systems Against Insider Attacks

Peng Jiang[1,2], Yi Mu[2], *Senior Member, IEEE*, Fuchun Guo[2], and Qiao-Yan Wen[1]

[1] *State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications Beijing 100876, China*

[2] *Centre for Computer and Information Security Research, School of Computing and Information Technology University of Wollongong, Wollongong, NSW 2522, Australia*

E-mail: {pj688, ymu, fuchun}@uow.edu.au; wqy@bupt.edu.cn

Received May 16, 2016; revised January 19, 2017.

Abstract The notion of searchable encrypted keywords introduced an elegant approach to retrieve encrypted data without the need of decryption. Since the introduction of this notion, there are two main searchable encrypted keywords techniques, symmetric searchable encryption (SSE) and public key encryption with keyword search (PEKS). Due to the complicated key management problem in SSE, a number of concrete PEKS constructions have been proposed to overcome it. However, the security of these PEKS schemes was only weakly defined in presence of outsider attacks; therefore they suffer from keyword guessing attacks from the database server as an insider. How to resist insider attacks remains a challenging problem. We propose the first searchable encrypted keywords against insider attacks (SEK-IA) framework to address this problem. The security model of SEK-IA under public key environment is rebuilt. We give a concrete SEK-IA construction featured with a constant-size trapdoor and the proposed scheme is formally proved to be secure against insider attacks. The performance evaluations show that the communication cost between the receiver and the server in our SEK-IA scheme remains constant, independent of the sender identity set size, and the receiver needs the minimized computational cost to generate a trapdoor to search the data from multiple senders.

Keywords public key encryption with keyword search, keyword privacy, insider attack, searchable encrypted keyword

1 Introduction

Large databases, such as cloud storage platforms, are widely used in outsourcing services of data storage. The widespread adoption of cloud-based database systems (e.g., Dropbox, PHR) has been driven by the tremendous economic benefit. As cloud servers are generally regarded as honest-but-curious, in order to protect outsourced data, encryption is considered as a fundamental approach of protecting data security and preventing data disclosure. However, searching encrypted data is not always easy. One of the trivial solutions is that users download and decrypt all the encrypted data to search for the target locally, which requires additional communication and computation overheads. Another solution is that the server executes decryption operations with authorized keys from users and returns wanted data to users, which compromises data privacy.

Keyword-based searchable encryption enters people's field of vision and gains growing interest as a promising tool for data retrieval in large database systems. There are two main keyword search techniques, symmetric searchable encryption (SSE) and public key encryption with keyword search (PEKS).

- SSE was presented by Song *et al.*[1] to take keywords to search encrypted data. SSE uses symmetric key cryptography and allows only the secret key holder to create searchable ciphertexts and trapdoors. The server returns the matching encrypted data.
- PEKS was introduced by Boneh *et al.*[2] to achieve encrypted data sharing. In PEKS, a sender (the party

Regular Paper

This work is supported by the National Natural Science Foundation of China under Grant Nos. 61300181 and 61502044, and the Fundamental Research Funds for the Central Universities of China under Grant No. 2015RC23.

who encrypts the data) generates the searchable ciphertext using the keyword and the public key of a receiver, and the receiver (the party who can search over the encrypted data) produces the trapdoor using the keyword and his/her secret key. Then the server returns corresponding encrypted data to the receiver when the trapdoor matches the ciphertext. This means that two keywords are identical.

Although SSE allows the searching over encrypted data, it does not provide encrypted data sharing to/from other entities and is subjected to complicated key distribution/management problems. The introduction of PEKS overcomes the weaknesses of SSE. PEKS achieves encrypted data sharing and provides an efficient way for the receiver to retrieve the target data based on the encrypted keywords without complicated key management.

In comparison with SSE, unfortunately, PEKS suffers from keyword guessing attacks where the keyword privacy in trapdoors can be compromised. Some existing PEKS schemes, such as [3-6], are secure against outsider attacks, which means any outside attacker, except the server, cannot distinguish the keyword from the trapdoor. However, the untrusted server may still intend to obtain the keyword from the given trapdoor. Keyword guessing attacks from the server are called as insider attacks[6].

Insider Attacks in PEKS. The problem of all previously published PEKS schemes is that the untrusted server can launch a keyword guessing attack to find the keyword from the given trapdoor by the brute-force method. We assume that there is a keyword set $\mathcal{W}$ in the system, where the size of $\mathcal{W}$ is N. Usually, the public key of any party is fully public and available to the whole system. Due to the public generation of searchable ciphertext, the untrusted server can guess the keyword with the following method.

1) Given the trapdoor T_w associated with some unknown keyword $w \in \mathcal{W}$, the untrusted server chooses a keyword $w' \in \mathcal{W}$ and the public key of the receiver to generate a searchable ciphertext $PEKS_{w'}$.

2) If $PEKS_{w'}$ matches T_w, then $w = w'$. The server infers the correct keyword w used in T_w.

3) Otherwise, the server continues to generate another searchable ciphertext $PEKS_{w''}$ until finding the correct keyword w from trapdoor T_w.

In a database system, the server, who is not trusted, provides data storage and data search. As the service provider, the server can collect some sensitive information to obtain the extra and illegal profit. When the server executes searching operations, it attempts to know what the receiver wants to search by insider attacks. In this way, the server collects associated keywords the receiver focuses on and sells them to those who are interested in this receiver. Insider attacks will harm keyword privacy and even data privacy. Therefore, insider attacks are important and need to be resisted. So far, outsider attacks can be solved but solutions cannot be applied in insider attacks. The keyword privacy has not yet been achieved in the current PEKS schemes due to insider attacks. How to enhance the security of PEKS against insider attacks remains an unresolved research problem.

Contributions. We present the searchable encrypted keywords against insider attacks (SEK-IA). Our contributions are two-fold.

First, we propose a new SEK-IA framework to eliminate the insider attacks. The new framework borrows the property of identity-based encryption to address the keyword guessing attack from the server. In particular, the searchable ciphertext is created with the private key of the sender, the keyword, and the public key of the receiver. The trapdoor is created with the secret key of the receiver, the keyword and a sender identity set. The trapdoor matches the searchable ciphertext if and only if the two keywords are identical and the searchable ciphertext is created with the private key of a sender, whose identity belongs to this identity set. Insider attacks do not work under this new framework because the server cannot create a valid searchable ciphertext to conduct the keyword guessing attack.

Second, we construct a provably secure SEK-IA scheme featured with a constant-size trapdoor. The trapdoor size is independent of the sender identity set size, which is composed of six group elements only. We prove the semantic security of the proposed scheme, where both the searchable ciphertext and the trapdoor are keyword-indistinguishable against adversaries including the untrusted server.

Organization. The remainder of the paper is organized as follows. In Section 2, we review related work. In Section 3, we describe the new SEK-IA framework for the database system with relevant definitions. We propose a concrete SEK-IA scheme with constant size trapdoor in Section 4. The semantic security of the proposal is formally proved in Section 5. We evaluate and discuss our scheme in Section 6. Finally, we conclude the paper in Section 7.

2 Related Work

Song *et al.*[1] initiated the research on searchable encryption and presented the first SSE solution, and several schemes based on SSE[7-9] have been proposed to improve the system performance. Due to the symmentric setting, SSE fails to provide encrypted data sharing to/from other entities and is subjected to complicated key distribution/management issues. Tang[10] used a public key cryptosystem to achieve a multiparty searchable encryption (MPSE) scheme. MPSE can execute the search over encrypted data with delegation function in the multi-user environment. However, MPSE is still in the symmetric setting, i.e., both the searchable ciphertext (index) and the trapdoor (search request) are generated by the user or his/her authorized entity. MPSE also suffers from the common weakness of SSE of the complicated key management and data sharing among independent entities.

Boneh *et al.*[2] introduced the notion of PEKS to overcome the weaknesses of SSE, where PEKS allows the search over encrypted data and avoids heavy key distribution and management. Afterwards, many PEKS variants have been proposed to improve PEKS. Public-key encryption with conjunctive keyword search (PECKS) schemes[11-14] were proposed to develop the scalability and the query expression. PEKS with temporary keyword[15] and PEKS with fuzzy keyword[16] were presented to enhance the database system usability. Authorized PEKS schemes[17-20] were proposed to support both data search and access control.

PEKS schemes against outsider attacks were studied in [3-6, 21-22]. Boneh *et al.*'s PEKS scheme[2] was proved to be semantically secure under chosen keyword attacks, where a secure channel was required between the receiver and the server. Byun *et al.*[21] and Yau *et al.*[3] raised the outsider keyword guessing attack on Boneh *et al.*'s PEKS scheme[2] under the case of no secure channel. The attacker could guess keywords in a given trapdoor. Also, the cost to build a secure channel is expensive between the server and the receiver, which is impractical for data storage systems. Baek *et al.*[4] also addressed the secure channel problem and indicated that an outside attacker could link keyword information from the trapdoor by generating a searchable ciphertext. They removed the need for a secure channel and applied a server's public/secret key pair to build secure-channel-free PEKS (SCF-PEKS) against the outsider attack. In the SCF-PEKS scheme, the sender uses both the server's public key and the receiver's public key to generate the searchable ciphertext, and only the designated server chosen by the receiver can perform searching operation to check the relationship between a PEKS ciphertext and a trapdoor. Rhee *et al.*[5] enhanced the security model and gave the first secure SCF-PEKS construction. They also achieved trapdoor unlinkability by introducing a random variable in the trapdoor. Fang *et al.*[6] advanced the SCF-PEKS scheme without random oracle.

Most of the existing schemes focus on resisting the outsider attacks, while the problem to resist insider attacks is still unsolved. The previous schemes share a common feature that the ciphertext can be publicly generated and the server is able to generate a trapdoor-match searchable ciphertext to distinguish the keyword from the trapdoor. Insider attacks were seen to be inherently unavoidable in PEKS systems[6,23], which are based on previous searchable ciphertext creation. Insider attacks frustrate out the keyword privacy and are still an open problem in searchable encryption schemes based on the public key system.

3 New SEK-IA System Framework

3.1 PEKS System Model

The general PEKS system is described in Fig.1, including three independent entities, i.e., sender, receiver and cloud server.

• Sender owns data m associated with keyword w. He/she generates the searchable ciphertext $PEKS_w$ with the algorithm *Encrypt* and the data ciphertext CT_m with some encryption algorithm *Enc*. Then the sender uploads $(PEKS_w, CT_m)$ to the cloud server. We note that the algorithm *Encrypt* inputs the public key of the receiver PK, the keyword w and certain public information (denoted as $\cdot$, which is not fully necessary).

• Receiver wants to retrieve the encrypted data associated with keyword w'. He/she generates a trapdoor $T_{w'}$ with the algorithm $Trapdoor$. The receiver sends $T_{w'}$ to the server as a request to search the corresponding encrypted data. The algorithm $Trapdoor$ inputs the secret key of the receiver SK, the keyword w and certain public information (also denoted it as $\cdot$).

• Cloud server provides data storage and data indexing service. The server stores multiple pieces of ciphertexts. We denote them as $(PEKS_{w_1},\ CT_{m_1})$, $(PEKS_{w_2}, CT_{m_2}), \cdots,\ (PEKS_{w_n},\ CT_{m_n})$. When the received trapdoor $T_{w'}$ matches $PEKS_w$, the server returns the corresponding ciphertext CT_m to the receiver. The server is not fully trusted, which means that it

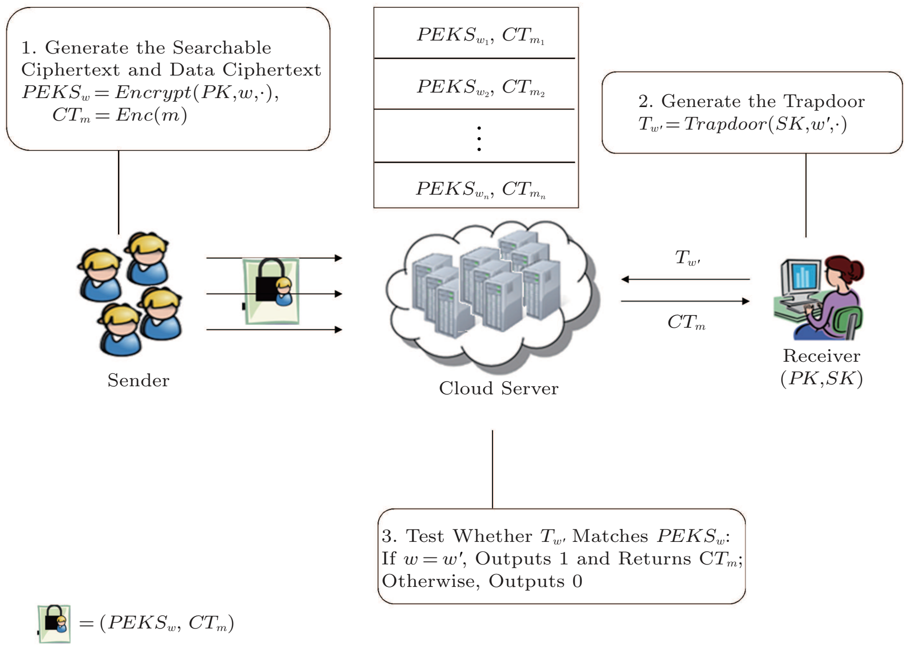
1. Generate the Searchable Ciphertext and Data Ciphertext $PEKS_w = Encrypt(PK,w,\cdot)$, $CT_m = Enc(m)$
$PEKS_{w_1}$, CT_{m_1}
$PEKS_{w_2}$, CT_{m_2}
$PEKS_{w_n}$, CT_{m_n}
2. Generate the Trapdoor $T_{w'} = Trapdoor(SK,w',\cdot)$
$T_{w'}$
CT_m
Sender
Cloud Server
Receiver (PK,SK)
3. Test Whether $T_{w'}$ Matches $PEKS_w$: If $w = w'$, Outputs 1 and Returns CT_m; Otherwise, Outputs 0
$= (PEKS_w, CT_m)$

Fig.1. PEKS system model.

honestly executes the searching operation but curiously learns the content associated with the encrypted data.

3.2 Design Goals and New SEK-IA Framework

The desired design goals for a keyword search scheme on encrypted data in the database system are listed as follows.

• *Chosen Keyword Attack.* Both the server and any outsider attackers should be unable to distinguish the keyword from the ciphertext.

• *Keyword Guessing Attack.* Both the server and any outsider attackers should be unable to distinguish the keyword from the trapdoor.

• *Efficiency.* The communication complexity between the receiver and the server should be as low as possible to reduce the bandwidth between them.

SEK-IA Framework. We present the overall description of the SEK-IA framework for the database systems in Fig.2.

A major difference for the entities between SEK-IA and PEKS is that the trusted third party (TTP) is involved in initializing the system. TTP is a trusted entity who manages the whole system. Under the SEK-IA framework, each entity is independent and we assume that any two entities cannot collude since the collusion of any two entities would directly compromise keyword privacy and data security.

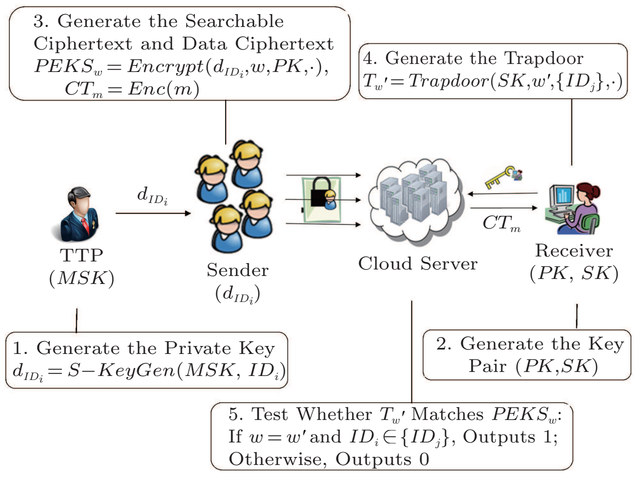
3. Generate the Searchable Ciphertext and Data Ciphertext $PEKS_w = Encrypt(d_{ID_i},w,PK,\cdot)$, $CT_m = Enc(m)$
4. Generate the Trapdoor $T_w' = Trapdoor(SK,w',\{ID_j\},\cdot)$
d_{ID_i}
CT_m
TTP (MSK)
Sender (d_{ID_i})
Cloud Server
Receiver (PK, SK)
1. Generate the Private Key $d_{ID_i} = S\text{-}KeyGen(MSK, ID_i)$
2. Generate the Key Pair (PK,SK)
5. Test Whether $T_{w'}$ Matches $PEKS_w$: If $w = w'$ and $ID_i \in \{ID_j\}$, Outputs 1; Otherwise, Outputs 0

Fig.2. New SEK-IA framework.

The basic process runs as follows. TTP generates the private key d_{ID_i} using the *S-KeyGen* algorithm with the input of ID_i and the master secret key MSK, and issues d_{ID_i} to the sender, and the receiver generates his/her public/secret key pair (PK, SK). The sender generates the searchable ciphertext $PEKS_w$ using the *Encrypt* algorithm with the input of his/her private key d_{ID_i}, the keyword w, the designated receiver's public key PK and other certain public information. Also, the sender generates the data ciphertext CT_m for the data m using some encryption algorithm *Enc*. Then the sender uploads $(PEKS_w, CT_m)$ to the cloud server. When the receiver wants to retrieve some data associated with the keyword w' from some senders $\{ID_j\}$, he/she generates a trapdoor $T_{w'}$ using the *Trapdoor* algorithm with the input of his/her secret key SK, the keyword w', the specified sender identity set $\{ID_j\}$ and certain public information. The server performs the searching operation by checking the identity verification $ID_i \in \{ID_j\}$ and the keyword matching $w = w'$. If both pass, it outputs 1 and the server returns the corresponding encrypted data to the receiver; otherwise, it outputs 0.

In the new SEK-IA framework, the server can neither distinguish the keyword from trapdoor nor do that from the ciphertext. The server cannot generate a correct searchable ciphertext with a keyword to match the trapdoor, and thus he/she cannot guess the keyword using the insider attack. Also, the server is unable to generate a valid trapdoor to match the stored searchable ciphertext.

3.3 Algorithm Definition

Before defining SEK-IA algorithms, we summarize the used notations of the whole paper in Table 1 to improve the readability.

Table 1. Notations

Notation	Description
ID	Identity of the sender
w	Keyword
$PEKS_w$	Searchable ciphertext for keyword w
T_w	Trapdoor (i.e., search request) for keyword w
$Enc(m)$	Data ciphertext for data m using the encryption algorithm Enc
PPT	Probabilistic polynomial time
TTP	Trust third party, who is fully trusted; TTP manages the whole system and issues the private key for the sender
PP	Public paramters
MSK	Master secret key, which is kept private by TTP
d_{ID}	Private key of the sender ID
PK, SK	Public/secret key pair of the receiver
N	Number of keywords in the system
n	Maximum number of senders in the trapdoor

The sender chooses the designated receivers and generates valid searchable ciphertexts, which are stored on the cloud. The receiver generates the trapdoor, which corresponds to the target dataset. With the trapdoor, the server searches the database and returns the corresponding dataset if and only if the encrypted keyword matches the keyword embedded in the trapdoor and the searchable ciphertext is created by the sender whose identity belongs to the receiver's specified identity set.

Definition 1. *An SEK-IA scheme consists of six algorithms as follows.*

Setup(1^λ). *Taking a security parameter 1^λ as input, it outputs the public parameter PP and the master secret key MSK.*

S-KeyGen(MSK, ID_i). *Taking the master secret key MSK and the sender identity ID_i as input, it outputs the private key of the sender d_{ID_i}, which is used for encryption.*

R-KeyGen(PP). *Taking the public parameters PP as input, it outputs the public/secret key pair of the receiver (PK, SK).*

Encrypt(d_{ID_i}, w, PK, PP). *Taking the public parameters PP, the sender's private key d_{ID_i}, the keyword w and the public key of the receiver PK as input, it outputs a searchable ciphertext $PEKS_w$.*

Trapdoor$(SK, w', \{ID_j\}, PP)$. *Taking the public parameters PP, the secret key of the receiver SK, the keyword w' and the specified sender identity set $\{ID_j\}$ as input, it outputs a trapdoor $T_{w'}$.*

Test$(PEKS_w, T_{w'}, ID_i, \{ID_j\}, PP)$. *Taking the public parameters PP, a searchable ciphertext $PEKS_w$ for ID_i and a trapdoor $T_{w'}$ for $\{ID_j\}$ as input, it outputs 1 if $ID_i \in \{ID_j\}$ and $w = w'$; otherwise, it outputs 0.*

3.4 Security Models

The full security model has been defined in [23], although it was thought to be difficult to work with its security proof. [22] also mentions that most papers leaked at least the search pattern or the access pattern. However, we find that the notion of the full security is only for the symmetric searchable encryption. SSE compromises key management and data sharing, thereby the full security model in SSE cannot be used in the asymmetric setting. We define the SEK-IA security models in the asymmetric setting by capturing chosen keyword attacks and keyword guessing attacks, based on [2, 5-6]. More formally, the following two games are played between a challenger and a PPT adversary.

Game 1. The semantic security against the chosen keyword attacks (SS-CKA) game allows the adversary $\mathcal{A}$ to launch chosen keyword attacks, where the adversary is anyone except the valid sender. $\mathcal{A}$ attempts to distinguish a searchable ciphertext for the keyword w_0 from a ciphertext for the keyword w_1. $\mathcal{A}$ plays the game with the challenger $\mathcal{C}$ as follows.

Init. $\mathcal{A}$ declares the challenge identity ID^*.

Setup. $\mathcal{C}$ runs the Setup algorithm and gives PP, PK to $\mathcal{A}$.

Phase 1. $\mathcal{A}$ performs a polynomially bounded number of queries.

• *Private Key Query.* $\mathcal{A}$ asks the private key query for ID_i, $ID_i \neq ID^*$. The challenger runs the S-KeyGen algorithm and responds d_{ID_i} to $\mathcal{A}$.

• *Trapdoor Query.* $\mathcal{A}$ issues $\{ID_i\}, w_i$ to $\mathcal{C}$. $\mathcal{C}$ responds the trapdoor T_{w_i} to $\mathcal{A}$ by running the Trapdoor algorithm.

• *Ciphertext Query.* $\mathcal{A}$ sends ID_i, w_i to $\mathcal{C}$. $\mathcal{C}$ responds the ciphertext $PEKS_{w_i}$ to $\mathcal{A}$ by running the Encrypt algorithm.

Challenge. $\mathcal{A}$ outputs two equal length keywords w_0, w_1 and ID^* on which it wants to be challenged. $\mathcal{A}$ did not previously query the private key for ID^*, or the trapdoor for w_0, w_1 for $\{ID_i\}$, where $ID^* \in \{ID_i\}$. $\mathcal{C}$ takes a random bit $\beta \in \{0,1\}$ and responds the challenge ciphertext $PEKS^*_{w_\beta}$ to $\mathcal{A}$.

Phase 2. $\mathcal{A}$ continues to query the private key for $ID_i \neq ID^*$, the trapdoor for $\{ID_i\}$ with $ID^* \notin \{ID_i\}$ or the trapdoor for $w_i \neq w_0, w_1$ for $\{ID_i\}$ with $ID^* \in \{ID_i\}$, and the ciphertext for any ID_i, w_i. $\mathcal{C}$ responds as phase 1.

Guess. $\mathcal{A}$ outputs its guess $\beta' \in \{0,1\}$ and wins the game if $\beta' = \beta$.

The advantage of $\mathcal{A}$ in this game is defined as $|\Pr[\beta' = \beta] - \frac{1}{2}|$, where the probability is taken over the random bit used by the challenger and the adversary.

Definition 2. *The SEK-IA scheme is semantically secure against chosen keyword attacks if there is no TTP adversary $\mathcal{A}$ who wins game* 1 *with a non-negligible advantage.*

Game 2. The indistinguishability against keyword guessing attacks (IND-KGA) game allows the adversary $\mathcal{A}$, including the insider server, to launch the keyword guessing attacks. $\mathcal{A}$ attempts to distinguish a trapdoor for w_0 from a trapdoor for w_1. The game between $\mathcal{A}$ and a challenger $\mathcal{C}$ is described as follows.

Init. $\mathcal{A}$ declares the challenge identity set $\{ID^*_i\}$.

Setup. $\mathcal{C}$ runs the Setup algorithm and gives PP, PK to $\mathcal{A}$.

Phase 1. $\mathcal{A}$ performs a polynomially bounded number of queries.

• *Private Key Query.* $\mathcal{A}$ issues ID_i to $\mathcal{C}$, $ID_i \notin \{ID^*_i\}$, and $\mathcal{C}$ responds d_{ID_i} to $\mathcal{A}$ by running the S-KeyGen algorithm.

• *Ciphertext Query.* $\mathcal{A}$ issues ID_i, w_i to $\mathcal{C}$. $\mathcal{C}$ responds the ciphertext $PEKS_{w_i}$ to $\mathcal{A}$ by running the Encrypt algorithm.

• *Trapdoor Query.* $\mathcal{A}$ issues any $w_i, \{ID_i\}$ to $\mathcal{C}$. $\mathcal{C}$ responds the trapdoor T_{w_i} to $\mathcal{A}$ by running the Trapdoor algorithm.

Challenge. $\mathcal{A}$ outputs two equal length keywords w_0, w_1 and $\{ID^*_i\}$, on which it wants to be challenged. $\mathcal{A}$ did not previously ask for the private key for ID_i, where $ID_i \in \{ID^*_i\}$, or the ciphertext for w_0, w_1 for ID_i, where $ID_i \in \{ID^*_i\}$. $\mathcal{C}$ takes a random bit $\beta \in \{0,1\}$ and responds the challenge trapdoor $T^*_{w_\beta}$ to $\mathcal{A}$.

Phase 2. $\mathcal{A}$ continues to query the private key for $ID_i \notin \{ID^*_i\}$, the ciphertext for $ID_i \notin \{ID^*_i\}$ or the ciphertext for ID_i, w_i, $ID_i \in \{ID^*_i\}, w_i \neq w_0, w_1$, and the trapdoor for any $w_i, \{ID_i\}$. $\mathcal{C}$ responds as phase 1.

Guess. $\mathcal{A}$ outputs its guess $\beta' \in \{0,1\}$ for β and wins the game if $\beta' = \beta$.

The advantage of adversary $\mathcal{A}$ in this game is defined as $|\Pr[\beta' = \beta] - \frac{1}{2}|$, where the probability is taken over the random bit used by the challenger and the adversary.

Definition 3. *The SEK-IA scheme is indistinguishable against keyword guessing attacks if there is no polynomial time adversary $\mathcal{A}$ who wins game* 2 *with a non-negligible advantage.*

4 Our Searchable Encrypted Keywords Scheme

In this section, we propose an SEK-IA scheme with a constant-size trapdoor. Our construction enables the receiver to search multiple ciphertexts from different senders with a constant-size trapdoor. Each receiver specifies both the keyword and the sender identity set.

4.1 Bilinear Pairing

We briefly review the bilinear pairings[24]. Let $\mathbb{G}_1, \mathbb{G}_2$ be two cyclic additive groups of prime order p, generated by P and Q respectively. Let also $\mathbb{G}_T$ be a cyclic multiplicative group with the same order p. We have $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a bilinear pairing with the following properties:

• bilinearity: $e(aP, bQ) = e(P,Q)^{ab}$ for all $P, Q \in \mathbb{G}_1 \times \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$;

• non-degeneracy: $e(P,Q) \neq 1$;

• computability: there is an efficient algorithm to compute $e(P,Q)$ for all $P, Q \in \mathbb{G}_1 \times \mathbb{G}_2$.

For simplicity of presentation, we assume a symmetric pairing, where $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$. We denote the bilinear pairing parameters with $(p, \mathbb{G}, \mathbb{G}_T, e(\cdot,\cdot))$, outputted by a PPT algorithm $\mathcal{BG}$ on input 1^λ.

4.2 Proposed Construction

Setup(1^λ). Taking as input the security parameter 1^λ, the bilinear pairings parameters $(p, \mathbb{G}, \mathbb{G}_T, e(\cdot,\cdot))$ are generated by running the $\mathcal{BG}$ algorithm as above. There are a set of keywords $\{w_1, \cdots, w_N\}$ and an identity set of senders $\{ID_1, \cdots, ID_n\}$ associated with the trapdoor from the receiver. We consider the case of single keyword and note as $w = w_i|_{i\in[1,N]}$ for simplicity. Let $P(w) \in \mathbb{G}$ correspond to keyword w. Choose generators $P_1, P_2, P_3, Q_1, Q_2, Q_3 \in \mathbb{G}$ satisfying $e(P_1, Q_1) = e(P_2, Q_2) = e(P_3, Q_3)$. The system chooses random numbers $\alpha, c \in \mathbb{Z}_p^*$, and computes $R_1 = \alpha P_1$, $R_2^i = \alpha^i P_2$, $S_1^j = c\alpha^j Q_1$, $S_2 = cQ_2$ and $S_3 = cQ_3$. The public parameters PP and the master secret key MSK are denoted as

$$\begin{aligned} PP &= \left(R_1, \{R_2^i\}, P_3, \{S_1^j\}, S_2, S_3, \{\alpha^l P(w)\}\right), \\ MSK &= (\alpha, c, P_1, P_2, Q_1, Q_2, Q_3), \end{aligned}$$

where $i \in [1, n-1]$, $j \in [0, n]$, and $l \in [1, 2]$.

S-KeyGen(MSK, ID_i). Taking as input the sender's identity ID_i and the master secret key MSK, the system generates the private key as $d_{ID_i} = \frac{1}{\alpha + ID_i} P_1$ and issues it to the sender through a secure channel.

R-KeyGen(PP). Taking as input the public parameters PP, the receiver chooses a random value $b \in \mathbb{Z}_p^*$ as his/her secret key, and computes $PK = bR_1$ as his/her public key. The receiver's public/secret key pair is denoted as $(PK, SK) = (bR_1, b)$.

Encrypt(d_{ID_i}, w, PK, PP). Taking as input the public parameters PP, the sender's private key d_{ID_i}, the keyword w and the receiver's public key PK, the sender ID_i picks a random number $r \in \mathbb{Z}_p^*$, and computes the searchable ciphertext as $PEKS_w = (C_1, C_2, \{C_{3,i}\}, C_4)$, where

$$\begin{aligned} C_1 &= \frac{r}{\alpha + ID_i} P_1, \\ C_2 &= rP_3, \\ C_{3,i} &= rR_2^i, i \in [1, n-1], \\ C_4 &= e(PK, P(w))^r. \end{aligned}$$

Then the sender uploads $PEKS_w$ to the server.

Trapdoor$(SK, w, \{ID_j\}, PP)$. Taking as input the public parameters PP, the receiver's secret key SK, the keyword w and the specified sender's identity set $\{ID_j\}$ with the size n, the receiver randomly picks $k_1, k_2 \in \mathbb{Z}_p^*$, and generates the trapdoor $T_w = (U_1, V_1, W_1, U_2, V_2, W_2)$, where

$$\begin{aligned} U_1 &= b\alpha^2 P(w) + k_1 c(\alpha + ID_1) + \cdots + (\alpha + ID_n)Q_1, \\ V_1 &= -k_1 S_2, \\ W_1 &= k_1 S_3, \\ U_2 &= b\alpha P(w) + k_2 c(\alpha + ID_1) + \cdots + (\alpha + ID_n)Q_1, \\ V_2 &= -k_2 S_2, \\ W_2 &= k_2 S_3, \end{aligned}$$

where $k_1 c(\alpha + ID_1) + \cdots + (\alpha + ID_n)Q_1, k_2 c(\alpha + ID_1) + \cdots + (\alpha + ID_n)Q_1$ are computable from S_1^i. Then the receiver delivers T_w to the server.

Test$(PEKS_w, T_w, ID_i, \{ID_j\}, PP)$. Taking as input the public parameters PP, a searchable ciphertext $PEKS_w$ for ID_i and a trapdoor T_w for specified $\{ID_j\}$, the server performs searching operation by checking the equation

$$\begin{aligned} & e\left(U_1 + ID_i U_2, C_1\right) e\left(V_1 + ID_i V_2, f_1(\alpha) r\alpha P_2\right) \\ =\ & C_4 \times e(C_2, W_1 + ID_i W_2)^{f_2} \end{aligned}$$

with $f_1(\alpha)$ being a polynomial of degree $n-2$ and f_2 being a constant in Z_p. If the above equation holds, the server outputs 1 and returns corresponding encrypted data to the receiver; otherwise, it outputs 0.

Since $f_1(\alpha)$ is of degree $n-2$, the item $f_1(\alpha) r\alpha P_2$ is computable from $C_{3,i}$. More precisely, we have

$$f_1(\alpha) = \frac{1}{\alpha}\left(\prod_{j\neq i}(\alpha + ID_j) - f_2\right), \ f_2 = \prod_{j\neq i} ID_j,$$

thereby

$$\begin{aligned} & e\left(U_1 + ID_i U_2, C_1\right) \\ =\ & C_4 \times e\left(P_1, Q_1\right)^{rc(k_1 + ID_i k_2)\prod\limits_{j\neq i}(\alpha + ID_j)}, \\ & e\left(V_1 + ID_i V_2, f_1(\alpha)\ r\alpha P_2\right) \\ =\ & e\left(P_2, Q_2\right)^{-rc(k_1 + ID_i k_2)(\prod\limits_{j\neq i}(\alpha + ID_j) - f_2)}. \end{aligned}$$

Combing $e(P_1, Q_1) = e(P_2, Q_2) = e(P_3, Q_3)$, we have

$$\begin{aligned} & e\left(U_1 + ID_i U_2, C_1\right) \times e\left(V_1 + ID_i V_2, f_1(\alpha) r\alpha P_2\right) \\ =\ & C_4 \times e(P_3, Q_3)^{rc(k_1 + ID_i k_2) f_2} \\ =\ & C_4 \times e(C_2, W_1 + ID_i W_2)^{f_2}. \end{aligned}$$

5 Security Proof

We formally prove the security of our SEK-IA scheme under the security model. To analyze the security, we define two new hard problems, namely (f, g, F)-MSE-DDH1 problem and (f, g, F)-MSE-DDH2 problem, which are special instances of the general MSE-DDH problems in [24]. The new hard problems with their intractability analysis can be found in Appendix.

Theorem 1. *The SEK-IA scheme is semantically secure against chosen keyword attacks in game 1 if (f, g, F)-MSE-DDH1 problem is intractable for any PPT algorithm.*

Proof. Suppose there exists a PPT adversary $\mathcal{A}$, in game 1, who can attack our scheme with advantage ϵ. We build a simulator $\mathcal{B}$, who has advantage ϵ/eq_T against (f, g, F)-MSE-DDH1 problem. $\mathcal{B}$'s running time is approximately the same with $\mathcal{A}$'s.

Init. Adversary $\mathcal{A}$ declares the challenge identity ID^*. We denote a general identity set as $\{ID_i\}_{i=1}^n$ and assume $ID^* \notin \{ID_i\}_{i=1}^n$.

Setup. Simulator $\mathcal{B}$ is given a group system $(p, \mathbb{G}, \mathbb{G}_T, e(\cdot,\cdot))$ as input, and the (f, g, F)-MSE-DDH1 instances. We also have two coprime polynomials f and q, of respective orders $n+1$ and 1, with their pairwise distinct roots, and $f(0) \neq 0, q(0) \neq 0$. $\mathcal{B}$ is further given $Z \in \mathbb{G}_T$, where Z is equal to $e(P_0, Q_0)^{rb\alpha^2 f(\alpha)q(\alpha)}$ or to some random element of $\mathbb{G}_T$. For simplicity, we state that f and q are unitary polynomials, but this is not a mandatory requirement,

$$f(x) = \prod_{i=1}^{n+1}(x + ID_i),\ q(x) = x + ID^*,$$

where ID_{n+1} is an additional dummy identity. For $i \in [1, n+1]$, we set $f_i(x) = \frac{f(x)}{x+ID_i}$.

To generate the system parameters, simulator $\mathcal{B}$ formally sets (note: the simulator only needs to compute P_3 and does not need to compute and publish other parameters)

$$\begin{aligned}
&P_1 = \alpha^2 f(\alpha)q(\alpha)P_0,\ Q_1 = \alpha Q_0,\\
&P_2 = \alpha q(\alpha)P_0,\ Q_2 = \alpha^2 f(\alpha)Q_0,\\
&P_3 = c\alpha^2 q(\alpha)P_0,\ Q_3 = \frac{\alpha}{c} f(\alpha)Q_0,
\end{aligned}$$

such that $e(P_1, Q_1) = e(P_2, Q_2) = e(P_3, Q_3)$, and calculates

$$\begin{aligned}
&R_1 = \alpha P_1 = \alpha^3 f(\alpha)q(\alpha)P_0,\\
&R_2^i = \alpha^i P_2 = \alpha^{i+1} q(\alpha)P_0, \quad i \in [1, n-1],\\
&S_1^i = c\alpha^j Q_1 = c\alpha^{j+1} Q_0, \quad j \in [0, n],\\
&S_2 = cQ_2 = c\alpha^2 f(\alpha)Q_0,\\
&S_3 = cQ_3 = \alpha f(\alpha)Q_0,\\
&PK = bR_1 = b\alpha^3 q(\alpha)f(\alpha)P_0.
\end{aligned}$$

The parameters above can be computed from the elements in the (f, g, F)-MSE-DDH1 instances. $\mathcal{B}$ specifies a keyword w_θ and implicitly sets the parameters bounded up with keyword w from two cases below,

$$\begin{aligned}
\text{if } w = w_\theta,\ &\alpha P(w) = a_i\alpha Q_0,\\
&\alpha^2 P(w) = a_i\alpha^2 Q_0;\\
\text{if } w \neq w_\theta,\ &\alpha P(w) = a_i\alpha^2 Q_0,\\
&\alpha^2 P(w) = a_i\alpha^3 Q_0.
\end{aligned}$$

Then $\mathcal{B}$ defines the public parameters as

$$PP = (R_1, \{R_2^i\}, P_3, \{S_1^j\}, S_2, S_3, \{\alpha^l P(w)\}),$$

where $i \in [1, n-1]$, $j \in [0, n]$, $l \in [1, 2]$. $\mathcal{B}$ sends (PP, PK) to $\mathcal{A}$.

Phase 1. $\mathcal{A}$ can issue a series of queries as follows.

Private Key Query. $\mathcal{A}$ asks for the private key query by sending ID_i to $\mathcal{B}$, where $ID_i \neq ID^*$. $\mathcal{B}$ responds $d_{ID_i} = \alpha^2 f_i(\alpha)q(\alpha)P_0$ to $\mathcal{A}$. One can verify that

$$d_{ID_i} = \frac{1}{\alpha + ID_i}P_1 = \frac{\alpha^2 f(\alpha)q(\alpha)}{\alpha + ID_i}P_0 = \alpha^2 f_i(\alpha)q(\alpha)P_0,$$

where $\alpha^2 f_i(\alpha)q(\alpha)P_0$ can be computed from elements in (f, g, F)-MSE-DDH1 instances.

Trapdoor Query. $\mathcal{A}$ asks for the trapdoor query by sending $(w, \{ID_i\}_{i=1}^n)$ to $\mathcal{B}$. $\mathcal{B}$ runs the Trapdoor algorithm and responds to $\mathcal{A}$ with simulated results.

1) If $ID^* \in \{ID_i\}_{i=1}^n$, $\mathcal{B}$ responds with the trapdoor.

• If $w = w_\theta$, $\mathcal{B}$ reports failure and terminates.

• Otherwise, we know $h_i = a_i\alpha Q_0$. $\mathcal{B}$ chooses random numbers k_1, $k_2 \in \mathbb{Z}_p^*$ and responds the trapdoor as $T_w = (U_1, V_1, W_1, U_2, V_2, W_2)$, where

$$\begin{aligned}
&U_1 = a_i b\alpha^3 Q_0 + k_1 c\alpha \prod_{i=1}^n(\alpha + ID_i)Q_0,\\
&V_1 = -k_1 c\alpha^2 f(\alpha)Q_0,\\
&W_1 = k_1\alpha f(\alpha)Q_0,\\
&U_2 = a_i b\alpha^2 Q_0 + k_2 c\alpha \prod_{i=1}^n(\alpha + ID_i)Q_0,\\
&V_2 = -k_2 c\alpha^2 f(\alpha)Q_0,\\
&W_2 = k_2\alpha f(\alpha)Q_0.
\end{aligned}$$

All items can be either directly obtained or indirectly computed from the elements of (f, g, F)-MSE-DDH1 instances.

One can verify the above trapdoor by implicitly using $P(w) = a_i\alpha Q_0$,

$$\begin{aligned} U_1 &= b\alpha^2 P(w) + k_1 c(\alpha + ID_1) + ... + (\alpha + ID_n)Q_1 \\ &= a_i b\alpha^3 Q_0 + k_1 c\alpha \prod_{i=1}^{n} (\alpha + ID_i)Q_0, \\ V_1 &= -k_1 S_2 \\ &= -k_1 c\alpha^2 f(\alpha)Q_0, \\ W_1 &= k_1 S_3 \\ &= k_1 \alpha f(\alpha)Q_0, \\ U_2 &= b\alpha P(w) + k_2 c(\alpha + ID_1) + ... + (\alpha + ID_n)Q_1 \\ &= a_i b\alpha^2 Q_0 + k_2 c\alpha \prod_{i=1}^{n} (\alpha + ID_i)Q_0, \\ V_2 &= -k_2 S_2 \\ &= -k_2 c\alpha^2 f(\alpha)Q_0, \\ W_2 &= k_2 S_3 \\ &= k_2 \alpha f(\alpha)Q_0. \end{aligned}$$

2) If $ID^* \notin \{ID_i\}_{i=1}^n$, $\mathcal{B}$ responds with the following trapdoor.

• If $w \neq w_\theta$, $\mathcal{B}$ responds as above.

• If $w = w_\theta$, we have $P(w) = a_i Q_0$.

Simulator $\mathcal{B}$ chooses random numbers $k_1,\ k' \in \mathbb{Z}_p^*$, and computes the trapdoor as $T_w = (U_1, V_1, W_1, U_2, V_2, W_2)$, where

$$\begin{aligned} &U_1 = a_i b\alpha^2 Q_0 + k_1 c\alpha \prod_{i=1}^{n}(\alpha + ID_i)Q_0, \\ &V_1 = -k_1 c\alpha^2 f(\alpha)Q_0, \\ &W_1 = k_1 \alpha f(\alpha)Q_0, \\ &U_2 = a_i(b\alpha Q_0 + mcQ_0) + k' c\alpha \prod_{i=1}^{n}(\alpha + ID_i)Q_0, \\ &V_2 = -a_i mc\alpha(\alpha + ID_{n+1})Q_0 - k' c\alpha^2 f(\alpha)Q_0, \\ &W_2 = a_i m(\alpha + ID_{n+1})Q_0 + k'\alpha f(\alpha)Q_0. \end{aligned}$$

Each item can be simulated according to the (f, g, F)-MSE-DDH1 instances.

One can verify the trapdoor by implicitly setting $k_2 = \frac{ma_i}{\alpha \prod_{i=1}^{n}(\alpha+ID_i)} + k'$, and then

$$\begin{aligned} U_1 &= b\alpha^2 P(w) + k_1 c(\alpha + ID_1) + ... + (\alpha + ID_n)Q_1 \\ &= a_i b\alpha^2 Q_0 + k_1 c\alpha \prod_{i=1}^{n}(\alpha + ID_i)Q_0, \\ V_1 &= -k_1 S_2 \\ &= -k_1 c\alpha^2 f(\alpha)Q_0, \\ W_1 &= k_1 S_3 \\ &= k_1 \alpha f(\alpha)Q_0, \\ U_2 &= b\alpha P(w) + k_2 c(\alpha + ID_1) + ... + (\alpha + ID_n)Q_1 \\ &= a_i(b\alpha Q_0 + mcQ_0) + k' c\alpha \prod_{i=1}^{n}(\alpha + ID_i)Q_0, \\ V_2 &= -k_2 S_2 \\ &= -a_i mc\alpha(\alpha + ID_{n+1})Q_0 - k' c\alpha^2 f(\alpha)Q_0, \\ W_2 &= k_2 S_3 \\ &= a_i m(\alpha + ID_{n+1})Q_0 + k'\alpha f(\alpha)Q_0. \end{aligned}$$

Ciphertext Query. $\mathcal{A}$ asks for the ciphertext query by sending (w, ID_i) to $\mathcal{B}$. $\mathcal{B}$ runs the Encrypt algorithm and responds to $\mathcal{A}$ with the simulated result.

1) If $ID_i \neq ID^*$, $\mathcal{A}$ can obtain the private key for ID_i, and then he/she can generate the ciphertext associated with any keyword.

2) If $ID_i = ID^*$, $\mathcal{B}$ responds with the ciphertext. We know $P(w_\theta) = a_\theta Q_0$ and $P(w \neq w_\theta) = a_i Q_0$. $\mathcal{B}$ picks $r'' \in_R \mathbb{Z}_p$ and responds with the searchable ciphertext $PEKS_w = (C_1, C_2, C_{3,i}, C_4)$ to $\mathcal{A}$, where

$$\begin{aligned} &C_1 = r'' s\alpha f(\alpha)P_0, \\ &C_2 = r'' sc\alpha q(\alpha)P_0, \\ &C_{3,i} = r'' s\alpha^i q(\alpha)P_0, \quad i \in [1, n-1], \\ &C_4 = \begin{cases} e(P_0,\ Q_0)^{r'' a_\theta sb\alpha^2 f(\alpha)q(\alpha)}, \text{ if } w = w_\theta, \\ e(P_0,\ Q_0)^{r'' a_i sb\alpha^3 f(\alpha)q(\alpha)}, \text{ otherwise.} \end{cases} \end{aligned}$$

These items can be obtained from the elements in (f, g, F)-MSE-DDH1 instances. One can verify it by implicitly setting $r' = \frac{r'' s}{\alpha}$, and then

$$\begin{aligned} C_1 &= \frac{r'}{\alpha + ID^*} P_1 = r'' s\alpha f(\alpha)P_0, \\ C_2 &= r' P_3 = r'' sc\alpha q(\alpha)P_0, \\ C_{3,i} &= r'\alpha^i P_2 = r'' s\alpha^i q(\alpha)P_0, \quad i \in [1, n-1], \\ C_4 &= e(b\alpha P_1, P(w_\theta))^{r'} \\ &= \begin{cases} e(P_0,\ Q_0)^{r'' a_\theta sb\alpha^2 f(\alpha)q(\alpha)}, \text{ if } w = w_\theta, \\ e(P_0,\ Q_0)^{r'' a_i sb\alpha^3 f(\alpha)q(\alpha)}, \text{ otherwise.} \end{cases} \end{aligned}$$

Challenge. $\mathcal{A}$ produces two equal length keywords w_0, w_1 that it wishes to be challenged on and sends $(w_0,\ w_1,\ ID^*)$ to $\mathcal{B}$. $\mathcal{A}$ did not previously query the private key for ID^*, or the trapdoor for w_0, w_1 for $\{ID_i\}_{i=1}^n$, where $ID^* \in \{ID_i\}_{i=1}^n$. $\mathcal{B}$ responds as follows.

• If $w_\theta \notin \{w_0, w_1\}$, $\mathcal{B}$ outputs failure and terminates.

• Otherwise, $\mathcal{B}$ picks w_θ from $w_0,\ w_1$, such that $h_\theta = a_\theta Q_0$ is implicitly set. $\mathcal{B}$ responds with the challenge ciphertext $PEKS^*_{w_\theta} = (C_1, C_2, C_{3,i}, C_4)$ to $\mathcal{A}$, where

$$C_1 = r\alpha f(\alpha)P_0,$$

$$C_2 = rc\alpha q(\alpha)P_0,$$
$$C_{3,i} = r\alpha^i q(\alpha)P_0, \quad i \in [1, n-1],$$
$$C_4 = Z^{a_\theta}.$$

These items can be obtained from the elements in (f, g, F)-MSE-DDH1 instances.

Note that if $Z = e(P_0,\ Q_0)^{rb\alpha^2 f(\alpha)q(\alpha)}$, by setting $r' = \frac{r}{\alpha}$, one can verify that

$$C_1 = \frac{r'}{\alpha + ID^*}P_1 = r\alpha f(\alpha)P_0,$$
$$C_2 = r'P_3 = rc\alpha q(\alpha)P_0,$$
$$C_{3,i} = r'R_2^i = r\alpha^i q(\alpha)P_0, \quad i \in [1, n-1],$$
$$C_4 = e(PK, P(w_\theta))^{r'} = e(P_0, Q_0)^{rb\alpha^2 f(\alpha)q(\alpha)a_\theta} = Z^{a_\theta}.$$

Phase 2. $\mathcal{A}$ continues to query the private key for $ID_i \neq ID^*$, the trapdoor for $\{ID_i\}_{i=1}^n$ with $ID^* \notin \{ID_i\}_{i=1}^n$ or the trapdoor for $w_i \neq w_0, w_1$ for $\{ID_i\}_{i=1}^n$ with $ID^* \in \{ID_i\}_{i=1}^n$, and the ciphertext for any ID_i, w_i. $\mathcal{C}$ responds as phase 1.

Guess. $\mathcal{A}$ outputs its guess θ' and wins the game if $\theta' = \theta$.

This completes the description of our simulation. We will analyze the advantage of $\mathcal{B}$ to solve the hard problem. If $\mathcal{B}$ does not abort then $|\Pr[\theta' = \theta] - \frac{1}{2}| \geqslant \epsilon$. The probability is over the random bits used by $\mathcal{A}$ and $\mathcal{B}$ as follows, where $\mathcal{B}$'s running time is approximately the same with $\mathcal{A}$'s. According to the above process, the probability that a trapdoor query causes $\mathcal{B}$ to abort is $1/(q_T + 1)$ and the private key query and the ciphertext query do not cause $\mathcal{B}$'s aborting. Suppose $\mathcal{A}$ makes a total of q_T trapdoor queries, the probability that $\mathcal{B}$ does not abort as a result of all queries is at least $(1 - 1/(q_T + 1))^{q_T} \geqslant 1/e$ in phase 1 or phase 2. In the challenge phase, $\mathcal{B}$ will abort if $\mathcal{A}$ can produce w_0, w_1 with $w_\theta \notin \{w_0, w_1\}$. Therefore, the probability that $\mathcal{B}$ aborts is $\Pr[w_\theta = w_i] = 1/(q_T + 1)$ for $i = 0, 1$. Since the values of w_0, w_1 are independent of each other, we have the probability that $\mathcal{B}$ does not abort is $\Pr[w_\theta \neq w_0, w_1] = (1-1/(q_T+1))^2 \leqslant 1-1/q_T$. Hence, the probability that $\mathcal{B}$ does not abort is at least $1/q_T$. Observe that since $\mathcal{A}$ can never query for the challenge identity ID^* and keywords w_0, w_1, we have $\mathcal{B}$'s advantage is at least ϵ/eq_T. □

Theorem 2. *The SEK-IA scheme is indistinguishable against keyword guessing attacks in game* 2 *if* (f, g, F)*-MSE-DDH*2 *problem is intractable for any PPT algorithm.*

Proof. Assume that there exists a PPT adversary $\mathcal{A}$ in game 2, who can attack our scheme with advantage ϵ. We build a simulator $\mathcal{B}$, who has advantage ϵ/eq_T against (f, g, F)-MSE-DDH2 problem. $\mathcal{B}$'s running time is approximately the same with $\mathcal{A}$'s.

Init. The adversary $\mathcal{A}$ declares the challenge identity set $\{ID_i^*\}_{i=1}^n$. We denote a general identity set as $\{ID_i\}_{i=1}^n$ and assume $\{ID_i^*\}_{i=1}^n \cap \{ID_i\}_{i=1}^n = \emptyset$.

Setup. The simulator $\mathcal{B}$ is given a group system $(p, \mathbb{G}, \mathbb{G}_T, e(\cdot,\cdot))$ as input, with the (f, g, F)-MSE-DDH2 instances. We also have two coprime polynomials f and q, deg f = deg $q = n$, with their pairwise distinct roots, and $f(0) \neq 0, q(0) \neq 0$. $\mathcal{B}$ is further given $Z_1, Z_2 \in \mathbb{G}$, where Z_1 is either equal to $b\alpha^4 R_0 + k_1^* cf(\alpha)Q_0$ or equal to some random element of $\mathbb{G}$, and Z_2 is either equal to $b\alpha^3 R_0 + k_2^* cf(\alpha)Q_0$ or equal to some random element of $\mathbb{G}$. For simplicity, we state that f and q are unitary polynomials, but this is not a mandatory requirement,

$$f(x) = \prod_{i=1}^n (x + ID_i^*),\ q(x) = \prod_{i=1}^n (x + ID_i).$$

For $i \in [1, n]$, we set $f_i(x) = \frac{f(x)}{x+ID_i^*}$, $q_i(x) = \frac{q(x)}{x+ID_i}$.

To generate the system parameters, the simulator $\mathcal{B}$ formally sets $\{P_i\}_{i\in[1,3]}, \{Q_i\}_{i\in[1,3]}$ such that $e(P_1, Q_1) = e(P_2, Q_2) = e(P_3, Q_3)$, but some items can be without computing, since they do not need to be published,

$$P_1 = \alpha^2 f(\alpha)q(\alpha)P_0,\ Q_1 = Q_0,$$
$$P_2 = c\alpha f(\alpha)P_0,\ Q_2 = \frac{\alpha}{c}q(\alpha)Q_0,$$
$$P_3 = c\alpha^2 f(\alpha)P_0,\ Q_3 = \frac{1}{c}q(\alpha)Q_0.$$

And then $\mathcal{B}$ gains some public parameters as

$$R_1 = \alpha P_1 = \alpha^3 f(\alpha)q(\alpha)P_0,$$
$$R_2^i = \alpha^i P_2 = c\alpha^{i+1} f(\alpha)P_0,$$
$$S_1^j = c\alpha^j Q_1 = c\alpha^j Q_0,$$
$$S_2 = cQ_2 = \alpha q(\alpha)Q_0,$$
$$S_3 = cQ_3 = q(\alpha)Q_0,$$
$$PK = bR_1 = b\alpha^3 f(\alpha)q(\alpha)P_0,$$

where $i \in [1, n-1], j \in [0, n]$. The parameters above can be computed from the elements in the (f, g, F)-MSE-DDH2 instances. $\mathcal{B}$ specifies a keyword w_θ and implicitly sets the parameters associated with keyword

w from two cases below,

$$\begin{aligned} &\text{if } w = w_\theta,\ \alpha P(w) = a_\theta \alpha^3 R_0,\\ &\qquad\qquad \alpha^2 P(w) = a_\theta \alpha^4 R_0;\\ &\text{if } w \neq w_\theta,\ \alpha P(w) = a_i \alpha R_0,\\ &\qquad\qquad \alpha^2 P(w) = a_i \alpha^2 R_0. \end{aligned}$$

Then $\mathcal{B}$ defines the public parameters as

$$PP = \left(R_1, \{R_2^i\}, P_3, \{S_1^j\}, S_2, S_3, \{\alpha^l P(w)\}\right),$$

where $i \in [1, n-1]$, $j \in [0, n]$, $l \in [1, 2]$. $\mathcal{B}$ sends (PP, PK) to $\mathcal{A}$.

Phase 1. $\mathcal{A}$ can issue a series of queries as follows.

Private Key Query. $\mathcal{A}$ asks for the private key query by sending ID_i to $\mathcal{B}$, where $ID_i \notin \{ID_i^*\}_{i=1}^n$. $\mathcal{B}$ responds $d_{ID_i} = \alpha^2 f(\alpha) q_i(\alpha) P_0$ to $\mathcal{A}$ by running algorithm S-KeyGen, where $\alpha^2 f(\alpha) q_i(\alpha) P_0$ can be computed from elements in (f, g, F)-MSE-DDH2 instances. One can verify that

$$d_{ID_i} = \frac{1}{\alpha + ID_i} P_1 = \alpha^2 f(\alpha) q_i(\alpha) P_0.$$

Ciphertext Query. $\mathcal{A}$ asks for the ciphertext query by sending w, ID_i to $\mathcal{B}$. $\mathcal{B}$ responds the ciphertext $PEKS_w$ to $\mathcal{A}$ by running the Encrypt algorithm.

1) If $ID_i \notin \{ID_i^*\}_{i=1}^n$, the simulator $\mathcal{B}$ chooses $r' \in_R \mathbb{Z}_p^*$, and responds the ciphertext as $PEKS_w = (C_1, C_2, \{C_{3,i}\}, C_4)$ by just running the Encrypt algorithm, where

$$\begin{aligned} &C_1 = r' \frac{1}{\alpha + ID_i} P_1,\\ &C_2 = r' P_3,\\ &C_{3,i} = r' \alpha^i P_2, \quad i \in [1, n-1],\\ &C_4 = e(b\alpha P_1,\ P(w))^{r'}. \end{aligned}$$

2) If $ID_i \in \{ID_i^*\}_{i=1}^n$, $\mathcal{B}$ responds the queried ciphertext as follows.

• If $w = w_\theta$, $\mathcal{B}$ reports failure and terminates.

• Otherwise, we know $h_i = a_i R_0$. $\mathcal{B}$ chooses $r \in_R \mathbb{Z}_p^*$ and responds the ciphertext $PEKS_w = (C_1, C_2, C_{3,i}, C_4)$, where

$$\begin{aligned} &C_1 = r\alpha f_i(\alpha) q(\alpha) P_0,\\ &C_2 = rc\alpha f(\alpha) P_0,\\ &C_{3,i} = rc\alpha^i f(\alpha) P_0, \quad i \in [1, n-1],\\ &C_4 = \left(e(P_0,\ R_0)^{rb\alpha^2 f(\alpha) q(\alpha)}\right)^{a_i}. \end{aligned}$$

By implicitly setting $r' = \frac{r}{\alpha}$, one can verify it as

$$\begin{aligned} &C_1 = \frac{r'}{\alpha + ID_i} P_1 = r\alpha f_i(\alpha) q(\alpha) P_0,\\ &C_2 = r' P_3 = rc\alpha f(\alpha) P_0,\\ &C_{3,i} = r' R_2^i = rc\alpha^i f(\alpha) P_0,\\ &C_4 = e(PK,\ P(w))^{r'} = e(P_0,\ R_0)^{rb\alpha^2 f(\alpha) q(\alpha)}. \end{aligned}$$

All ciphertext items are obtained from the elements in (f, g, F)-MSE-DDH2 instances.

Trapdoor Query. $\mathcal{A}$ asks for the trapdoor query by sending $(w, \{ID_i\}_{i=1}^n)$ to $\mathcal{B}$. $\mathcal{B}$ responds to $\mathcal{A}$ by running algorithm Trapdoor. We first denote F^i as the coefficient of α^i in the polynomial $\prod_{i=1}^n (\alpha + ID_i)$.

1) If $w = w_\theta$, we know $P(w_\theta) = a_\theta \alpha^2 Q_0$. $\mathcal{B}$ chooses $k_1', k_2' \in_R \mathbb{Z}_p^*$ and responds the trapdoor as $T_w = (U_1, V_1, W_1, U_2, V_2, W_2)$ to $\mathcal{A}$, where

$$\begin{aligned} U_1 &= a_\theta (b\alpha^4 R_0 + s_1 c\alpha^n Q_0) + \Sigma_{i=0}^{n-1} F^i s_1 c\alpha^i Q_0 +\\ &\quad k_1' c \prod_{i=1}^n (\alpha + ID_i) Q_1,\\ V_1 &= -a_\theta s_1 \alpha q(\alpha) Q_0 - k_1' c Q_2,\\ W_1 &= a_\theta s_1 q(\alpha) Q_0 + k_1' c Q_3,\\ U_2 &= a_\theta (b\alpha^3 R_0 + s_2 c\alpha^n Q_0) + \Sigma_{i=0}^{n-1} F^i s_2 c\alpha^i Q_0 +\\ &\quad k_2' c \prod_{i=1}^n (\alpha + ID_i) Q_1,\\ V_2 &= -a_\theta s_2 \alpha q(\alpha) Q_0 - k_2' c Q_2,\\ W_2 &= a_\theta s_2 q(\alpha) Q_0 + k_2' c Q_3. \end{aligned}$$

These items can be obtained from the elements in (f, g, F)-MSE-DDH2 instances.

One can verify it by implicitly setting $k_1 = a_\theta s_1 + k_1'$, $k_2 = a_\theta s_2 + k_2'$, and then

$$\begin{aligned} U_1 &= b\alpha^2 P(w_\theta) + k_1 c \prod_{i=1}^n (\alpha + ID_i) Q_1\\ &= a_\theta (b\alpha^4 R_0 + s_1 c\alpha^n Q_0) + \Sigma_{i=0}^{n-1} F^i a_\theta s_1 c\alpha^i Q_0 +\\ &\quad k_1' c \prod_{i=1}^n (\alpha + ID_i) Q_1,\\ V_1 &= -k_1 c Q_2\\ &= -a_\theta s_1 \alpha q(\alpha) Q_0 - k_1' c Q_2,\\ W_1 &= k_1 c Q_3\\ &= a_\theta s_1 q(\alpha) Q_0 + k_1' c Q_3,\\ U_2 &= b\alpha P(w_\theta) + k_2 c \prod_{i=1}^n (\alpha + ID_i) Q_1\\ &= a_\theta (b\alpha^3 R_0 + s_2 c\alpha^n Q_0) +\\ &\quad \Sigma_{i=0}^{n-1} F^i a_\theta s_2 c\alpha^i Q_0 +\\ &\quad k_2' c \prod_{i=1}^n (\alpha + ID_i) Q_1, \end{aligned}$$

$$\begin{aligned} V_2 &= -k_2cQ_2 \\ &= -a_\theta s_2\alpha q(\alpha)Q_0 - k_2'cQ_2, \\ W_2 &= k_2cQ_3 \\ &= a_\theta s_2 q(\alpha)Q_0 + k_2'cQ_3. \end{aligned}$$

2) If $w \neq w_\theta$, we know $P(w) = a_iQ_0$. $\mathcal{B}$ chooses $k_1', k_2' \in_R \mathbb{Z}_p^*$ and responds the trapdoor as $T_w = (U_1, V_1, W_1, U_2, V_2, W_2)$ to $\mathcal{A}$, where

$$\begin{aligned} U_1 &= a_i(b\alpha^2R_0 + s_3c\alpha^nQ_0) + \Sigma_{i=0}^{n-1}F^ia_is_3c\alpha^iQ_0 + \\ &\quad k_1'c\prod_{i=1}^{n}(\alpha + ID_i)Q_1, \\ V_1 &= -a_is_3\alpha q(\alpha)Q_0 - k_1'cQ_2, \\ W_1 &= a_is_3q(\alpha)Q_0 + k_1'cQ_3, \\ U_2 &= a_i(b\alpha R_0 + s_4c\alpha^nQ_0) + \Sigma_{i=0}^{n-1}F^ia_is_4c\alpha^iQ_0 + \\ &\quad k_2'c\prod_{i=1}^{n}(\alpha + ID_i)Q_1, \\ V_2 &= -a_is_4\alpha q(\alpha)Q_0 - k_2'cQ_2, \\ W_2 &= a_is_4q(\alpha)Q_0 + k_2'cQ_3. \end{aligned}$$

These items can be obtained from the elements in (f, g, F)-MSE-DDH2 instances. One can verify the trapdoor by implicitly setting $k_1 = a_\theta s_3 + k_1'$ and $k_2 = a_\theta s_4 + k_2'$. The method is the same as above.

Challenge. $\mathcal{A}$ provides two same length keywords w_0 and w_1 that it wishes to be challenged and sends $(w_0, w_1, \{ID_i^*\}_{i=1}^n)$ to $\mathcal{B}$. $\mathcal{A}$ did not previously ask for the private key for ID_i, where $ID_i \in \{ID_i^*\}_{i=1}^n$, or the ciphertext for w_0, w_1 for ID_i, where $ID_i \in \{ID_i^*\}_{i=1}^n$. $\mathcal{B}$ responds the challenge trapdoor to $\mathcal{A}$ as follows.

• If $w_\theta \notin \{w_0, w_1\}$, $\mathcal{B}$ outputs failure and terminates.

• Otherwise, $\mathcal{B}$ picks w_θ from w_0, w_1, such that $h_\theta = a_\theta\alpha^2R_0$. Then $\mathcal{B}$ chooses $k_1', k_2' \in_R \mathbb{Z}_p^*$ and responds the challenge trapdoor as $T_{w_\theta}^* = (U_1^*, V_1^*, W_1^*, U_2^*, V_2^*, W_2^*)$ to $\mathcal{A}$, where

$$\begin{aligned} U_1^* &= a_\theta Z_1 + k_1'c\Pi_{i\in[1,n]}(\alpha + ID_i^*)Q_1, \\ V_1^* &= -a_\theta k_1^*\alpha q(\alpha)Q_0 - k_1'cQ_2, \\ W_1^* &= a_\theta k_1^*q(\alpha)Q_0 + k_1'cQ_3, \\ U_2^* &= a_\theta Z_2 + k_2'c\Pi_{i\in[1,n]}(\alpha + ID_i^*)Q_1, \\ V_2^* &= -a_\theta k_2^*\alpha q(\alpha)Q_0 - k_2'cQ_2, \\ W_2^* &= a_\theta k_2^*q(\alpha)Q_0 + k_2'cQ_3. \end{aligned}$$

These items can be obtained from the elements in (f, g, F)-MSE-DDH2 instances.

Note that if the following conditions hold from the (f, g, F)-MSE-DDH2 instances, i.e.,

$$Z_1 = b\alpha^4R_0 + k_1^*cf(\alpha)Q_0,$$

$$Z_2 = b\alpha^3R_0 + k_2^*cf(\alpha)Q_0,$$

we implicitly set the coefficient

$$k_1 = a_\theta k_1^* + k_1',\ k_2 = a_\theta k_2^* + k_2',$$

and can verify the trapdoor

$$\begin{aligned} U_1^* &= b\alpha^2P(w) + k_1c(\alpha + ID_1^*) + \cdots + (\alpha + ID_n^*)Q_1 \\ &= a_\theta Z_1 + k_1'c\prod_{i=1}^{n}(\alpha + ID_i^*)Q_1, \\ V_1^* &= -k_1S_2 = -a_\theta k_1^*\alpha q(\alpha)Q_0 - k_1'cQ_2, \\ W_1^* &= k_1S_3 = a_\theta k_1^*q(\alpha)Q_0 + k_1'cQ_3, \\ U_2^* &= b\alpha P(w) + k_2c(\alpha + ID_1^*) + \cdots + (\alpha + ID_n^*)Q_1 \\ &= a_\theta Z_2 + k_2'c\prod_{i=1}^{n}(\alpha + ID_i^*)Q_1, \\ V_2^* &= -k_2S_2 = -a_\theta k_2^*\alpha q(\alpha)Q_0 - k_2'cQ_2, \\ W_2^* &= k_2S_3 = a_\theta k_2^*q(\alpha)Q_0 + k_2'cQ_3. \end{aligned}$$

Phase 2. $\mathcal{A}$ continues to query the private key for $ID_i \notin \{ID_i^*\}_{i=1}^n$, the ciphertext for $ID_i \notin \{ID_i^*\}_{i=1}^n$ or the ciphertext for ID_i, w_i, $ID_i \in \{ID_i^*\}_{i=1}^n, w_i \neq w_0, w_1$, and the trapdoor for any $w_i, \{ID_i\}_{i=1}^n$. $\mathcal{C}$ responds as phase 1.

Guess. $\mathcal{A}$ outputs its guess θ' and wins the game if $\theta' = \theta$.

This completes the description of our simulation. The probability can be analyzed with the similar method to Theorem 1. Suppose $\mathcal{A}$ makes a total of q_C ciphertext queries, we have that $\mathcal{B}$'s advantage is at least ϵ/eq_C. □

6 Evaluation and Discussion

6.1 Performance Evaluation

We evaluate the proposed SEK-IA scheme from the terms of the public parameter size, the searchable ciphertext size, and the trapdoor size. We only consider the case for single keyword in searchable ciphertext and trapdoor.

Fig.3 first shows the public parameter size over the sender identity set size, i.e., the maximum number of the senders in the trapdoor. We vary the sender identity set size from 0 to 30 and depict the curve of the public parameter size. When the sender identity set size is 30, the public parameter size runs up to 4 310 bytes. From Fig.3, we observe that the public parameter size keeps smoothly linear increasing with the sender identity set size.

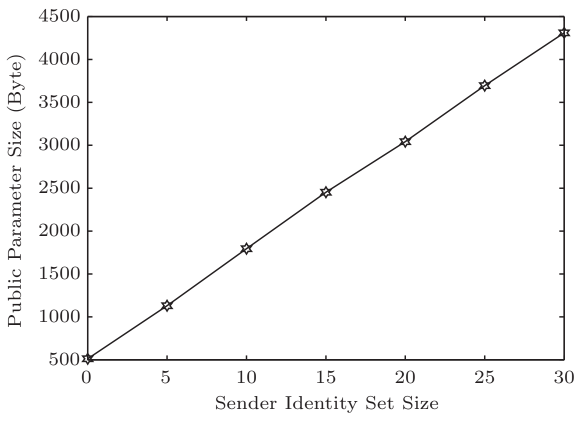

Fig.3. Public parameter size vs sender identity set size.

Fig.4 shows the searchable ciphertext size versus the sender identity set size. We consider the same method to vary the sender identity set size. Under the case that there are maximumly 30 senders allowed in the trapdoor, each searchable ciphertext costs 2 055 bytes. The searchable ciphertext size grows almost linearly with the sender identity set size. Fig.5 shows that the trapdoor size versus the sender identity set size. We handle the sender identity set size as above. From the curve, the trapdoor size remains almost constant, steadily fluctuating around 383 bytes, regardless of the sender identity set size. From the above analysis, we can see that our SEK-IA scheme achieves the constant-size trapdoor, while both the public parameter size and the searchable ciphertext size grow almost linearly with the maximum number of senders in the trapdoor.

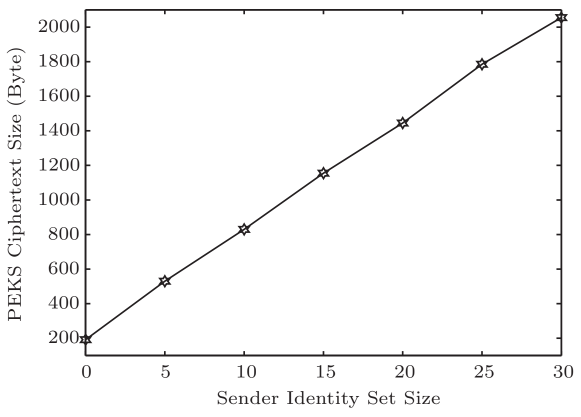

Fig.4. Searchable ciphertext size vs sender identity set size.

We also take the simulation about the computation cost of the trapdoor generation with varying the sender identity set size. We use OpenSSL library to test the running time, where the device is with Inter® Dual-Core™ E5300 2.60 GHz with 3 GB, the operation system is Windows 7 and the program language is C. During our evaluation, the sender identity set size varies from 1 to 30, and the experimental result is shown in Fig.6. In Fig.6, the solidline curve with stars represents our proposal and the dotted line curve with squares means the case that one trapdoor only searches one ciphertext from one sender, i.e., separate operation to sender identity set. From the comparison with the separate operation, our SEK-IA costs less time to generate the trapdoor when the sender identity set size grows.

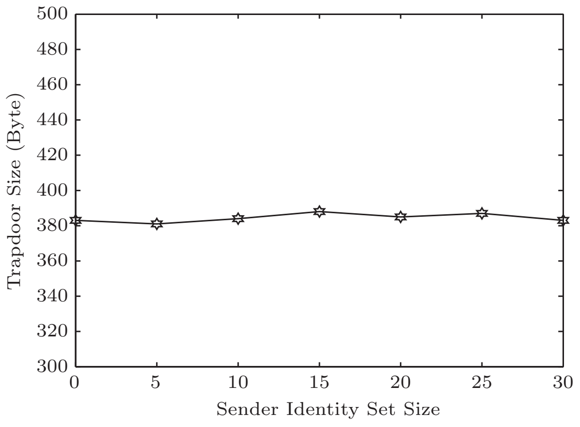

Fig.5. Trapdoor size vs sender identity set size.

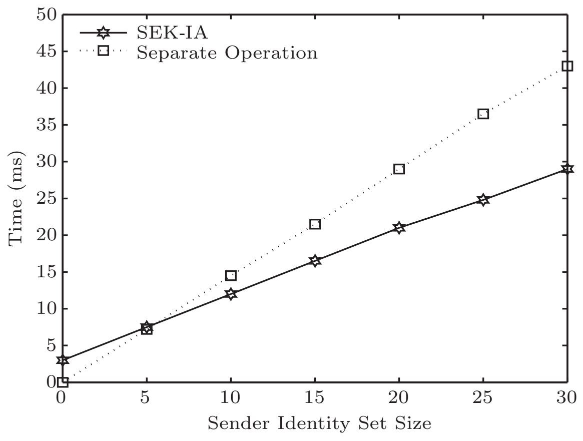

Fig.6. Trapdoor computation cost vs sender identity set size.

Comparison. We compare the SEK-IA scheme with other related schemes in asymmetric setting in terms of functionality and complexity. The comparison results are shown in Table 2, where n is the number of senders. For the functionality, SEK-IA can achieve insider attack resistance while other schemes at most resist outsider attacks. For the complexity, our SEK-IA scheme attains the constant-complexity trapdoor, not more than that of other schemes, although it is at the expense

Table 2. Performance Comparison with Related Work

Term	CKA	Outsider Attack	Insider Attack	Test Entity	Ciphertext Size	Trapdoor Size	Server-Receiver Bandwidth
[2]	✓	×	×	Any server	$O(1)$	$O(1)$	$O(n)$
[5]	✓	✓	×	Designated server	$O(1)$	$O(1)$	$O(n)$
[6]	✓	✓	×	Designated server	$O(1)$	$O(1)$	$O(n)$
SEK-IA	✓	✓	✓	Any server	$O(n)$	$O(1)$	$O(1)$

of linear-complexity ciphertext. The proposed SEK-IA minimized the bandwidth between the server and the receiver when searching data from multiple senders, where the bandwidth complexity is $O(1)$ independent of the number of senders.

6.2 Discussion and Future Work

We give some discussions about our SEK-IA scheme in terms of parameter size, TTP and the implementation.

6.2.1 Parameter Size

In our proposed SEK-IA scheme, the receiver submits a constant size trapdoor to the server for searching multiple ciphertexts from different senders. The communication cost is constant between the receiver and the server. The constant-size trapdoor is achieved at the expense of the linear-size ciphertext and the linear-size public parameter. If the size of both the ciphertext and the trapdoor is constant, the public parameter will not increase linearly, but the ciphertext and the trapdoor will be associated with only one sender. As a result, the receiver needs to submit multiple trapdoors to search data from different senders. The bandwidth between the receiver and the server increases linearly with the number of associated senders in the trapdoor. However, the original motivation of the PEKS is to reduce the bandwidth between the receiver and the server as small as possible and improve the data utilization for the receiver. Compared with the above method, our proposal can significantly reduce the bandwidth overhead between the server and the receiver when searching large databases. This is an inherent tradeoff among the public parameter size, the ciphertext size and the trapdoor size. Our work is the first one to solve insider attacks in public key based search over encrypted data. How to construct an unbounded searchable encrypted keywords scheme with constant size ciphertext is one future research work.

6.2.2 TTP

In our proposed SEK-IA scheme, a TTP is required to issue the private key to the sender. For a large system, a single TTP might become a potential bottleneck. The system we consider consists of an authority and group users (in an enterprise for instance), which can locally maintain a server that provides the computing service for users in order to solve the potential issue. We can also adopt various security techniques to guarantee the local server to be trusted and the communication between the server and users to be secured (e.g., establishing a secure channel between the server and users in the local area network). This approach is quite common in practice. To reduce trust on TTP, we could deploy multiple TTPs for private key generation, thereby addressing both security and efficiency. How to deploy multiple TTPs in details will be provided in our future work.

6.2.3 Outline of Prototyping

Our SEK-IA scheme is derived from cryptography, while it might be implemented in practice. Intuitively for implementation, the entries on the database management system (DBMS) can be stored in the form of "encrypted keyword || pointer 1 || pointer 2 || $\cdots$" and we can associate each entry with a keyword. An entry includes one (encrypted) keyword field and several file fields. Each file field records a pointer pointing to the file that contains the corresponding keyword. For a keyword search, the search function works with the Test algorithm, so that the DBMS can return all entries, for the matching keywords. The corresponding files for the matching keywords can be read. However, existing DBMS only supports exact and range queries; therefore we cannot directly use SQL statements to implement the Test algorithm which is the probabilistic encryption of the keywords. Instead, it might be feasible to realize this functionality in the following two ways. 1) Utilizing a trusted middle-layer server. Namely, we implement a trusted computing layer outside the database. Given a trapdoor, the server picks one entry from the database, runs the Test algorithm on the encrypted keyword and trapdoor, and returns the corresponding pointers if the output is true. 2) Hacking DBMS. We can build on an existing open-source database system (e.g., SQLite), and modify its code to support the Test algorithm. Both approaches require to scan the whole database, and cannot work together with existing index-based optimization (e.g., B+ tree). The concrete implementation will be considered in our future work.

7 Conclusions

We proposed the first searchable encrypted keyword against insider attacks. In our SEK-IA framework, the server cannot launch the insider attacks to distinguish the keyword from a trapdoor since it is unable to generate a searchable ciphertext without the sender's private key. Under the new framework, we built the SEK-IA security model capturing chosen keyword attacks and keyword guessing attacks, even if the adversary is the insider attacker. We constructed a provably secure SEK-IA scheme with a constant size trapdoor. The scheme empowers the receiver to search multiple data from different senders with one trapdoor. We formally proved that the proposed scheme is semantically secure against chosen keyword attacks and indistinguishable against keyword guessing attacks under the security model. With the performance evaluation, the proposed SEK-IA scheme achieves the constant-size communication cost between the receiver and the server, independent of the maximum number of the senders allowed in a trapdoor.

References

[1] Song D X, Wagner D, Perrig A. Practical techniques for searches on encrypted data. In *Proc. the IEEE Symposium on Security and Privacy*, May 2000, pp.44-55.

[2] Boneh D, Di Crescenzo G, Ostrovsky R, Persiano G. Public key encryption with keyword search. In *Lecture Notes in Computer Science 3027*, Cachin C, Camenisch J L (eds.), Springer, 2004, pp.506-522.

[3] Yau W C, Heng S H, Goi B M. Off-line keyword guessing attacks on recent public key encryption with keyword search schemes. In *Lecture Notes in Computer Science 5060*, Rong C M, Jaatun M C, Sandnes F E, Yang L T, Ma J H (eds.), Springer, 2008, pp.100-105.

[4] Baek J, Safavi-Naini R, Susilo W. Public key encryption with keyword search revisited. In *Lecture Notes in Computer Science 5072*, Gervasi O, Murgante B, Laganà A, Taniar D, Mun Y, Gavrilova M L (eds.), Springer, 2008, pp.1249-1259.

[5] Rhee H S, Park J H, Susilo W, Lee D H. Improved searchable public key encryption with designated tester. In *Proc the 4th International Symposium on Information, Computer and Communications Security*, March 2009, pp.376-379.

[6] Fang L M, Susilo W, Ge C P, Wang J D. Public key encryption with keyword search secure against keyword guessing attacks without random oracle. *Information Sciences*, 2013, 238: 221-241.

[7] Curtmola R, Garay J A, Kamara S, Ostrovsky R. Searchable symmetric encryption: Improved definitions and efficient constructions. In *Proc. the 13th ACM Conference on Computer and Communications Security*, October 30-November 3, 2006, pp.79-88.

[8] Kamara S, Papamanthou C, Roeder T. Dynamic searchable symmetric encryption. In *Proc. the ACM Conference on Computer and Communications Security*, October 2012, pp.965-976.

[9] Cash D, Jarecki S, Jutla C, Krawczyk H, Rosu M, Steiner M. Highly-scalable searchable symmetric encryption with support for Boolean queries. In *Lecture Notes in Computer Science 8042*, Canetti R, Garay J A (eds.), Springer, 2013, pp.353-373.

[10] Tang Q. Nothing is for free: Security in searching shared and encrypted data. *IEEE Transactions on Information Forensics and Security*, 2014, 9(11): 1943-1952.

[11] Park D J, Kim K, Lee P J. Public key encryption with conjunctive field keyword search. In *Lecture Notes in Computer Science 3325*, Lim C H, Yung M (eds.), Springer, 2005, pp.73-86.

[12] Bethencourt J, Song D X, Waters B. New constructions and practical applications for private stream searching (extended abstract). In *Proc. IEEE Symposium on Security and Privacy*, May 2006, pp.132-139.

[13] Boneh D, Waters B. Conjunctive, subset, and range queries on encrypted data. In *Lecture Notes in Computer Science 4392*, Vadhan S P (ed.), Springer, 2007, pp.535-554.

[14] Sedghi S, van Liesdonk P, Nikova S, Hartel P, Jonker W. Searching keywords with wildcards on encrypted data. In *Lecture Notes in Computer Science 6280*, Garay J A, De Prisco R (eds.), Springer, 2010, pp.138-153.

[15] Abdalla M, Bellare M, Catalano D, Kiltz E, Kohno T, Lange T, Malone-Lee J, Neven G, Paillier P, Shi H X. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. In *Lecture Notes in Computer Science 3621*, Shoup V (ed.), Springer, 2005, pp.205-222.

[16] Xu P, Jin H, Wu Q H, Wang W. Publickey encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack. *IEEE Transactions on Computers*, 2013, 62(11): 2266-2277.

[17] Camenisch J, Kohlweiss M, Rial A, Sheedy C. Blind and anonymous identity-based encryption and authorised private searches on public key encrypted data. In *Lecture Notes in Computer Science 5443*, Jarecki S, Tsudik G (eds.), Springer, 2009, pp.196-214.

[18] Zheng Q J, Xu S H, Ateniese G. VABKS: Verifiable attribute-based keyword search over outsourced encrypted data. In *Proc. IEEE INFOCOM*, April 27-May 2, 2014, pp.522-530.

[19] Shi J, Lai J Z, Li Y J, Deng R H, Weng J. Authorized keyword search on encrypted data. In *Lecture Notes in Computer Science 8712*, Kutyłowski M, Vaidya J (eds.), Springer, 2014, pp.419-435.

[20] Li J W, Li J, Chen X F, Jia C F, Liu Z L. Efficient keyword search over encrypted data with fine-grained access control in hybrid cloud. In *Lecture Notes in Computer Science 7645*, Xu L, Bertino E, Mu Y (eds.), Springer, 2012, pp.490-502.

[21] Byun J W, Rhee H S, Park H A, Lee D H. Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. In *Lecture Notes in Computer Science 4165*, Jonker W, Petković M (eds.), Springer, 2006, pp.75-83.

[22] Bösch C, Hartel P, Jonker W, Peter A. A survey of provably secure searchable encryption. *ACM Computing Surveys*, 2015, 47(2): Article No. 18.

[23] Shen E, Shi E, Waters B. Predicate privacy in encryption systems. In *Lecture Notes in Computer Science 5444*, Reingold O (ed.), Springer, 2009, pp.457-473.

[24] Boneh D, Boyen X, Goh E J. Hierarchical identity based encryption with constant size ciphertext. In *Lecture Notes in Computer Science 3494*, Cramer R (ed.), Springer, 2005, pp.440-456.

[25] Delerablée C, Pointcheval D. Dynamic threshold public-key encryption. In *Lecture Notes in Computer Science 5157*, Wagner D (ed.), Springer, 2008, pp.317-334.

Peng Jiang received her B.S. degree in mathematics from Southeast University, Nanjing, in 2010. She is currently a Ph.D. candidate in the Department of State Key Laboratory of Networking and Switch Technology, Beijing University of Posts and Telecommunications, Beijing. She is also a visiting Ph.D. student at the School of Computing and Information Technology, University of Wollongong, Wollongong. Her research interests include information security and privacy concerns.

Yi Mu received his Ph.D. degree in computer science from Australian National University, Canberra, in 1994. He is currently a full professor at the University of Wollongong, Wollongong. He is the editor-in-chief of the International Journal of Applied Cryptography and serves as an associate editor for many international journals. He has served in program committees for a number of international security conferences. He is a senior member of IEEE and a member of the IACR. His current research interests include cryptography and information security.

Fuchun Guo received his Ph.D. degree in computer science from University of Wollongong, Wollongong, in 2013. He is currently a lecturer of the School of Computing and Information Technology, University of Wollongong, Wollongong. His primary research interest is the public-key cryptography, in particular, protocols, encryption and signature schemes, and security proof.

Qiao-Yan Wen received her Ph.D. degree in cryptography from Xidian University, Xi'an, in 1997. She is a professor of the State Key Laboratory of Networking and Switch Technology, Beijing University of Posts and Telecommunications, Beijing. Her research interests include coding theory, cryptography, information security, Internet security, and applied mathematics.

Appendix Hard Problems and Intractability Analysis

As the analysis made in [24-25], our security proof can be reduced to the general Diffe-Hellman exponent problems. In our case, we present two MSE-DDH (Multi-Sequence of Exponents Diffie-Hellman) problems, which are special instances of the general DHE problems. Our new hard problems still preserve hardness and the intractability analysis of them are given.

(f, g, F)-MSE-DDH1 Problem. Let n be integers and $(p, \mathbb{G}, \mathbb{G}_T, e(\cdot,\cdot))$ be a bilinear map group system. Let P_0, Q_0 be the generators of $\mathbb{G}$. Given two random coprime polynomials f and q with pairwise distinct roots, of order $n+1$ and 1 respectively, $f(0) \neq 0, q(0) \neq 0$, and several sequences of group elements,

$$\begin{array}{rcr}
\alpha^2 P_0, & \cdots, & \alpha^{n+5} P_0, \\
b\alpha^2 P_0, & \cdots, & b\alpha^{n+5} P_0, \\
r\alpha q(\alpha) P_0, & \cdots, & r\alpha^{n-1} q(\alpha) P_0, \\
c\alpha^2 P_0, & c\alpha^3 P_0, & rc\alpha q(\alpha) P_0, \\
s\alpha q(\alpha) P_0, & \cdots, & s\alpha^{n-1} q(\alpha) P_0, \\
r\alpha f(\alpha) P_0, & s\alpha f(\alpha) P_0, & sc\alpha q(\alpha) P_0, \\
\alpha Q_0, & \cdots, & \alpha^{n+2} Q_0, \\
c\alpha Q_0, & \cdots, & c\alpha^{n+3} Q_0, \\
b\alpha^2 Q_0, & b\alpha^3 Q_0, & b\alpha Q_0 + mc Q_0, \\
m Q_0, & m\alpha Q_0, & e(P_0, Q_0)^{sb\alpha^2 f(\alpha) q(\alpha)}, \\
mc\alpha Q_0, & mc\alpha^2 Q_0, & e(P_0, Q_0)^{sb\alpha^3 f(\alpha) q(\alpha)},
\end{array}$$

and $Z \in \mathbb{G}_T$, distinguish whether Z is equal to $e(P_0, Q_0)^{rb\alpha^2 f(\alpha)q(\alpha)}$ or equal to some random element of $\mathbb{G}$.

To any PPT algorithm, it is difficult to solve (f, g, F)-MSE-DDH1 problem.

(f, g, F)-MSE-DDH2 Problem. Let n be integers and $(p, \mathbb{G}, \mathbb{G}_T, e(\cdot,\cdot))$ be a bilinear map group system. Let P_0, Q_0, R_0 be the generators of $\mathbb{G}$. Given two random coprime polynomials f and g with pairwise distinct roots, of order n, $f(0) \neq 0, q(0) \neq 0$, and several sequences of group elements,

$$\begin{array}{rr}
\alpha^3 f(\alpha)q(\alpha)P_0, & b\alpha^3 f(\alpha)q(\alpha)P_0, q(\alpha)Q_0,\\
\alpha q(\alpha)Q_0, & k_1^* q(\alpha)Q_0, k_1^*\alpha q(\alpha)Q_0,\\
k_2^* q(\alpha)Q_0, & k_2^*\alpha q(\alpha)Q_0, s_1 q(\alpha)Q_0,\\
s_1\alpha q(\alpha)Q_0, & s_2 q(\alpha)Q_0, s_2\alpha q(\alpha)Q_0,\\
s_3 q(\alpha)Q_0, & s_3\alpha q(\alpha)Q_0, s_4 q(\alpha)Q_0,\\
s_4\alpha q(\alpha)Q_0, & e(P_0,\ R_0)^{rb\alpha^2 f(\alpha)q(\alpha)},\\
c\alpha^2 f(\alpha)P_0,\ \cdots, & c\alpha^n f(\alpha)P_0,\\
r\alpha q(\alpha)P_0,\ \cdots, & r\alpha^n q(\alpha)P_0,\\
rc\alpha f(\alpha)P_0,\ \cdots, & rc\alpha^{n-1} f(\alpha)P_0,\\
\alpha^2 f(\alpha)P_0,\ \cdots, & \alpha^{n+1} f(\alpha)P_0,\\
c\alpha Q_0,\ \cdots, & c\alpha^n Q_0,\\
s_1 c\alpha Q_0,\ \cdots, & s_1 c\alpha^{n-1}Q_0,\\
s_2 c\alpha Q_0,\ \cdots, & s_2 c\alpha^{n-1}Q_0,\\
s_3 c\alpha Q_0,\ \cdots, & s_3 c\alpha^{n-1}Q_0,\\
s_4 c\alpha Q_0,\ \cdots, & s_4 c\alpha^{n-1}Q_0,\\
\alpha R_0,\ \cdots, & \alpha^4 R_0,\\
b\alpha^4 R_0 + s_1 c\alpha^n Q_0, & b\alpha^3 R_0 + s_2 c\alpha^n Q_0,\\
b\alpha^2 R_0 + s_3 c\alpha^n Q_0, & b\alpha R_0 + s_4 c\alpha^n Q_0,
\end{array}$$

and $Z_1, Z_2 \in \mathbb{G}$, distinguish whether both Z_1 is equal to $b\alpha^4 R_0 + k_1^* cf(\alpha)Q_0$ or equal to some random element of $\mathbb{G}$, and Z_2 is either equal to $b\alpha^3 R_0 + k_2^* cf(\alpha)Q_0$ or equal to some random element of $\mathbb{G}$.

To any PPT algorithm, it is difficult to solve (f, g, F)-MSE-DDH2 problem.

Intractability of (f, g, F)-MSE-DDH1 Problem. Since P_0 and Q_0 are both the generators of group $\mathbb{G}$, we pose $Q_0 = \beta P_0$. Our problem can be reformulated as D, E, F,

$$D = \begin{pmatrix}
\alpha^2, & \cdots, & \alpha^{n+5},\\
b\alpha^2, & \cdots, & b\alpha^{n+5},\\
r\alpha q(\alpha), & \cdots, & r\alpha^{n-1}q(\alpha),\\
c\alpha^2, & c\alpha^3, & rc\alpha q(\alpha),\\
s\alpha q(\alpha), & \cdots, & s\alpha^{n-1}q(\alpha),\\
r\alpha f(\alpha), & s\alpha f(\alpha), & sc\alpha q(\alpha),\\
\beta\alpha, & \cdots, & \beta\alpha^{n+2},\\
c\beta\alpha, & \cdots, & c\beta\alpha^{n+3},\\
b\beta\alpha^2, & b\beta\alpha^3, & b\beta\alpha + mc\beta,\\
m\beta, & m\beta\alpha, & \\
mc\beta\alpha, & mc\beta\alpha^2 &
\end{pmatrix},$$

$$E = sb\beta\alpha^2 f(\alpha)q(\alpha), sb\beta\alpha^3 f(\alpha)q(\alpha),$$
$$F = rb\beta\alpha^2 f(\alpha)q(\alpha).$$

We need to show that F is independent of (D, E), i.e., no coefficients $\{x_{i,j}\}$ and y_1 exist such that $F = \Sigma x_{i,j}d_i d_j + \Sigma y_i e_i$, where the polynomials d_i, e_i are listed in D, E above respectively. By making all possible products of two polynomials from D which are multiples of $rb\beta$ and $rcm\beta$, we want to prove that no linear combination among the polynomials from the lists F' and F'' below leads to F,

$$F' = \begin{pmatrix}
rb\beta\alpha^3 f(\alpha), rb\beta\alpha^4 f(\alpha),\\
rb\beta\alpha^3 q(\alpha), \cdots, rb\beta\alpha^{n+2}q(\alpha),\\
(rb\beta\alpha + rmc\beta)\alpha f(\alpha),\\
(rb\beta\alpha + rmc\beta)\alpha q(\alpha), \cdots,\\
(rb\beta\alpha + rmc\beta)\alpha^{n-1}q(\alpha)
\end{pmatrix},$$

$$F'' = \begin{pmatrix}
rmc\beta\alpha^2 f(\alpha), & rmc\beta\alpha^3 f(\alpha),\\
rmc\beta\alpha q(\alpha), & \cdots,\ rmc\beta\alpha^{n+1}q(\alpha)
\end{pmatrix}.$$

Any such linear combination associated with $rb\beta$ can be written as the following equation:

$$\begin{aligned}
&rb\beta\alpha^2 f(\alpha)q(\alpha)\\
&= rb\beta\alpha^3 A(\alpha)f(\alpha) + B\alpha f(\alpha)(rb\beta\alpha + rmc\beta) +\\
&\quad rb\beta\alpha^3 C(\alpha)q(\alpha) + \alpha D(\alpha)q(\alpha)(rb\beta\alpha + rmc\beta),
\end{aligned}$$

where $A(\alpha), C(\alpha), D(\alpha)$ are polynomials and B is the constant, deg $A(\alpha) \leqslant 1$, deg $C(\alpha) \leqslant n-1$, deg $D(\alpha) \leqslant n-2$. We can simplify the equation corresponding to $rb\beta$ as

$$\begin{aligned} f(\alpha)q(\alpha) = {} & \alpha f(\alpha)A(\alpha) + Bf(\alpha) + \\ & \alpha q(\alpha)C(\alpha) + q(\alpha)D(\alpha), \end{aligned}$$

which is also written as

$$(f(\alpha) - \alpha C(\alpha) - D(\alpha))q(\alpha) = (\alpha A(\alpha) + B)f(\alpha).$$

Since f and q are coprime, we have

$$f(\alpha)|f(\alpha) - \alpha C(\alpha) - D(\alpha) \Rightarrow f(\alpha)|\alpha C(\alpha) + D(\alpha).$$

Since deg $(\alpha C(\alpha) + D(\alpha)) \leqslant n$ and deg $f(\alpha) = n+1$, it implies $\alpha C(\alpha) + D(\alpha) = 0$. Further simplifying it, we have $f(\alpha)q(\alpha) = \alpha f(\alpha)A(\alpha) + Bf(\alpha)$, which can be simplified as $q(\alpha) = \alpha A(\alpha) + B$.

Then we come to consider the items associated with $rcm\beta$. To cancel out $rcm\beta$ elements from the equation associated with $rb\beta$, any such linear combination can be written as

$$B\alpha f(\alpha) + \alpha q(\alpha)D(\alpha) = \alpha^2 f(\alpha)E(\alpha) + \alpha q(\alpha)F(\alpha),$$

which can be simplified as

$$(B - \alpha E(\alpha))f(\alpha) = (F(\alpha) - D(\alpha))q(\alpha),$$

where $E(\alpha), F(\alpha)$ are polynomials such as deg $E(\alpha) \leqslant 1$, deg $F(\alpha) \leqslant n$. By the assumption, f and q are coprime.

• If $B - \alpha E(\alpha) \neq 0$, we have $f(\alpha)|F(\alpha) - D(\alpha)$. Since deg $(F(\alpha) - D(\alpha)) \leqslant n$ and deg $f(\alpha) = n+1$, it implies

$$F(\alpha) - D(\alpha) = 0 \Rightarrow\ B - \alpha E(\alpha) = 0,$$

which contradicts the assumption since deg $B = 0$ and deg $\alpha E(\alpha) \geqslant 1$.

• Then if $B - \alpha E(\alpha) = 0$, we have $B = \alpha E(\alpha)$. Since deg $\alpha E(\alpha) \geqslant 1$, this indicates deg $B \geqslant 1$, which contradicts that B is a constant. Thus we have $B = 0, E(\alpha) = 0$. Hence we can deduct the equation $q(\alpha) = \alpha A(\alpha)$. When $\alpha = 0, \alpha A(\alpha)|_{\alpha=0} = 0$, it implies $q(\alpha)|_{\alpha=0} = 0$, which contradicts the assumption $q(0) \neq 0$.

Hence, there exist no coefficients $\{x_{i,j}\}, y_1$ such that $F = \Sigma x_{i,j} d_i d_j + \Sigma y_1 e_1$ holds, thereby (f, g, F)-MSE-DDH1 problem is intractable.

Intractability of (f, g, F)-MSE-DDH2 Problem. We assume $Q_0 = \beta P_0,\ R_0 = \gamma P_0$, where P_0, Q_0, R_0 are the generators of $\mathbb{G}$. With the same method as above, it can be also reformulated as D, E, F.

$$D = \begin{pmatrix} \alpha^3 f(\alpha)q(\alpha), & b\alpha^3 f(\alpha)q(\alpha), \\ \beta q(\alpha)Q_0, & \beta\alpha q(\alpha), \\ k_1^*\alpha q(\alpha)\beta, & k_1^* q(\alpha)\beta, \\ k_2^*\alpha q(\alpha)\beta, & k_2^* q(\alpha)\beta, \\ s_1\alpha q(\alpha)\beta, & s_1 q(\alpha)\beta, \\ s_2\alpha q(\alpha)\beta, & s_2 q(\alpha)\beta, \\ s_3\alpha q(\alpha)\beta, & s_3 q(\alpha)\beta, \\ s_4\alpha q(\alpha)\beta, & s_4 q(\alpha)\beta, \\ c\alpha^2 f(\alpha), & \cdots,\ c\alpha^n f(\alpha), \\ r\alpha q(\alpha), & \cdots,\ r\alpha^n q(\alpha), \\ rc\alpha f(\alpha), & \cdots,\ rc\alpha^{n-1} f(\alpha), \\ \alpha^2 f(\alpha), & \cdots,\ \alpha^{n+1} f(\alpha), \\ c\beta\alpha, & \cdots,\ c\beta\alpha^n, \\ s_1 c\beta\alpha, & \cdots,\ s_1 c\beta\alpha^{n-1}, \\ s_2 c\beta\alpha, & \cdots,\ s_2 c\beta\alpha^{n-1}, \\ s_3 c\beta\alpha, & \cdots,\ s_3 c\beta\alpha^{n-1}, \\ s_4 c\beta\alpha, & \cdots,\ s_4 c\beta\alpha^{n-1}, \\ \gamma\alpha, & \cdots,\ \gamma\alpha^4, \\ b\gamma\alpha^4 + s_1 c\beta\alpha^n, & b\gamma\alpha^3 + s_2 c\beta\alpha^n, \\ b\gamma\alpha^2 + s_3 c\beta\alpha^n, & b\gamma\alpha + s_4 c\beta\alpha^n \end{pmatrix},$$

$$\begin{aligned} E = {} & rb\gamma\alpha^2 f(\alpha)q(\alpha), \\ F = {} & b\gamma\alpha^4 + k_1^* c f(\alpha)\beta,\ b\gamma\alpha^3 + k_2^* c f(\alpha)\beta. \end{aligned}$$

We also need to present that F is independent of (D, E), i.e., no coefficients $\{x_i\}$, $\{y_{i,j}\}$ and z exist such that $ze_1 = \Sigma x_i f_1 d_i + \Sigma y_{i,j} d_i d_j$ and $ze_1 = \Sigma x_i f_2 d_i + \Sigma y_{i,j} d_i d_j$, where polynomials d_i, f_i are listed in D, F above respectively. We come to take the case of Z_1 into account, and the same method can be used for the case of Z_2. We divide two cases to discuss the independence of F.

1) $z \neq 0$. By making all possible products of two polynomials from D and F which are multiples of $rb\gamma$, we want to prove that no linear combination among the

polynomials from the lists F_1, F_2 below leads to F:

$$F_1 = \begin{pmatrix} rk_1^* c\beta\alpha f(\alpha)q(\alpha), \cdots, rk_1^* c\beta\alpha^n f(\alpha)q(\alpha), \\ rk_2^* c\beta\alpha f(\alpha)q(\alpha), \cdots, rk_2^* c\beta\alpha^n f(\alpha)q(\alpha) \end{pmatrix},$$

$$F_2 = \begin{pmatrix} r\alpha q(\alpha)(b\alpha^3\gamma + k_2^* c\beta f(\alpha)), \\ \vdots \\ r\alpha^n q(\alpha)(b\alpha^3\gamma + k_2^* c\beta f(\alpha)), \\ r\alpha q(\alpha)(b\alpha^4\gamma + k_1^* c\beta f(\alpha)), \\ \vdots \\ r\alpha^n q(\alpha)(b\alpha^4\gamma + k_1^* c\beta f(\alpha)) \end{pmatrix}.$$

Any such linear combination can be written as

$$\begin{aligned} & zrb\gamma\alpha^2 f(\alpha)q(\alpha) \\ = {} & r\alpha q(\alpha)C_1(\alpha)(b\alpha^3\gamma + k_2^* c\beta f(\alpha)) + \\ & rk_2^* c\beta\alpha f(\alpha)q(\alpha)D_1(\alpha) + \\ & r\alpha q(\alpha)C_2(\alpha)(b\alpha^4\gamma + k_1^* c\beta f(\alpha)) + \\ & rk_1^* c\beta\alpha f(\alpha)q(\alpha)D_2(\alpha), \end{aligned}$$

where $C_i(\alpha)$, $D_i(\alpha)$ are polynomials with deg $C \leqslant n-1$, deg $D \leqslant n-1$. Then, any such linear combination associated with $rb\gamma$ can be written as

$$\begin{aligned} & zf(\alpha)q(\alpha) = (\alpha^2 C_1(\alpha) + \alpha^3 C_2(\alpha))q(\alpha) \\ \Leftrightarrow {} & zf(\alpha) = \alpha^2 C_1(\alpha) + \alpha^3 C_2(\alpha). \end{aligned}$$

When $\alpha = 0$, $\alpha^2 C_1(\alpha) + \alpha^3 C_2(\alpha)|_{\alpha=0} = 0$, this implies $f(\alpha)|_{\alpha=0} = 0$, which contradicts $f(0) \neq 0$.

2) $z = 0$. We only consider the case of Z_2 and the same way to Z_1. By making all possible products of two polynomials from D and F which are multiples of $b\gamma$, any such linear combination can be written as

$$\begin{aligned} & (b\gamma\alpha^3 + k_2^* c\beta f(\alpha))A\alpha^3 f(\alpha)q(\alpha) + \\ & (b\gamma\alpha^3 + k_2^* c\beta f(\alpha))B(\alpha)\alpha^2 f(\alpha) \\ = {} & b\gamma\alpha^4 f(\alpha)q(\alpha)C(\alpha) + k_2^* c\beta\alpha^2 f(\alpha)q(\alpha)D(\alpha), \end{aligned}$$

where $B(\alpha)$, $C(\alpha)$, $D(\alpha)$ are polynomials and A is a constant, deg $B(\alpha) \leqslant n-1$, deg $C(\alpha) \leqslant 3$ and deg $D(\alpha) \leqslant n-1$.

To the polynomials with the coefficient $b\gamma$, we have

$$\begin{aligned} & A\alpha^6 f(\alpha)q(\alpha) + \alpha^5 f(\alpha)B(\alpha) \\ = {} & \alpha^4 f(\alpha)q(\alpha)C(\alpha). \end{aligned}$$

Therefore, we have $(A\alpha - C'(\alpha))q(\alpha) = B(\alpha)$, where deg $C'(\alpha) \leqslant 2$. Then we have $q|B(\alpha)$. Since deg $B(\alpha) \leqslant n-1$ and deg $q(\alpha) = n$, we have $B(\alpha) = 0$. Further simplifying it,

$$\begin{aligned} & (b\gamma\alpha^3 + k_2^* c\beta f(\alpha))A\alpha^3 f(\alpha)q(\alpha) \\ = {} & b\gamma\alpha^4 f(\alpha)q(\alpha)C(\alpha) + \\ & k_2^* c\beta\alpha^2 f(\alpha)q(\alpha)D_\alpha. \end{aligned}$$

To the polynomials with the coefficient $k_2^* c\beta$, we have

$$A\alpha^3 f^2(\alpha)q(\alpha) = \alpha^2 D(\alpha)f(\alpha)q(\alpha).$$

Therefore, $A\alpha f(\alpha) = D(\alpha)$. We have $f|D(\alpha)$. Since deg $D(\alpha) \leqslant n-1$ and deg $f(\alpha) = n$, we have $D(\alpha) = 0 \Rightarrow A = 0$.

Hence there exist no coefficients $\{x_i\}, \{y_{i,j}\}$ and z such that both $ze_1 = \Sigma x_i f_1 d_i + \Sigma y_{i,j} d_i d_j$ and $ze_1 = \Sigma x_i f_2 d_i + \Sigma y_{i,j} d_i d_j$ hold. Therefore (f, g, F)-MSE-DDH2 problem is intractable.