Information Sciences 465 (2018) 21–37

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Enabling verifiable multiple keywords search over encrypted cloud data

Yinbin Miao[a,b], Jian Weng[c], Ximeng Liu[d], Kim-Kwang Raymond Choo[e], Zhiquan Liu[c,*], Hongwei Li[f]

[a] Department of Cyber Engineering, Xidian University, Xi'an 710071, China
[b] Key Laboratory of Optical Communication and Networks, Chongqing 4000565, China
[c] College of Information Science and Technology, College of Cyber Security, Jinan University, Guangzhou 510632, China
[d] Department of Information Systems, Singapore Management University, 80 Stamford Road, Singapore
[e] Department of Information Systems and Cyber Security, The University of Texas at San Antonio, San Antonio, TX 78249 USA
[f] Department of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610051, China

ARTICLE INFO

Article history:
Received 16 November 2017
Revised 28 June 2018
Accepted 30 June 2018
Available online 30 June 2018

Keywords:
Searchable encryption
Certificate management
Key escrow
Ciphertexts indistinguishability
Signatures unforgeability

ABSTRACT

Searchable Encryption (SE) enables a user to search over encrypted data, such as data stored in a remote cloud server. Existing certificate-, identity-, and attribute-based SE schemes suffer from certificate management or key escrow limitations. Furthermore, the semi-honest-but-curious cloud may conduct partial search operations and return a fraction of the search results (i.e., incomplete results) in order to reduce costs. In this paper, we present a secure cryptographic primitive, Verifiable Multiple Keywords Search (VMKS) over ciphertexts, which leverages the Identity-Based Encryption (IBE) and certificateless signature techniques. The VMKS scheme allows the user to verify the correctness of search results and avoids both certificate management or key escrow limitations. We then demonstrate the security of proposed VMKS scheme (i.e., the scheme achieves both ciphertext indistinguishability and signature unforgeability). We also use a real-world dataset to evaluate its feasibility and efficiency.

1. Introduction

During the outsourcing of data (e.g. text, image, video) to a remote cloud service provider, data owners (individuals and organizations) generally encrypt their (sensitive) data in order to ensure data confidentiality [7,20,24,27,36–38,40]. In addition, some organizations may need to ensure that they are compliant with the relevant industry regulations and privacy requirements (e.g. the new European Union's General Data Protection Regulation).

Despite the benefits of encrypting the data prior to outsourcing, searching over encrypted data (and dataset) remains a challenge. Searchable encryption (SE) was designed to allow users to securely search over ciphertexts, based on pre-defined keywords, and selectively retrieve files of interest [1,25,26,29]. Examples of SE schemes include public key encryption with keyword search (PEKS), and the latter can be broadly categorize into certificate-based keyword search [3,5] and identity (or attribute)-based keyword search [32,43] schemes. In certificate-based keyword search schemes, the data owner shares

* Corresponding author.
E-mail addresses: ybmiao@xidian.edu.cn (Y. Miao), cryptjweng@gmail.com (J. Weng), xmliu@smu.edu.sg (X. Liu), raymond.choo@fulbrightmail.org (K.-K. Raymond Choo), zqliu@jnu.edu.cn (Z. Liu), hongweili@uestc.edu.cn (H. Li).

https://doi.org/10.1016/j.ins.2018.06.066
0020-0255/

his/her data by encrypting them with a specific data user's public key. The key limitation is certificate management, as one needs to verify the certificates and public keys via the certificate management system. In other words, scalability can be a challenge in practice. The key limitation with identity (or attribute)-based keyword search schemes is key escrow, since the trusted authority center can decrypt any ciphertext in the system.

A number of researchers have attempted to address such limitations. For example, the keyword search scheme presented in [42] was designed to mitigate limitations in most existing SE schemes, such as those of [12,13,16]. However, the keyword search scheme presented in [42] assumes that the cloud is honest-but-curious, in the sense that the cloud service provider will faithfully follow the established protocols but at the same time, it is curious to deduce valuable information. Such an assumption is usually insufficient in practical applications, since the cloud may be financially motivated to return incomplete search results (e.g. to minimize computation and bandwidth resources). Therefore, we consider a semi-honest-but-curious cloud [2], which executes a fraction of the requested search operations and returns incomplete search results in practice. We then provide a result verification mechanism to guarantee the accuracy of the search results by appending a signature to each file stored in a cloud. We also observe that for verifiable keyword search schemes, there is a need to support multiple-keyword search in order to minimize bandwidth resources and improve user search experience (as a single keyword search returns many irrelevant search results [21,30]).

To realize the above search functionalities simultaneously, we design a cryptographic primitive – hereafter referred to as **V**erifiable **M**ultiple **K**eywords **S**earch (VMKS). The latter allows one to perform a search over encrypted (cloud) data scheme by leveraging existing public auditing techniques, such as those presented in [33,34]. In other words, a specific data user can issue multi-keyword search and verify the search results' correctness with expressive index construction and certificateless signature. Moreover, the VMKS scheme can mitigate certificate management and key escrow limitations, and has constant trapdoor and ciphertexts retrieval sizes (both of which are important features when deploying on resource-constrained devices). We summarize the key features of the proposed VMKS scheme to be as follows:

- **Multiple keywords search**. The proposed VMKS scheme allows a specific data user to issue multiple keywords search[1] which includes conjunctive keyword search and disconjunctive keyword search in a single search query without increasing the trapdoor size[2] and ciphertexts search size, which improves the user search experience.
- **Search results verification**. The VMKS scheme allows one to verify the search results' accuracy by appending a signature to each file.
- **Certificateless**. To eliminate the certificate management and key escrow limitations in the existing SE schemes, the VMKS scheme is certificateless.

The remainder of this paper is organized as follows. Sections 2 and 3 review the literature and background relevant to the proposed VMKS scheme. Section 4 presents the system model, threat model, scheme definition and security model. The concrete construction of VMKS scheme is given in Section 5. In Section 6, we demonstrate the correctness and evaluate the security and performance of the VMKS scheme. Section 7 concludes the paper.

2. Related work

As discussed earlier, there have been extensive research on SE as evidenced by the different types of SE schemes proposed in the literature (e.g. single keyword search [17], multi-keyword search [14,15], and verifiable keyword search [21,31]).

The first ciphertext retrieval scheme designed for a symmetric setting [9,11] is first proposed by Song et al. [29], and a few years later Boneh et al. [1] presented the first PEKS scheme for a asymmetric setting [22,23]. Since then, there have been a large body of work focusing on extending the search functionalities and security of SE and related schemes. For example, Kurosawa et al. [10] demonstrated how to support document update operations (i.e. modify, delete, and add) in a verifiable way, as well as detecting cheating behavior of a malicious server. Miao et al. [22] constructed a basic cryptographic primitive, attribute-based keyword search over hierarchical data scheme, in order to support multi-keyword search and user revocation.

Conventional SE schemes only support exact keyword search, which limit system usability and impact user search experience. Thus, Li et al. [18] presented a fuzzy keyword search scheme by utilizing edit distance to handle minor typographical errors and format inconsistencies. However, in practice, SE schemes should also support multi-keyword (conjunctive or disconjunctive keyword) search to further minimize the search scope as a single keyword search often yields many irrelevant search results [4,35,39].

Zhang et al. [41] presented a ranked multi-keyword search scheme, which supports results relevance ranking and dynamic secret key generation in a multi-owner model. This reduces the need to return all search results. In a more challenging, but realistic, scenario, a malicious cloud service provider may intentionally return false search results for selfish reasons, such as to minimize costs. Thus, users should be assured of the correctness of the search results using some data verification solution [8,28]. Sun et al. [31] proposed an efficient tree-based index structure to enable an authenticity check

[1] Conjunctive keyword search means each result contains all queried keywords, while disconjunctive keyword search means that each result contains at least queried keyword. In VMKS scheme, we mainly discuss the conjunctive keyword search.

[2] In VMKS scheme, the trapdoor size is not affected by the number of queried keywords as it still contains five elements in group G_1 for each query.

Table 1
Functional comparison in various schemes .

Schemes	Function 1	Function 2	Function 3	Function 4
VABKS [43]	✓			
ABKS-UR [32]	✓	✓		
CLKS [42]			✓	✓
VCKS [30]	✓	✓		
HA-CLS [33]			✓	
Re-dtPECK [39]		✓		
VMKS	✓	✓	✓	✓

"Function1": Search result verification;
"Function2": Multi-keyword search;
"Function3": Certificateless;
"Function4": Constant trapdoor size.

over the returned search results. Miao et al. [21] also presented a verifiable conjunctive keyword search scheme in a shared multi-owner setting by utilizing the multi-signature technique.

To avoid requiring the data owner to establish a certificate management infrastructure in order to verify the data user's public key, Zheng et al. [43] presented the verifiable attribute-based keyword search scheme by utilizing attribute-based encryption (ABE) [19,20] and bloom filter. With a similar aim, Sun et al. [32] presented the attribute-based keyword search with efficient user revocation (ABKS-UR) scheme in a multi-owner setting. Although both schemes can verify the correctness of search results, the schemes do not resolve the key escrow limitation. Wang et al. [33] addressed both certificate management and key escrow limitations in their homomorphic authenticable certificateless signature (HA-CLS) scheme, but the scheme does not support ciphertext retrieval. Zheng et al. [42] presented a certificateless keyword search (CLKS) scheme, which achieves keyword search over ciphertexts, but it does not guarantee the authenticity of the search results.

To achieve all these functionalities, we present an enabling verifiable multiple keywords search over encrypted cloud data scheme by leveraging both certificateless signature scheme and public auditing appraoches [33,34]. Unlike existing SE schemes, our VMKS scheme avoids the shortcomings inherent of certificate-based keyword search schemes, as well as the false search results due to semi-honest-but-curious cloud service providers. In addition, our VMKS scheme supports multi-keyword search without a corresponding increase in the trapdoor size. We also remark that no other scheme in the literature, at the time of this research, achieves search results verification, multi-keyword search, certificateless and constant trapdoor size at the same time – see Table 1.

3. Preliminaries

In this section, we review the relevant background materials required in the understanding of the VMKS scheme.

Let $x \in_R X$ denote an element x being selected uniformly at random from the set X, [1, Y] be an integer set $\{1, 2, \ldots, \Upsilon\}$, G_1, G_2 be two multiplicative cyclic groups of prime order p, and g be a generator of group G_1. e is the bilinear map $G_1 \times G_1 \to G_2$, with the following properties: (1) Bilinearlity. $e(a^u, b^v) = e(a^v, b^u) = e(a, b)^{uv}$ for all $a,\ b \in_R G_1,\ u, v \in_R Z_p^*$; (2) Non-degeneracy. $e(g, g) \neq 1$; and (3) Computability. There is an efficient algorithm to compute $e(a, b)$ for $a, b \in_R G_1$.

The security assumptions associated with the security of our VMKS scheme are described as follows:

Definition 1 (Computational Diffie-Hellman (CDH) Assumption)**.** Let G_1 be a multiplicative cyclic group of order p, and g be a generator of G_1. Given the tuple $g, g^a, g^b \in_R G_1$ and two randomly selected elements $a, b \in_R Z_p^*$, it is computationally infeasible to compute $g^{ab} \in_R G_1$ for any probabilistic time adversary $\mathcal{A}$ with a negligible advantage ϵ, where the advantage of $\mathcal{A}$ is defined as $Pr[\mathcal{A}_{CDH}(g, g^a, g^b) = g^{ab}] < \epsilon$.

Definition 2 (Discrete Logarithm (DL) Assumption)**.** Let G_1 be a group of order p, and g be the generator of G_1. For any probabilistic polynomial time adversary $\mathcal{A}$, its advantage on solving the DL problem in group G_1 is negligible, which is defined as $Pr[\mathcal{A}(g, g^a) = a] < \epsilon$, where $a \in_R Z_p^*$.

Definition 3 (Decisional Linear (DLIN[3]) Assumption)**.** Let G_1 be a group of order p, and g be the generator of G_1. Given tuples $(g, u, v, g^c, u^b, v^{c+b})$, (g, u, v, g^c, v^b, T), the goal of probabilistic time adversary $\mathcal{A}$ is to distinguish v^{a+b} from a random element T in group G_1, where $u, v, T \in_R G_1$, $c, b \in_R Z_p^*$. Then we say DLIN assumption holds if $\mathcal{A}$'s advantage $Adv_{\mathcal{A}}^{DLIN}(k)$ in breaking DLIN problem is negligible, where the advantage $Adv_{\mathcal{A}}^{DLIN}(k)$ is defined with Eq. (1).

$$Adv_{\mathcal{A}}^{DLIN}(k) = \left| \begin{array}{l} Pr[\mathcal{A}(g, u, v, g^c, u^b, v^{c+b}) = 1] \\ -\, Pr[\mathcal{A}(g, u, v, u^a, v^b, T) = 1] \end{array} \right| < \epsilon. \tag{1}$$

[3] To distinguish from DL (Discrete Logarithm) assumption, we abbreviate the Decisional Linear assumption as DLIN assumption.

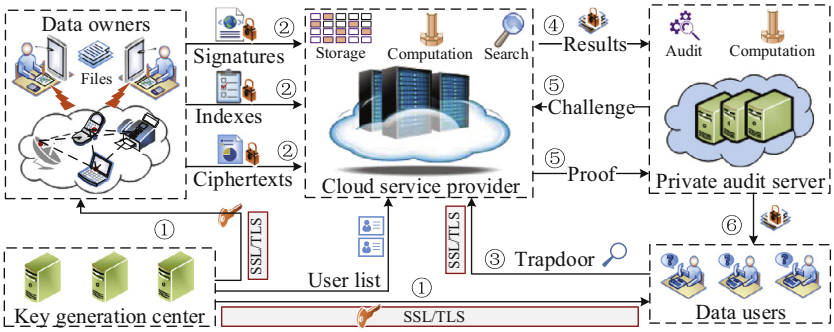

Fig. 1. System model of VMKS scheme.

4. Problem formulations

In this section, we present the system model, threat model, scheme definition and security model.

4.1. System and threat models

We consider a cloud storage system which involves five entities, namely: Data Owner (**DO**), multiple Data Users (**DUs**), Cloud Service Provider (**CSP**), Key Generation Center (**KGC**) and Private Auditing Server (**PAS**), as shown in Fig. 1.

To avoid certificate management and key escrow, the KGC generates the partial secret keys for DO and DUs. The remaining part of the secret keys will be established by the respective DO and DUs. Then, the DO builds the indexes and outsources them to CSP so that the DO can minimize local storage and computational overhead and achieve ciphertexts retrieval. The authorized DU can conduct multi-keyword search queries by submitting a trapdoor to CSP. Once the CSP has checked that the trapdoor matches with the indexes, the CSP returns the search results to the PAS who can then verify the search results' correctness. Finally, the correct search results are returned to the DU. The role of each entity is explained as follows:

- **KGC**: The KGC is responsible for generating partial private key for both DO and DU, according to their corresponding identities. The remaining part of the private keys are generated by DO and DUs themselves, as shown in Step ①.
- **DO**: The DO collects data files and generates ciphertexts, including file ciphertexts, file indexes and file signatures. These generated ciphertexts are then outsourced to the CSP – see Step ②.
- **DU**: When a DU wants to issue search queries via a secure channel, such as SSL (Secure Socket Layer) and TLS (Transport Layer Security), (s)he needs to submit a search query (multiple keywords) and generate a trapdoor (or search token), as shown in Step ③.
- **CSP**: When the CSP receives the trapdoor, it matches the trapdoor with stored indexes and returns the relevant search results to PAS, as shown in Step ④.
- **PAS**: To verify the search results' correctness, the PAS generates challenge information and sends it to CSP that needs to return the proof. Then, the PAS checks whether the proof information can pass the results verification mechanism, as shown in Step ⑤. If the search results are determined not to be forged, then the PAS sends them to DU; otherwise, the PAS sends $\perp$ to DU, as shown in Step ⑥.

Unlike prior SE schemes, we do not assume that the CSP is an honest-but-curious entity. Specifically, we assume that the CSP in the VMKS scheme is semi-honest-but-curious. That is, the CSP may be selfish to execute a fraction of the search operations and return a part of the search results to minimize costs. We also assume that the KGC is honest-but-curious, similar to the assumption in the CLKS scheme [42]. A DU may also impersonate another DU in order to learn valuable information about the target DU, but the impersonator cannot collude with the KGC. We also assume that the PAS is fully-trusted, but the restriction is that the PAS cannot collude with the CSP.

4.2. Overview of VMKS scheme

The VMKS scheme is a tuple of algorithms: **Setup, ParKeyGen, KeyGen, Enc, Trap, Search** and **Verify**, – see Fig. 2.

4.3. Security model

Similar to prior certificateless schemes [6,42], we consider two kinds of adversaries, namely: $\mathcal{A}_1$ and $\mathcal{A}_2$.

- $\mathcal{A}_1$ is an outside attacker who is allowed to control DUs' public keys, with the exception of the master key *msk*.
- $\mathcal{A}_2$ is an internal attacker who is able to access the master key *msk*, without controlling DUs' public keys.

The VMKS scheme definition

The VMKS scheme is a tuple of seven algorithms which are shown as follows:

- **Setup**$(1^k) \rightarrow \{pk, msk, PK, SK\}$: Given a security parameter k, the KGC outputs the public key pk and master key msk, and returns the public/secret key pair (PK, SK) of RSA algorithm.
- **ParKeyGen**$(pk, msk, ID_u, ID_o) \rightarrow \{psk_u, psk_o\}$: Given the identities (ID_u, ID_o), the KGC runs this algorithm to return partial private keys (psk_u, psk_o) for DU and DO, respectively.
- **KeyGen**$(pk, ID_u, ID_o) \rightarrow \{(pk_u, sk_u), (pk_o, sk_o)\}$: On input (ID_u, ID_o), DU and DO output final public/secret key pairs (pk_u, sk_u), (pk_o, sk_o), respectively.
- **Enc**$(F, W, pk, PK, ID_u, ID_o, pk_o, sk_o, pk_u) \rightarrow \{I, C, \delta\}$: Given the file set F and keyword dictionary W, the DO runs this algorithm to output the ciphertext set C, index set I and signature set δ. Then, the DO sends the tuple (I, C, δ) to CSP. In addition, the DO sends ID_o to PAS so as to check the search results' correctness.
- **Trap**$(sk_u, pk_u, W', pk, ID_u) \rightarrow T_{W'}$: On input the search query W', the relevant DU runs this algorithm to generate the search token $T_{W'}$ and sends it to CSP.
- **Search**$(pk, T_{W'}, I, C) \rightarrow (C', id')$: After gaining the trapdoor $T_{W'}$, the CSP attempts to match it with the index I and outputs the search results C' as well as the corresponding identity set id'. Then, it sends the tuple (C', id') to PAS for verification of search results' accuracy.
- **Verify**$(C', ID_o, pk, id') \rightarrow \{0, 1\}$: Upon receiving the search results C', the PAS calls the challenge-response mode by interacting with CSP. If C' passes the result verification mechanism, then the PAS outputs "1"; otherwise, "0" is returned.

Fig. 2. Overview of the VMKS scheme.

We will show that the VMKS scheme is secure against both adversities using the following two security games. Let $\mathcal{A}_1$ be the adversary in **Game 1**, $\mathcal{A}_2$ be the adversary in **Game 2**, and $\mathcal{C}$ be the challenger who maintains the following two lists:

- **TokenList**: For the tuple (ID_u, w), the search token associated with the specific keyword w has been queried by $(\mathcal{A}_1, \mathcal{A}_2)$.
- **UserInfoList**: Given the tuple (ID_u, psk_u, sk_u, pk_u, ϱ_1, ϱ_2, ϱ_3), we assume that psk_u has been queried by $(\mathcal{A}_1, \mathcal{A}_2)$ when $\varrho_1 = 1$, but it has not been queried when $\varrho_1 = 0$. Similarly, $\varrho_2 = 1, \varrho_3 = 1$ means that sk_u, pk_u have been queried by $(\mathcal{A}_1, \mathcal{A}_2)$, respectively; otherwise, sk_u, pk_u have not been queried by these adversaries.

To be specific, the **Game 1** (or **Game 2**) is played between $\mathcal{A}_1$ (or $\mathcal{A}_2$) and $\mathcal{C}$. Due to page limitations, we will restrict our discussion to **Game 1**, and refer readers interested in **Game 2** (similar to **Game 1**, except that it does not has **Phase 1**) to [6,42].

Setup: $\mathcal{C}$ runs **Setup** to output the public parameters pk and master key msk, sends pk to $\mathcal{A}_1$, and sets the lists (**TokenList** and **UserInfoList**) null.

Phase 1: Assume that $\mathcal{A}_1$ can polynomially query the following oracles, then $\mathcal{C}$ can conduct the following algorithms:

- *ParKeyGen*. Given the user identity ID from $\mathcal{A}_1$, $\mathcal{C}$ returns psk_{ID} on condition that psk_{ID} in **UserInfoList** is not empty; otherwise, $\mathcal{C}$ runs the *ParKeyGen* algorithm to gain psk_{ID} by utilizing msk, adds the tuple (ID, psk_{ID}, $*$, $*$, 1, 0, 0) to **UserInfoList**, and sends psk_{ID} to $\mathcal{A}_1$, where the symbol "$*$" denotes null.
- *KeyGen*. Given ID from $\mathcal{A}_1$, $\mathcal{C}$ selects sk_{ID} from **UserInfoList** if sk_{ID} is not null; If psk_{ID} in **UserInfoList** is not null, then $\mathcal{C}$ calls the *ParKeyGen* algorithm to generate sk_{ID} by leveraging psk_{ID}, and $\mathcal{C}$ adds sk_{ID} in **UserInfoList**. If the above two conditions do not hold, then $\mathcal{C}$ runs both *ParKeyGen* and *KeyGen* algorithms to gain psk_{ID} and sk_{ID}, respectively, and then $\mathcal{C}$ adds the tuple (ID, psk_{ID}, sk_{ID}) in **UserInfoList**. Finally, $\mathcal{C}$ updates $\varrho_1 = 1$, $\varrho_2 = 1$ in **UserInfoList** associated with user identity ID and sends sk_{ID} to $\mathcal{A}_1$.
  If pk_{ID} in **UserInfoList** is not null, then $\mathcal{C}$ selects pk_{ID}; if sk_{ID} in **UserInfoList** is not null, then $\mathcal{C}$ selects sk_{ID} and executes *KeyGen* to output pk_{ID} which will be added into **UserInfoList** associated with ID; if psk_{ID} in **UserInfoList** is not full, then $\mathcal{C}$ runs the *KeyGen* algorithm to generate sk_{ID}, pk_{ID} by using psk_{ID}. Note that $\mathcal{C}$ still adds the tuple (sk_{ID}, pk_{ID}) into **UserInfoList**. Otherwise, $\mathcal{C}$ executes both *ParKeyGen* and *KeyGen* algorithms to generate psk_{ID}, sk_{ID}, pk_{ID}, and adds the tuple (ID, psk_{ID}, sk_{ID}, pk_{ID}, 0, 0, 0) into **UserInfoList**. Finally, $\mathcal{C}$ sends pk_{ID} to $\mathcal{A}_1$.
- *Replace-KeyGen*: Given ID and the replaced public key pk given by $\mathcal{A}_1$, $\mathcal{C}$ updates the element pk_{ID} in **UserInfoList** with pk. Note that pk_{ID} has been queried by $\mathcal{A}_1$. Finally, $\mathcal{C}$ sets $\varrho_3 = 1$.
- *Trap*: On input ID and queried keyword set W' submitted by $\mathcal{A}_1$, $\mathcal{C}$ selects the secret key sk_{ID} in **UserInfoList**, performs the *Trap* algorithm to generate the trapdoor, adds the tuple (ID, W') into **TokenList**, and sends the trapdoor to $\mathcal{A}_1$.

Challenge phase: $\mathcal{A}_1$ submits two different keyword sets W_0, W_1 with same length and user identity ID^*. Given (ID^*, psk_{ID^*}, sk_{ID^*}, pk_{ID^*}, $\varrho_1, \varrho_2, \varrho_3$) in **UserInfoList**, we still require that psk_{ID^*}, sk_{ID^*} cannot be queried by $\mathcal{A}_1$ if the con-

Table 2
Notation descriptions .

Notations	Descriptions
$\{PK, SK\}$	Public/secret keys of RSA algorithm
psk_u, psk_o	Partial private key for DU and DO
$\{pk_u, sk_u\}$	Public/private key pair of DU
$\{pk_o, sk_o\}$	Public/private key pair of DO
$F = \{f_i\}_{i\in[1,d]}$	File set
$C = \{c_i\}_{i\in[1,d]}$	Ciphertext set for F
$\delta = \{\delta_i\}_{i\in[1,d]}$	Signature set for F
$id = \{id_i\}_{i\in[1,d]}$	Identity set for F
$W = \{w_j\}_{j\in[1,m]}$	Keyword dictionary
$I = \{I_i\}_{i\in[1,d]}$	Index set for F
$W' = \{w'_\tau\}_{\tau\in[1,l]}$	Queried keyword set
$T_{W'} = (T_1, T_2, T_3, T_4, T_5)$	Trapdoor (or search token) for W'
$L = \{\rho_1(1), \cdots, \rho_1(l)\}$	Location set of W' in W
$C' = \{c'_r\}_{r\in[1,q]}$	Search results
$id' = \{id'_r\}_{r\in[1,q]}$	Corresponding identity set for C'
$\{r, \xi_r\}_{r\in[1,q]}$	Challenging information
$\{\theta, \delta^*\}$	Proof information

dition $\varrho_1 = 0$, $\varrho_2 = 0$ holds. Besides, both (ID^*, W_0), (ID^*, W_1) are not stored in **TokenList**. If $\varrho_3 = 0$, then the pk_{ID^*} can be replaced; otherwise, pk_{ID^*} cannot be replaced. $\mathcal{C}$ selects a random bit $\kappa \in 0, 1$ and encrypts the W_κ by running the *Enc* algorithm and returns the ciphertexts I^* to $\mathcal{A}_1$.

Phase 2: This phase is similar to **Phase 1**, the only restriction is that $\mathcal{A}_1$ cannot issue the following queries: $ParKeyGen(ID^*)$, $KeyGen(ID^*)$, $Trap(ID^*, W_0)$ or $Trap(ID^*, W_1)$.

Guess: $\mathcal{A}_1$ returns a guess bit κ' and wins the above game on condition that the equation $\kappa' = \kappa$ holds.

Definition 4. If for any probabilistic polynomial time algorithm $\mathcal{A}_1$, it wins the **Game 1** with a negligible advantage $|Pr[\kappa = \kappa'] - \frac{1}{2}|$, then the VMKS scheme achieves ciphertexts indistinguishability against $\mathcal{A}_1$.

The VMKS scheme also provides a homomorphic certificateless verification mechanism to guarantee the search results' correctness in a semi-honest-but-curious cloud computing environment. To prove that it is computationally infeasible to forge valid signatures in the VMKS scheme, we consider two types of adversaries with different attack capabilities, namely **Type-I adversary** $\mathcal{A}'_1$ and **Type-II adversary** $\mathcal{A}'_2$.

- **Type-I adversary**. Although $\mathcal{A}'_1$ cannot access the master key of the KGC, $\mathcal{A}'_1$ can replace the public key of each entity with a random element as the certificateless signature scheme avoids certificate management.
- **Type-II adversary**. In contrast to $\mathcal{A}'_1$, $\mathcal{A}'_2$ can access the master key generated by the KGC, whereas $\mathcal{A}'_2$ does not have the ability to replace the public key of any entity with a selected element because the certificateless signature scheme avoids the key escrow.

5. Construction of VMKS scheme

Based on the certificateless keyword search scheme [42], we aim to achieve a more practical SE scheme supporting both results verification and multi-keyword search. Specifically, the VMKS scheme should verify the search results' validity and allow the DU to issue conjunctive keyword search. As for the certificateless results verification, the VMKS scheme appends a signature to each file and then verifies the search results' correctness. Furthermore, the scheme needs to avoid both certificate management and key escrow.

We first define two mapping functions, namely: $\rho_1(\cdot)$ maps the keyword locations in W' to the corresponding locations in W, $\rho_2(\cdot)$ maps the ciphertext locations in C' to the corresponding locations in C. Then, we will present the summary of notations used in the VMKS scheme in Table 2. The specific construction of the VMKS scheme includes system initialization, key generation, ciphertexts generation, trapdoor generation, ciphertexts retrieval and results verification.

5.1. System initialization

The KGC runs **Setup** to return the global public parameters pk and master key msk owned by itself, and it also outputs the public/secret key pair (PK, SK).

- **Setup**(1^k): Given a security parameter k, the KGC generates the global public parameters $\mathcal{GP} = (G_1, G_2, e, p, g)$. Then, it selects $x, y, z \in_R Z_p^*$, $g_0, u, u_1, \cdots, u_n \in_R G_1$, and computes g^x, g^y, g^z. Finally, it chooses two anti-collision hash functions $h : \{0,1\}^* \to_R Z_p^*$, H_0: $\{0,1\}^* \to_R G_1$ and defines a hash function $H_1(ID) = u\prod_{k=1}^{n} u_k^{ID_k}$, where $ID = \{ID_1, \cdots, ID_n\}$. In addition, it generates the public/secret key pair (PK, SK). The global public parameters pk and master key msk are defined by

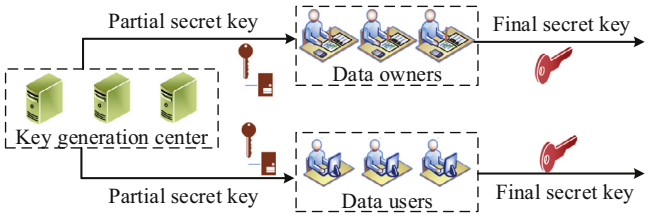

Fig. 3. Key generation process in VMKS scheme.

following equation, where *msk* is owned by KGC, *SK* is shared among authorized cloud clients (i.e., DO, DU, etc.).

$$\begin{aligned} pk &= \{g, h, H_0, H_1, g_0, u, u_1, \cdots, u_n, g^x, g^y, g^z\}; \\ msk &= \{x, y, z\}. \end{aligned} \tag{2}$$

5.2. Key generation

Note that key generation in the VMKS scheme is divided into two steps, that is the KGC generates the partial secret key for each entity (i.e., DO, DU, etc.) in **ParKeyGen** and the entity generates the final secret key by itself in **KeyGen**, – see Fig. 3. Thus, the VMKS scheme enjoys the characteristics of identity (or attribute)-based keyword search schemes (e.g. no certificate management, etc.) and avoids key escrow.

- **ParKeyGen**(pk, msk, ID_u, ID_o): Given the identity ID_u of a specific DU, the KGC selects an element $a_u \in_R Z_p^*$ and computes $psk_{u,1} = g^{a_u}$, $psk_{u,2} = g^{xz} H_1(ID_u)^{a_u}$. Then, the KGS sets the DU's partial private key as $psk_u = \{psk_{u,1}, psk_{u,2}\}$. The DO's partial private key is set as $psk_o = \{g^{a_o}, g^{xz} H_1(ID_o)^{a_o}\}$ in the same way.
- **KeyGen**(pk, ID_u, ID_o): The DU selects two elements $s_u, t_u \in_R Z_p^*$ and computes g^{s_u}, g^{t_u}. Then, the public/secret key pair $\{pk_u, sk_u\}$ of the DU is set as $pk_u = \{g^{s_u}, g^{t_u}\}$, $sk_u = \{psk_u, s_u, t_u\}$. Similarly for the DO's public/private key pair denoted as $pk_o = \{g^{s_o}, g^{t_o}\}$, $sk_o = \{psk_o, s_o, t_o\}$.

5.3. Ciphertexts generation

The resource-limited DO needs to build index and generate signature for each file before outsourcing them to CSP. Note that the expressive indexes enable the DUs to issue multi-keyword search without incurring bandwidth and computation costs. Furthermore, appending a signature to each file allows the PAS to check the search results' correctness. Although **Enc** incurs computational burden on the DO, this one-time cost does not affect the user's search experience.

- **Enc**(F, W, pk, PK, pk_o, sk_o, pk_u, ID_o, ID_u): For each file $f_i \in F = \{f_1, \cdots, f_d\}$ with identity id_i, the DO first encrypts it as c_i with PK (the public key of a conventional public encryption algorithm, such as RSA) rather than the public key pk_u of each DU and computes $\varphi_i = H_0(ID_o || g^{s_o} || id_i) \cdot g_0^{h(c_i)}$. Then the DO returns the signature $\delta_i = \varphi_i^{s_o} \cdot H_1(ID_o)^{s_o}$.
  In addition, the DO extracts the keyword set $W_i \subseteq W = \{w_1, \cdots, w_m\}$ from f_i and builds index I_i for this file. The DO selects two elements $b, c \in_R Z_p^*$ and computes the tuple $(\lambda_1, \lambda_2, \lambda_3, \lambda_4)$, where $\lambda_1 = g^{(z+t_u)b}$, $\lambda_2 = g^{(x+s_u)(b+c)}$, $\lambda_3 = g^c$, $\lambda_4 = H_1(ID_u)^c$. If the keyword $w_j \in W (1 \le j \le m)$ is included in f_i, then the DO sets $\mu_{i,j} = g^{yh(w_j)b}$; otherwise, $\mu_{i,j} = 1$.
  Then, I_i is denoted as $I_i = (\lambda_1, \lambda_2, \lambda_3, \lambda_4, \{\mu_{i,j}\})$. Finally, the DO sends the index set $I = \{I_1, \cdots, I_d\}$, ciphertexts $C = \{c_1, \cdots, c_d\}$, signatures $\delta = \{\delta_1, \cdots, \delta_d\}$, and corresponding identities $id = \{id_1, \cdots, id_d\}$ to CSP, and the identity ID_o is returned to PAS so that the latter can verify the search results' accuracy. The **Enc** is shown in Algorithm 1 .

5.4. Trapdoor generation

In **Trap**, the DUs can submit multiple keywords in a single search query without increasing the trapdoor size. This is an important feature for deploying resource-limited devices. Besides, the search query is encrypted before being sending to the CSP; thus the semi-honest-but-curious CSP will not learn the query plaintext information.

- **Trap**(sk_u, pk_u, W', pk, ID_u): Given the queried keyword set $W' = \{w'_1, \cdots, w'_l\} \subseteq W$, the DU selects an element $v \in_R Z_p^*$ and computes the trapdoor $T_{W'} = (T_1, T_2, T_3, T_4, T_5)$, where $T_1 = g^{(x+s_u)v}$, $T_2 = g^{a_u v}$, $T_3 = g^{(z+t_u)v}$, $T_4 = g^{(x+s_u)(z+t_u)v} H_1(ID_u)^{a_u v}$, $T_5 = \prod_{\tau=1}^{l} g^{yh(w'_\tau)v}$. Then, the DO sends $T_{W'}$ and the corresponding location set L (i.e., the locations of W' in W) to CSP, where $L = \{\rho_1(1), \cdots, \rho_1(l)\}$. The specific trapdoor generation process is shown in Algorithm 2 .

5.5. Ciphertexts retrieval

In **Search**, the CSP can provide ciphertexts retrieval services according to a certain DU's trapdoor. The CSP first checks whether the trapdoor matches with indexes. If yes, then it returns the search results of interest; otherwise, it

Algorithm 1: Ciphertexts generation.

Input: Public parameters PK, pk, pk_o, pk_u, secret keys sk_o, identities ID_o, ID_u, files F and keywords W.
Output: File ciphertexts C, indexes I and signatures δ

Given $F = \{f_1, \cdots, f_d\}$, $W = \{w_1, \cdots, w_m\}$;
for $1 \le i \le d$ **do**
  Encrypt f_i as c_i with PK ;
  Generate signature $\delta_i = \varphi_i^{s_o} \cdot H_1(ID_o)^{s_o}$;
  Choose elements $b, c \in_R Z_p^*$;
  Compute $\lambda_1 = g^{(z+t_u)b}$, $\lambda_2 = g^{(x+s_u)(b+c)}$, $\lambda_3 = g^c$, $\lambda_4 = H_1(ID_u)^c$;
  for $1 \le j \le m$ **do**
    if $w_j \in f_i$ **then**
      Set $\mu_{i,j} = g^{yh(w_j)b}$;
    else
      Set $\mu_{i,j} = 1$;
  Set $I_i = (\lambda_1, \lambda_2, \lambda_3, \lambda_4, \{\mu_{i,j}\})$;
Return $C = \{c_1, \cdots, c_d\}$, $\delta = \{\delta_1, \cdots, \delta_d\}$, $I = \{I_1, \cdots, I_d\}$.

Algorithm 2: Trapdoor generation.

Input: Public parameters pk, pk_u, secret keys sk_u, identities ID_u and queried keywords W'.
Output: Trapdoor $T_{W'}$, the location set L.

Given $W' = \{w'_1, \cdots, w'_l\} \subseteq W$;
for $1 \le \tau \le l$ **do**
  Select $v \in_R Z_p^*$;
  Compute $T_1 = g^{(x+s_u)v}$, $T_2 = g^{a_u v}$, $T_3 = g^{(z+t_u)v}$, $T_4 = g^{(x+s_u)(z+t_u)v} H_1(ID_u)^{a_u v}$, $T_5 = \prod_{\tau=1}^{l} g^{yh(w'_\tau)v}$;
Return $T_{W'} = (T_1, T_2, T_3, T_4, T_5)$, $L = \{\rho_1(1), \cdots, \rho_1(l)\}$.

returns $\perp$. Compared with the single keyword search, the VMKS scheme can quickly locate the relevant results without the need to also return irrelevant ones. This results in bandwidth and computation savings. More importantly, the storage and computational costs of ciphertexts retrieval are constant, which are not affected by the number of queried keywords.

- **Search**$(pk, T_{W'}, I, L, C)$: After getting the trapdoor $T_{W'}$, the CSP computes the tuple $(\sigma_1, \sigma_2, \sigma_3)$, where $\sigma_1 = e(\lambda_2 \prod_{\tau=1}^{l} \mu_{i,\rho_1(\tau)}, T_3)$, $\sigma_2 = e(\lambda_1, T_1 T_5)$, $\sigma_3 = e(\lambda_3, T_4)/e(\lambda_4, T_2)$. Then, the CSP matches the trapdoor $T_{W'}$ with the index set I using Eq. 3. If Eq. 3 holds, then the CSP returns the relevant ciphertexts $C' = \{c'_1, \ldots, c'_q\} \subseteq C$ and its corresponding identity set $id' = \{id'_1, \cdots, id'_q\}$ to PAS; otherwise, the CSP returns $\perp$. The specific ciphertexts retrieval process is shown in Algorithm 3 .

Algorithm 3: Ciphertexts retrieval.

Input: Global public parameters pk, trapdoor $T_{W'}$, indexes I, ciphertexts C and queried keyword locations L.
Output: Search results C' and corresponding file identities id'.

Given $T_{W'} = (T_1, T_2, T_3, T_4, T_5)$, $C = \{c_1, \cdots, c_d\}$, $I = \{I_1, \cdots, I_d\}$, $L = \{\rho_1(1), \cdots, \rho_1(l)\}$;
for $1 \le i \le d$ **do**
  for $1 \le \tau \le l$ **do**
    Set $Init_0 = 1$;
    Find $\mu_{i,\rho_1(\tau)}$;
    Compute $Init_\tau = \mu_{i,\rho_1(\tau)} \cdot Init_{\tau-1}$;
  Compute $\sigma_1 = e(\lambda_2 \cdot Init_l, T_3)$, $\sigma_2 = e(\lambda_1, T_1 T_5)$, $\sigma_3 = e(\lambda_3, T_4)/e(\lambda_4, T_2)$;
  Check $\sigma_1 \stackrel{?}{=} \sigma_2 \cdot \sigma_3$ (a) ;
  if *Eq. (a) holds* **then**
    $W' \in f_i$ and pick c_i ;
  else
    $W' \notin f_i$ and abort c_i ;
Return $C' = \{c'_1, \ldots, c'_q\} \subseteq C$, $id' = \{id'_1, \cdots, id'_q\}$.

$$\sigma_1 = \sigma_2 \cdot \sigma_3. \tag{3}$$

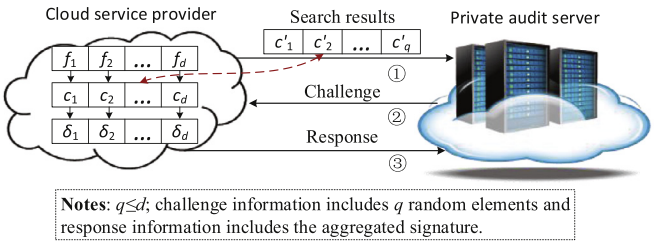

Fig. 4. Results verification process in VMKS scheme.

5.6. Results verification

Since we assume that the search results are returned by some semi-trusted third-parties, the PAS should furnish a result verification mechanism in **Verify** to verify the search results' correctness, according to the challenge-response mode between PAS and CSP, – see Fig. 4. Note that the red dotted line in Fig. 4 denotes the same ciphertext.

- **Verify**(C', ID_o, pk, id'): Upon receiving the search results C', the PAS needs to verity the correctness of C'. The PAS selects an element $\xi_r \in_R Z_p^*$ for each ciphertext $c'_r (r \in [1, q])$. Then, the PAS sends the challenging information $\{r, \xi_r\}_{r\in[1,q]}$ to CSP. Finally, the CSP responds as follows:
  - Computes $\theta = \sum_{r=1}^{q} \xi_r h(c_{\rho_2(r)})$.
  - Outputs the aggregated signature $\delta^* = \prod_{r=1}^{q} \delta_{\rho_2(r)}^{\xi_r}$.
  - Returns the proof information $proof = (\theta, \delta^*)$ to PAS.

  After getting $proof$, the PAS computes $\gamma_o = H_1(ID_o)$, $\beta_r = H_0(ID_o||g^{s_o}||id_{\rho_2(r)})$, and checks whether C' is correct or not according to Eq. (4). If Eq. (4) holds, then the PAS accepts these results C' and returns them to the specific DU; otherwise, it declines. The results verification mechanism is presented in Algorithm 4 .

Algorithm 4: Result verification.

Input: Global public parameters pk, identity ID_o, search results C' and corresponding file identities id'.
Output: "Yes" or "No".

Given $C' = \{c'_1, \ldots, c'_q\}$, $id' = \{id'_1, \cdots, id'_q\}$;
for $1 \le r \le q$ **do**
  $\xi_r \in_R Z_p^*$;
Send $\{r, \xi_r\}_{r\in[1,q]}$ to CSP ;
for $1 \le r \le q$ **do**
  Set $\theta_0 = 0$, $\delta_0^* = 1$;
  Compute $\theta_r = \theta_0 + \xi_r h(c_{\rho_2(r)})$, $\delta_r^* = \delta_0^* \cdot \delta_{\rho_2(r)}^{\xi_r}$;
Set $\theta = \theta_q$, $\delta^* = \delta_q^*$;
Send $proof = (\theta, \delta^*)$ to PAS ;
Compute $\gamma_o = H_1(ID_o)$, $\beta_r = H_0(ID_o||g^{s_o}||id_{\rho_2(r)})$;
Check $e(\delta^*, g) \stackrel{?}{=} e(\prod_{r=1}^{q}(\gamma_o \cdot \beta_r)^{\xi_r}, g^{s_o}) e(g_0^{\theta}, g^{s_o})$ (b) ;
if *Eq. (b) holds* **then**
  Send C' to DU ;
  Return "Yes" ;
else
  Abort C' ;
  Return "No".

$$e(\delta^*, g) = e\left(\prod_{r=1}^{q}(\gamma_o \cdot \beta_r)^{\xi_r}, g^{s_o}\right) e(g_0^{\theta}, g^{s_o}). \tag{4}$$

Remarks. The VMKS scheme not only allows a trusted third-party to verify the search results' correctness but also provides a certificateless SE scheme to avoid certificate management and key escrow. Furthermore, the VMKS scheme enables the DUs to issue conjunctive keyword search with constant trapdoor size and ciphertexts retrieval size. Note that in **Verify**, the PAS only needs to select q random elements and compute the tuple $(\gamma_o, \{\beta_r\})$. Thus, we assume that the PAS is honest-but-curious rather than fully-trusted. Thus, the deployment of the VMKS system is flexible and can be customized based on user requirements.

6. Analysis of VMKS scheme

In this section, we analyze the correctness, security and performance of the VMKS scheme.

6.1. Correctness

To demonstrate the VMKS scheme's correctness, we should guarantee that the CSP honestly performs cipertexts retrieval operations without forging incorrect search results by checking whether Eqs. (3) and (4) hold.

For Eq. (3), we first have

$$\sigma_1 = e(g,g)^{(x+s_u)(z+t_u)(b+c)v+(z+t_u)bv\sum_{\tau=1}^{l} yh(w_{\rho_1(\tau)})};$$

$$\sigma_2 = e(g,g)^{(x+s_u)(z+t_u)bv+(z+t_u)bv\sum_{\tau=1}^{l} yh(w'_{\tau})};$$

$$\sigma_3 = \frac{e(g^c, H_1(ID_u)^{a_u v})e(g,g)^{(x+s_u)(z+t_u)cv}}{e(H_1(ID_u)^c, g^{a_u v})} = e(g,g)^{(x+s_u)(z+t_u)cv}.$$

If $W' \subseteq W$, then we have $\sigma_1 = \sigma_2 \cdot \sigma_3$. Thus, we verify that Eq. (3) holds.

For Eq. (4), we first have

$$e(\delta^*, g) = e\left(\prod_{r=1}^{q}(\varphi_{\rho_2(r)}^{s_o} \cdot H_1(ID_o)^{s_o})^{\xi_r}, g\right) = e\left(\prod_{r=1}^{q}(\gamma_o \cdot \beta_r)^{\xi_r}, g^{s_o}\right) \cdot e\left(\prod_{r=1}^{q} g_0^{\xi_r h(c_{\rho_2(r)})}, g^{s_o}\right) = e\left(\prod_{r=1}^{q}(\gamma_o \cdot \beta_r)^{\xi_r}, g^{s_o}\right) e(g_0^{\theta}, g^{s_o}),$$

then we can also check that Eq. (4) holds. Thus, the VMKS scheme's correctness is obtained.

6.2. Security analysis

Superior to the certificate-based schemes, the VMKS scheme allows the DU to put less trust on the KGC and avoids certificate management. In addition, the VMKS scheme avoids key escrow in the identity (or attribute)-based keyword search schemes. More importantly, the VMKS scheme achieves ciphertexts indistinguishability and signatures unforgeability. With regard to the honest-but-curious KGC in the VMKS scheme, the outside and inside adversaries should not break the ciphertexts indistinguishability with a non-negligible advantage under the DLIN assumption. As the threat model of the CLKS scheme [42] also assumes that the KGC is hones-but-curious in Game 1 and Game 2, then the VMKS scheme can achieve the aforementioned security goal by utilizing the same security games. Besides, the semi-honest-but-curious CSP cannot forge valid proof information under CDH and DL assumptions. The aforementioned security goals in the VMKS scheme can be guaranteed by the following theorems.

Theorem 1. *If $\mathcal{A}_1$ makes at most q_1 queries to **ParKeyGen** oracle and q_2 queries to **KeyGen** oracle in **Game 1**, and Adv_h denotes the $\mathcal{A}_1$'s advantage in breaking the collision-resistant hash function h, and Adv_{DLIN} represents the $\mathcal{A}_1$'s advantage in breaking the DLIN assumption, then the $\mathcal{A}_1$'s advantage in breaking the ciphertexts indistinguishability is $Adv_{\mathcal{A}_1} \leq Adv_h + 2(q_1+q_2)(n+1)Adv_{DLIN}$, where n denotes the bit length of ID_u.*

Proof. Let E_i be the event in which the $\mathcal{A}_1$ wins $game_i$, and Adv_i be the $\mathcal{A}_1$'s advantage in breaking $Game_i$. In addition, the process of $Game_{i+1}$ is the same as that of $Game_i$ except for the case that $Game_{i+1}$ terminates and returns a random bit when the pre-defined E_p occurs. If E_p is detectable and independent from E_i, then we have $Pr[E_{i+1}] = Pr[E_{i+1}|E_p]Pr[E_p] + Pr[E_{i+1}|\neg E_p]Pr[\neg E_p] = \frac{1}{2}(1 - Pr[\neg E_p]) + Pr[E_i]Pr[\neg E_p]$, where the symbol "¬" denotes "NOT" operation.

Thus, we can get $|Pr[E_{i+1}] - \frac{1}{2}| = Pr[\neg E_p]|Pr[E_i] - \frac{1}{2}|$, $Adv_{i+1} = Pr[\neg E_p]Adv_i$. Next, we present a series of security games as follows:

$Game_1$: $\mathcal{A}_1$ proceeds with the same steps as shown in **Game 1**, and $\mathcal{C}$ outputs the (msk, pk, psk_u). Given the identity ID_{u^*} of the DU and his corresponding public key pk_{u^*}, $\mathcal{C}$ generates the ciphertexts $I_i^* = (\lambda_1^*, \lambda_2^*, \lambda_3^*, \lambda_4^*, \{\mu_{i,j}^*\})$ and returns them to $\mathcal{A}_1$.

$Game_2$: $\mathcal{A}_1$ proceeds with the same steps as shown in $Game_1$ except that h is considered to be a perfect collision-resistant function. Thus, we have $Pr[E_2] = Pr[E_1] - Adv_h$, $Adv_2 = Adv_1 - Adv_h$.

$Game_3$: $\mathcal{C}$ issues this game as the same as $Game_2$ except that the generation of pk, according to the following steps:

- Chooses $x, y, z \in_R Z_p^*$, and selects $\alpha_u \in \{0, 1, \cdots, n\}$ and $\beta_u \in \{0, 1, \cdots, p\}$ such that $\beta_u(n+1) < p$.

- Chooses $\gamma'_u \in_R Z_{\beta_u}$ and a tuple $\{\gamma_{u,1}, \cdots, \gamma_{u,n}\} \in_R Z^n_{\beta_u}$.
- Chooses $\vartheta'_u \in_R Z_p$ and a tuple $\{\vartheta_{u,1}, \cdots, \vartheta_{u,n}\} \in_R Z^n_p$.
- The remaining part of global public parameters pk is defined by Eq. 5, where $k \in [1, n]$.

$$u = (g^x)^{\gamma'_u - \alpha_u \beta_u} g^{\vartheta'_u}; \; u_k = (g^x)^{\gamma_{u,k}} g^{\vartheta_{u,k}}. \tag{5}$$

Thus, we have $Pr[E_3] = Pr[E_2]$, $Adv_3 = Adv_2$.

Game$_4$: $\mathcal{C}$ conducts this game as the same as *Game*$_3$ apart from the guess phase. $\mathcal{C}$ gives two functions $A_u(\cdot)$, $B_u(\cdot)$ for an identity $ID = \{ID_1, \cdots, ID_n\}$ through Eq. (6).

$$\begin{aligned} A_u(ID) &= \gamma'_u - \alpha_u \beta_u + \sum_{k=1}^{n} ID_k \gamma_{u,k}; \\ B_u(ID) &= \vartheta'_u + \sum_{k=1}^{n} ID_k \vartheta_{u,k}. \end{aligned} \tag{6}$$

In the guess phase, $\mathcal{C}$ checks $A_u(ID^*) \stackrel{?}{=} 0$. If $A_u(ID^*) = 0$, then $\mathcal{C}$ terminates and returns a random bit $b' \in \{0, 1\}$ as $\mathcal{A}_1$'s guess; otherwise, $\mathcal{C}$ has the same procedures as those of *game*$_3$. As the tuple $\{\gamma'_u, \gamma_{u,1}, \cdots, \gamma_{u,n}\} \in_R Z^n_{\beta_u}$ is unknown to $\mathcal{A}_1$ and the equation $Pr[A_u(ID^*) = 0 \bmod p] = \frac{1}{\beta_u(n+1)}$ holds, we can get $Adv_4 = \frac{Adv_3}{\beta_u(n+1)}$.

Game$_5$: $\mathcal{C}$ conducts this game as the same as *Game*$_4$ apart from the guess phase. In the guess phase, $\mathcal{C}$ checks whether one of the following situations occurs:

- For the identity ID, $A_u(ID) = 0$ and it is queried to the **ParKeyGen** oracle.
- For the identity ID, $A_u(ID) = 0$ and it is queried to the **KeyGen** oracle.

If one of above situations happens, $\mathcal{C}$ terminates and returns a random bit $b' \in \{0, 1\}$ as $\mathcal{A}_1$'s guess. As the guess phase in *Game*$_5$ might not be independent of *game*$_4$, we can have the lower bound of $Pr[\neg E_p] = Pr[(\bigcap_{ID \in S} A_u(ID) \neq 0) \bigwedge (\bigcap_{ID \in S'} A_u(ID) \neq 0)] \geq 1 - \frac{q_1+q_2}{\beta_u}$, where S is defined as the identity set queried to **ParKeyGen** oracle, S' represents the identity set queried to **KeyGen** oracle. Hence, we have $Adv_5 \geq \frac{1}{2} Adv_4$.

Game$_6$: $\mathcal{C}$ conducts this game as the same as *Game*$_5$ apart from the public keys $\{g^x, g^y, g^z\}$, and x, y, z are unknown to $\mathcal{C}$. Based on the pre-defined algorithm descriptions, $\mathcal{C}$ proceeds with the oracles apart from **ParKeyGen** oracle.

Given ID of the DU, $\mathcal{C}$ will terminate if $A_u(ID) = 0$; otherwise, $\mathcal{C}$ selects $a'_u \in_R Z^*_p$ and defines $psk_{u,1} = (g^z)^{\frac{-1}{A_u(ID)}} g^{a'_u}$, $psk_{u,2} = (g^z)^{\frac{-B_u(ID)}{A_u(ID)}} H_1(ID)^{a'_u}$. If the equation $a_u = a'_u - \frac{z}{A_u(ID)}$ holds, then the partial private keys' validity can be verified by Eq. 7

$$\begin{aligned} H_1(ID) &= g^{xA_u(ID)} g^{B_u(ID)}; \\ psk_{u,1} &= g^{a'_u - \frac{z}{A_u(ID)}}; \\ psk_{u,2} &= g^{xz} (g^{xA_u(ID) + B_u(ID)})^{\frac{-z}{A_u(ID)}} H_1(ID)^{a'_u} \\ &= g^{xz} H_1(ID)^{a'_u - \frac{z}{A_u(ID)}}. \end{aligned} \tag{7}$$

Note that the distributions of *msk*, *pk*, psk_u are the same as those of *game*$_5$. Thus, we have $Adv_6 = Adv_5$.

Game$_7$: $\mathcal{C}$ conducts this game as the same as *Game*$_6$ except for the challenge phase. $\mathcal{A}_1$ issues the tuple (pk_u, ID^*, W', W''), and $\mathcal{C}$ needs to check $A_u(ID^*) \stackrel{?}{=} 0$. If $A_u(ID^*) \neq 0$, then $\mathcal{C}$ terminates; otherwise, $\mathcal{C}$ outputs a random bit $b'' \in \{0, 1\}$ and selects two elements $b, c \in_R Z^*_p$. Thus, the ciphertexts are set as $(\lambda^*_1, \lambda^*_2, \lambda^*_3, \lambda^*_4, \{\mu^*_{i,j}\})$, where $\lambda^*_1 = (g^z g^{s_u})^b$, $\lambda^*_2 = (g^x g^{t_u})^{b+c}$, $\lambda^*_3 = g^c$, $\lambda^*_4 = g^{A_u(ID^*)c}$, $\mu^*_{i,j} = g^{yh(w^*_{b''})b}$. When $b'' = 0$, then $w^*_{b''} \in W'$; otherwise $w^*_{b''} \in W''$. Finally, we can have $Adv_7 = Adv_6$ as the distribution of challenging ciphertexts does not change.

Game$_8$: $\mathcal{C}$ conducts this game as the same as *Game*$_7$ except for the generation of challenging ciphertexts. Given the DLIN instance (g, u, v, g^c, u^b, T), the challenging ciphertexts are set as $(\lambda^*_1, \lambda^*_2, \lambda^*_3, \lambda^*_4, \{\mu^*_{i,j}\})$, where $\lambda^*_1 = u^b$, $\lambda^*_2 = T$, $\lambda^*_3 = g^c$, $\lambda^*_4 = g^{A_u(ID^*)c}$, $\mu^*_{i,j} = u^{h(w^*_{b''})b}$ when $u = g^y$, $g^z = u / g^{t^*_u}$, $g^x = v / g^{s^*_u}$. Note that the tuple (x, y, z, b, c) is not used by $\mathcal{C}$ in *game*$_8$, and the distinguishable probability between *game*$_7$ and *game*$_8$ has to do with the DLIN assumption. Hence, we have $|Pr[E_8] - Pr[E_7]| \leq Adv_{DLIN}$. Besides, we also get $Pr[E_8] = \frac{1}{2}$.

According to above simulation, we have the following inequalities $Adv_1 = Adv_2 + Adv_h$, $Adv_2 = Adv_3 = \beta_u(n+1) Adv_4$, $Adv_4 \leq 2 Adv_5$, $Adv_5 = Adv_6 = Adv_7 \leq Adv_{DLIN}$. Hence, we further have $Adv_{\mathcal{A}_1} \leq Adv_h + 2(q_1 + q_2)(n+1) Adv_{DLIN}$. This completes the proof of Theorem 1. □

Theorem 2. *Assume that $\mathcal{A}_2$ makes at most q_2 queries to **KeyGen** oracle and q_3 queries to the **Trap** oracle in **Game 2**, $Adv_{h'}$ denotes $\mathcal{A}_2$'s advantage in breaking the collision-resistant hash function h, and $Adv_{DLIN'}$ represents $\mathcal{A}_2$'s advantage in breaking the DLIN assumption, then $\mathcal{A}_2$'s advantage in breaking the ciphertexts indistinguishability is $Adv_{\mathcal{A}_2} \leq Adv_{h'} + 2q_2q_3(n+1) Adv_{DLIN'}$.*

Proof. As the proof of Theorem 2 is similar to that of Theorem 1, we omit it here. The difference is that $\mathcal{A}_2$ cannot issue the queries to **ParKeyGen** oracle since $\mathcal{A}_2$ has the master key *msk*. □

Theorem 3. *For the semi-honest-but-curious CSP, it is computationally infeasible to forge the valid proof information to pass the result verification mechanism under the CDH and DL assumptions.*

Proof. To forge the proof information, the CSP can achieve it in two ways. First, we assume that the CSP forges the signature on each record. However, the CSP cannot forge the valid signature set as the signature is generated by the DO's private key. Thus, it is computationally infeasible to forge the valid signatures under the CDH assumption, the detailed proof is shown by the following Lemma 1.

Lemma 1. *It is computationally infeasible to forge the valid file signatures under the CDH assumption.*

Proof. Assume that either $\mathcal{A}_1'$ or $\mathcal{A}_2'$ can generate the valid signatures in the VMKS scheme, and there exists an algorithm $\mathcal{B}$ which can break the CDH problem, which contradicts the CDH assumption. Next, we consider two types of adversaries when proofing Lemma 1.

- **Type-I adversary**. To forge valid signatures in the VMKS scheme, $\mathcal{A}_1'$ should ask for five different types of queries to $\mathcal{B}$, namely *Setup* query, hash-I query, *ParKeyGen* query, hash-II query and *Enc* query. Moreover, $\mathcal{A}_1'$ can issue public key replacement in this security game, and the hash function H_1 is considered as a random oracle. Given the tuple $(g, g^{a'}, g^{b'})$ of CDH problem, $\mathcal{B}$ simulates the security game as follows:
  - *Setup query*. When $\mathcal{A}_1'$ issues the system initialization query, $\mathcal{B}$ returns the public system parameters (G_1, G_2, g, g_0, h, H_0, H_1) to $\mathcal{A}_1'$.
  - *hash-I query*. $\mathcal{A}_1'$ issues the hash-I query on DO with identity ID_o. Then, $\mathcal{B}$ selects an element $r^* \in_R Z_p^*$ and a random bit $\kappa \in \{0, 1\}$. If $\kappa = 1$, $\mathcal{B}$ sets $H_1(ID_o) = g^{r*}$; otherwise, $\mathcal{B}$ sets $H_1(ID_o) = (g^{b'})^{r*}$. Finally, $\mathcal{B}$ sends $H_1(ID_o)$ to $\mathcal{A}_1'$. Notice that $\mathcal{A}_1'$ cannot distinguish the value of κ according to returned $H_1(ID_o)$ as g^{r*}, $(g^{b'})^{r*}$ have different distributions in G_1.
  - *ParKeyGen query*. $\mathcal{A}_1'$ issues the partial secret key query on DO with identity ID_o. If $\kappa = 1$, $\mathcal{B}$ returns $H_1(ID_o)^{a_0} = (g^{a'})^{r^*}$; otherwise, $\mathcal{B}$ outputs $\perp$.
  - *Public key replacement*. $\mathcal{A}_1'$ picks an element $s_o^* = a' \in_R Z_p^*$ and computes $g^{s_o^*} = g^{a'}$, then $\mathcal{A}_1'$ sends $(ID_o, s_o^*, g^{s_o^*})$ to $\mathcal{B}$.
  - *hash-II query*. $\mathcal{A}_1'$ issues the query for file f_i with identity id_i, and $\mathcal{B}$ computes $\varphi_i = H_0(ID_o||g^{s_o^*}||id_i)g_0^{h(c_i)}$, where c_i denotes the ciphertext for file f_i. Then, $\mathcal{B}$ sends φ_i to $\mathcal{A}_1'$.
  - *Enc query*. $\mathcal{A}_1'$ issues the signature query for file f_i with identity id_i by submitting φ_i to $\mathcal{B}$. If $\kappa = 1$, $\mathcal{B}$ returns the signature $\delta_i = \varphi_i^{s_o^*}(g^{a'})^{r^*}$; otherwise, $\mathcal{B}$ returns $\perp$.

  Based on above game, $\mathcal{A}_1'$ forges a signature δ_i on the tuple (ID_o, f_i, id_i). Then, $\mathcal{B}$ obtains $H_1(ID_o) = (g^{b'})^{r^*}$ and $\delta_i = \varphi_i^{s_o^*}H_1(ID_o)^{s_o^*} = \varphi_i^{s_o^*}(g^{a'b'})^{r^*}$. Finally, $\mathcal{B}$ gains $g^{a'b'} = (\delta_i/\varphi_i^{s_o^*})^{r^{*(-1)}}$ as $\mathcal{B}$ has the tuple $(\delta_i, \varphi_i^{s_o^*}, r^*)$. Thus, we deduce that $\mathcal{B}$ can break the CDH problem on condition that $\mathcal{A}_1'$ forges a valid signature.
- **Type-II adversary**. To forge a valid signature, $\mathcal{A}_2'$ needs to issue the same queries as those in **Type-I adversary**, but the differences are that $\mathcal{B}$ should return master key to $\mathcal{A}_2'$ and $\mathcal{A}_2'$ cannot replace the public key of any entity. Notice that the hash function H_0 is considered as a random oracle in **Type-II adversary**. As the analysis of **Type-II adversary** is similar to that of **Type-I adversary** except the aforementioned restrictions, we omit it here, and the reader can refer to the CLKS scheme [42] to learn its detail analysis.

According to two types of security games outlined above, $\mathcal{B}$ can solve the CDH problem on condition that $\mathcal{A}_1'$ (or $\mathcal{A}_2'$) successfully forges a valid signature, which contradicts the CDH assumption shown in Definition 1. Therefore, it is computationally infeasible for adversaries to forge valid signatures on condition that CDH assumption holds. This completes the proof of Lemma 1. □

Based on Lemma 1, we assume that the CSP can forge the proof information on incorrect data by breaking the following security game:

To check the search results' correctness, the PAS sends the challenging formation $\{r, \xi_r\}_{r\in[1,q]}$ to CSP, and the proof information on correct search results C' should be $\{\theta, \delta^*\}$. However, the semi-honest-but-curious CSP may forge the proof information $\{\theta', \delta^*\}$ on the corrupted search results $C'' = <ddq>$, where $\theta' = \sum_{r=1}^{q}\xi_r h(c^*_{\rho_2(r)})$. Let $\triangle\theta = \theta' - \theta$, then $\triangle\theta \neq 0$ as $C' \neq C''$. If the forged proof information can pass the result verification mechanism, then the CSP wins the security game; otherwise, the CSP fails.

Assume the CSP can win the security game, then we get $e(\delta^*, g) = e(\prod_{r=1}^{q}(\gamma_o \cdot \beta_r)^{\xi_r}, g^{s_o})e(g_0^{\theta'}, g^{s_o})$. In addition, we have $e(\delta^*, g) = e(\prod_{r=1}^{q}(\gamma_o \cdot \beta_r)^{\xi_r}, g^{s_o})e(g_0^{\theta}, g^{s_o})$. Thus, we further get $g_0^{\theta} = g_0^{\theta'} \Leftrightarrow g_0^{\triangle\theta} = 1$. For two elements ζ_1, ζ_2 in the cyclic group G_1, there must be an element $\varpi \in_R Z_p^*$ such that $\zeta_1 = \zeta_2^{\varpi}$. Without loss of generality, g_0 can be defined as $g_0 = \zeta_1^{\varpi_1}\zeta_2^{\varpi_2}$, where $\varpi_1, \varpi_2 \in_R Z_p^*$, then we have the following Eq. (8).

$$g_0^{\triangle\theta} = (\zeta_1^{\varpi_1}\zeta_2^{\varpi_2})^{\triangle\theta} = \zeta_1^{\varpi_1\triangle\theta}\zeta_2^{\varpi_2\triangle\theta} = 1. \tag{8}$$

Clearly, there is a solution to solve the DL problem with an advantage $1 - \frac{1}{p}$. Specifically, given the elements $\zeta_2, \zeta_1 = \zeta_2^{\varpi} \in G_1$, we gain $\zeta_1 = \zeta_2^{-\varpi_2\triangle\theta/\varpi_1\triangle\theta}$, $\varpi = \frac{-\varpi_2\triangle\theta}{\varpi_1\triangle\theta}$ on the condition that $\varpi_1\triangle\theta \neq 0$. However, we know that $\triangle\theta \neq 0$ and ϖ_1 is

Table 3
Storage costs of various schemes.

Algorithms	VMKS	CLKS [42]	ABKS-UR [32]
KeyGen	$(\|\mathcal{U}\|+1)(2\|Z_p\|+2\|G_1\|)$	$\|\mathcal{U}\|(2\|Z_p\|+2\|G_1\|)$	$\|\mathcal{U}\|(\|Z_p\|+\|G_2\|)+(2\|\mathcal{N}\|+1)\|G_1\|+\|Z_p\|$
Trap	$5\|G_1\|+\|Z_p\|$	$l\|G_1\|+3\|G_1\|+\|Z_p\|$	$(2\|\mathcal{N}\|+1)\|G_1\|+2\|Z_p\|$
Search	$3\|G_2\|$	$(2+l)\|G_2\|$	$(\|\mathcal{N}\|+2)\|G_2\|$
Verify	$(q+2)\|G_1\|+(q+1)\|Z_p\|+3\|G_2\|$	–	–

"$|\mathcal{U}|$": Number of data users; "$|\mathcal{N}|$": Number of attributes in system; <ddq>"l": Number of queried keywords; <ddq>"q": Number of search results.

Table 4
Computational complexity of various schemes.

Algorithms	VMKS	CLKS [42]	ABKS-UR [32]
KeyGen	$2\|\mathcal{U}\|E_1+2E_1$	$2\|\mathcal{U}\|E_1$	$(2\|\mathcal{N}\|+1)E_1+2\|\mathcal{U}\|E_2$
Enc	$(10+m)E_1+H_0$	$(8+m)E_1$	$(\|\mathcal{N}\|+1)E_1+E_2$
Trap	$10E_1$	$(9+l)E_1$	$(2\|\mathcal{N}\|+1)E_1$
Search	$4P$	$lP+3P$	$(\|\mathcal{N}\|+1)P+E_1$
Verify	$(2q+2)E_1+qH_0+3P$	–	–

"m": Number of keyword fields for each record.

an element of Z_p. Hence, the denominator is nonzero with an advantage $1-\frac{1}{p}$. It means that we can solve the DL problem with an advantage $1-\frac{1}{p}$ if the CSP can win the aforementioned security game, which is contrary to the DL assumption. Therefore, it is computationally infeasible to forge valid proof information to pass the result verification mechanism for the semi-honest-but-curious CSP. This completes the proof of Theorem 3. □

6.3. Performance analysis

In this section, we evaluate the theoretical and practical performance of the VMKS scheme by comparing with the state-of-the-art ABKS-UR [32] and CLKS [42] schemes. With regard to theoretical performance, we mainly present the analysis in terms of storage cost and computational complexity. Afterwards, we conduct empirical experiments using a real-world dataset to demonstrate the efficiency and feasibility of our VMKS scheme.

6.3.1. Theoretical performance

Let $|G_1|$, $|G_2|$[4], $|Z_p|$ be the bit-length of an element in groups G_1, G_2 and filed Z_p, respectively. From Table 3 we notice that the VMKS scheme and the CLKS scheme have the similar storage costs $(2|\mathcal{U}|(|Z_p|+|G_1|))$ in the **KeyGen**, but the storage cost of the ABKS-UR scheme is still affected by the number of attributes in system ($|\mathcal{N}|$), which brings additional costs $(2|\mathcal{N}|+1)|G_1|$. In the **Trap** and **Search**, the storage costs of our VMKS scheme are constant and are less than those of CLKS and ABKS-UR schemes. For example, the storage overhead of trapdoor generation and ciphertexts retrieval in the VMKS scheme is $5|G_1|+|Z_p|$ and $3|G_2|$, respectively, but that of other two schemes almost linearly increases with factors $l, |\mathcal{N}|$, respectively. As the CLKS scheme cannot support results verification and the ABKS-UR scheme yields the high false positive rate caused by Bloom Filter, we just analyze the storage cost of our VMKS scheme in **Verify**. Due to the small value of q, the VMKS scheme does not incur a lot of storage costs.

In addition, we present the computational complexity of aforementioned three schemes by mainly considering the time-consuming operations, i.e., exponentiation operation E_1 (resp., E_2) in group G_1 (resp., G_2), pairing operation P, hash operation H_0 which maps an arbitrary bit string to group G_1, etc. Note that we omit the hash operation h as it is much more efficient than other operations. From Table 4, we notice that the VMKS scheme is much more efficient than the CLKS and ABKS-UR schemes in **Trap** and **Search** algorithms. For example, the computational costs of trapdoor generation and ciphertexts retrieval in the VMKS scheme are $10E_1$ and $4P$, respectively, while those of other two schemes (namely the CLSK scheme and ABKS-UR scheme) are still affected by factors $l, |\mathcal{N}|$, respectively. In **KeyGen** and **Enc**, the computational overhead of our VMKS scheme is approximately equal to that of the CLKS scheme, and these two schemes are much more efficient than the ABKS-UR scheme in **KeyGen** because of extra computational overhead $(2|\mathcal{N}|+1)E_1$. With the same reason shown in Table 3, we just show the computational complexity of VMKS scheme in **Verify**.

6.3.2. Actual performance

To assess the actual performance of the aforementioned three schemes, we perform a series of empirical experiments using a real-world dataset[5]. This public email dataset used in many SE schemes contains half a million records from about

[4] The element length in group G_2 is equal to that of group G_1.
[5] http://www.cs.cmu.edu/~enron/.

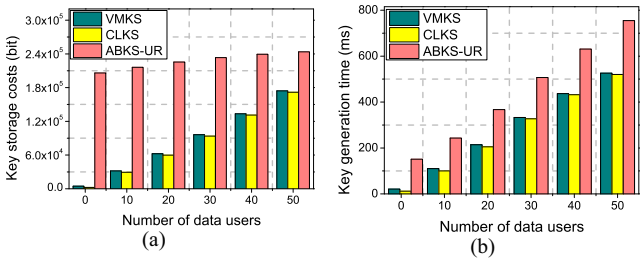

Fig. 5. Storage and computational costs in **KeyGen** algorithm: (a) Storage costs of key generation; (b) Computational costs of key generation.

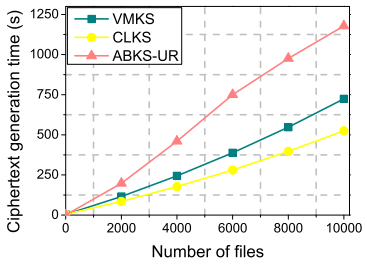

Fig. 6. Computational costs in **Enc** algorithm.

150 users, mostly senior management of Enron, and the Enron corpus contains a total of about 0.5 megabytes. The experiments are performed on an Ubuntu Server 15.04 with Intel Core i5 Processor 2.3 GHz by using C and PBC Library[6]. In PBC Library, the Type A is denoted as $E(F_\kappa): y^2 = x^3 + x$, the groups G_1 and G_2 of order p are the subgroups of $E(F_\kappa)$, where the parameter p and κ are equivalent to 160 bits and 512 bits. Thus, we have $|G_1| = |G_2| = 1024$ bits, $|Z_p| = 160$ bits. In line with the ABKS-UR scheme, we randomly select 10,000 records from this dataset and perform experiments for 100 times. For convenience, we set $|\mathcal{U}| \in [1, 50]$, $m \in [1, 1000]$, $l \in [1, 50]$, $q \in [1, 50]$. Note that we also set $|\mathcal{N}| = 100$ throughout this paper. In Fig. 5 (a) and (b), we demonstrate the storage and computational costs in **KeyGen**, respectively. We notice that the storage costs of the VMKS and CLKS schemes are affected by the number of DUs ($|\mathcal{U}| \in [1, 50]$), while that of the ABKS-UR scheme is influenced by both factors $|\mathcal{U}|$ and $|\mathcal{N}|$. For comparison, we set $|\mathcal{N}| = 100$ and vary the value of $|\mathcal{U}|$ from 1 to 50. Due to $|\mathcal{N}| \gg |\mathcal{U}|$, the storage cost of the ABKS-UR scheme is much more than those of our VMKS scheme and the CLKS scheme. In addition, the VMKS scheme is approximately equal to the CLKS scheme in terms of storage costs. For example, the storage cost of key generation in the VMKS scheme is 7.79KB when setting $|\mathcal{U}| = 20$, while those of CLKS and ABKS-UR schemes are 7.49KB and 28.2KB, respectively. For the same reason, the VMKS scheme and the CLKS scheme are superior to the ABKS-UR scheme regarding computational costs which almost linearly increase with the value of $|\mathcal{U}|$. Note that the performance rangeability of storage cost (or computational cost), which varying with the variable $|\mathcal{U}|$, in the ABKS-UR scheme is not very obvious as $|Z_p|$ is less than $|G_1|$ (or E_2 operation is much efficient than E_1 operation). For example, with the same value $|\mathcal{U}| = 20$, the computational overhead of key generation in the VMKS scheme is 214ms, while that of CLKS and ABKS-UR schemes is 205ms and 367ms, respectively.

By outsourcing the huge amount of data to CSP, the DO can reduce the high storage burden, especially for the storage and computation resource-limited cloud clients. Thus, we just analyze the computational cost of **Enc** in Fig. 6. The ABKS-UR scheme needs $|\mathcal{N}| + 1$ exponential operations for each file, while the other two schemes both just need mE_1 for the whole files with m keyword fields, thus the computational burden of the ABKS-UR scheme is still higher than that of the VMKS and CLKS schemes in spite of $m \gg |\mathcal{N}|$. In addition, as the VMKS scheme needs to generate the signature for each file in order to verify the search results' correctness, the CLKS scheme slightly outperforms our VMKS scheme. For example, when encrypting 10,000 files, the VMKS scheme needs 724 s, while the CLKS and ABKS-UR schemes take 525 s and 1178s, respectively. However, in practice, the **Enc** is just one-time cost operation, and does not affect the user search experience. Hence, the VMKS scheme is still acceptable in actual scenarios.

In Fig. 7 (a), we show that the VMKS scheme has less storage cost than those of the other two schemes in **Trap**. When supporting conjunctive keyword search, the VMKS scheme just generates an element in group G_1 and has constant trapdoor size $5|G_1| + |Z_p|$, but the CLKS scheme will yield l elements in a single search query. In other words, the storage cost of the CLKS scheme increases linearly as the number of queried keywords increases, while that of the VMKS scheme almost keeps unchanged. Besides, as the storage cost of the ABKS-UR scheme is affected by the number of attributes in system ($|\mathcal{N}| = 100$), the VMKS scheme and the CLKS scheme have less storage costs than the ABKS-UR scheme. For example, when setting $l = 50$, the trapdoor size of the VMKS scheme is 0.85KB, whereas those of the CLKS and ABKS-UR schemes are

[6] https://crypto.stanford.edu/pbc/download.html.

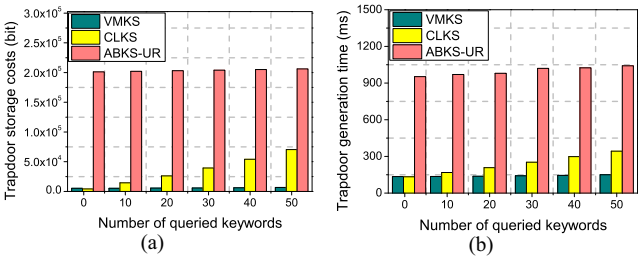

Fig. 7. Storage and computational costs in **Trap** algorithm: (a) Storage costs of trapdoor generation; (b) Computational costs of trapdoor generation.

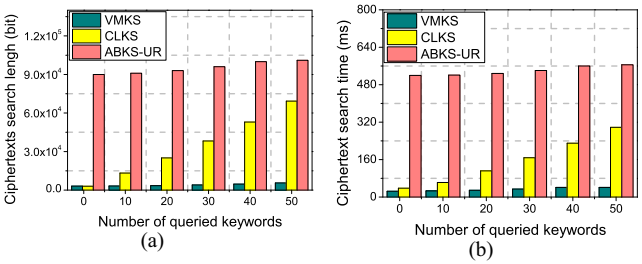

Fig. 8. Storage and computational costs in **Search** algorithm: (a) Storage costs of ciphertexts retrieval; (b) Computational costs of ciphertexts retrieval.

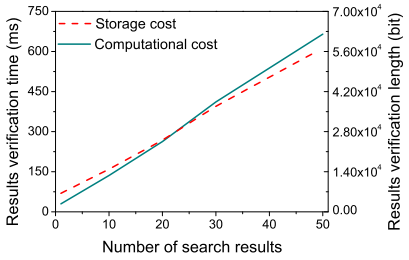

Fig. 9. Verify algorithm in VMKS scheme.

8.82KB and 25.77KB, respectively. With the same reason, we also notice that the VMKS scheme is much efficient than the CLKS and ABKS-UR schemes in terms of computational costs in Fig. 7 (b). For example, the computational cost of the VMKS scheme in terms of trapdoor generation is less than 200ms, while those of the CLKS and ABKS-UR schemes are more than 300ms and 900ms, respectively. Thus, the VMKS scheme is appropriate for the lightweight entities (i.e., mobile terminals and sensor nodes, etc.). In Fig. 8, we notice that the storage and computational costs of **Search** in the VMKS scheme and the ABKS-UR scheme are not almost affected by the number of queried keywords ($l \in [1, 50]$). In addition, without supporting conjunctive keyword search, the storage and computational costs of the CLKS scheme increase linearly as the value of l increases, but the CLKS scheme still has less storage and computational costs than those of the ABKS-UR scheme as the ABKS-UR scheme is also influenced by the variable $|\mathcal{N}|$. When setting $l = 50$, the storage and computational costs of our VMKS scheme is 0.4KB and 26ms, respectively, and those of other two schemes (i.e., CLKS scheme, ABKS-UR scheme, etc.) are (1.66KB, 62ms), (11.38KB, 521ms). Hence, the VMKS scheme provides better performance than other two schemes in the **Search**. Due to high false-positive rate incurred by Bloom Filter, the ABKS-UR cannot accurately verify the search results' correctness. Thus, in Fig. 9 we just demonstrate the storage and computational costs of **Verify** in the VMKS scheme, which almost linearly change with the number of search results ($q \in [1, 50]$). When $q = 50$, the storage cost of results verification is less than 9KB, and the computational cost is less than 600ms. In addition, the results verification operations are performed by the PAS, which will not exert high storage and computational burden on the resource-limited DUs.

7. Conclusions

The capability to search over encrypted data will be increasingly important as more of our data are being shared across services and organizations.

In this paper, we proposed a verifiable multiple keywords search scheme, which allows DUs to ensure the accuracy of search results with the help of PAS. Furthermore, the proposed scheme avoids certificate management and key escrow, as well as allowing the DUs to issue multiple keywords in a single search query. The latter feature significantly reduces computation and bandwidth costs. The proposed scheme also has constant trapdoor and ciphertexts retrieval sizes; thus, suitable

for deploying on resource-constrained devices. We proved that the VMKS scheme achieves ciphertexts indistinguishability and signatures unforgeability, and the experimental results showed that it is efficient and practical.

Future research includes extending the VMKS scheme to also support dynamic operations, such as file insertion, modification and deletion.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. 61702404, No. 61702105, No. 61672413, No. 61472310), the Project funded by China Postdoctoral Science Foundation (No. 2017M613080), the Fundamental Research Funds for the Central Universities (No. JB171504, No. 11618332), the Key Program of NSFC (No. U1405255), and the Shaanxi Science & Technology Coordination & Innovation Project (No. 2016TZC-G-6-3) and the 111 project (No. B16037).

References

[1] D. Boneh, G.D. Crescenzo, R. Ostrovsky, G. Persiano, Public key encryption with keyword search, in: Proc. International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'04), 2004, pp. 506–522, doi:10.1007/978-3-540-24676-3_30.
[2] Q. Chai, G. Gong, Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers, in: Proc. IEEE International Conference on Communications (ICC'12), 2012, pp. 917–922, doi:10.1109/ICC.2012.6364125.
[3] R. Chen, Y. Mu, G. Yang, F. Guo, X. Huang, X. Wang, Y. Wang, Server-aided public key encryption with keyword search, IEEE Trans. Inf. Forensics Secur. 11 (12) (2016) 2833–2842, doi:10.1109/TIFS.2016.2599293.
[4] R. Chen, Y. Mu, G. Yang, F. Guo, X. Wang, A new general framework for secure public key encryption with keyword search, in: Proc. Australasian Conference on Information Security and Privacy (ACISP'15), vol. 9144, 2015, pp. 59–76, doi:10.1007/978-3-319-19962-7_4.
[5] R. Chen, Y. Mu, G. Yang, F. Guo, X. Wang, Dual-server public-key encryption with keyword search for secure cloud storage, IEEE Trans. Inf. Forensics Secur. 11 (4) (2016) 789–798, doi:10.1109/TIFS.2015.2510822.
[6] Y. Chen, R. Tso, W. Susilo, X. Huang, G. Horng, Certificateless signatures: structural extensions of security models and new provably secure schemes, IACR Cryptol. ePrint Arch. 2013 (2013) 193–219.
[7] Z. Deng, K. Li, K. Li, J. Zhou, A multi-user searchable encryption scheme with keyword authorization in a cloud storage, Future Gen. Comput. Syst. 72 (2017) 208–218, doi:10.1016/j.future.2016.05.017.
[8] B. Kang, J. Wang, D. Shao, Certificateless public auditing with privacy preserving for cloud-assisted wireless body area networks, Mobile Inf. Syst. 2017 (2017), doi:10.1155/2017/2925465.
[9] K. Kurosawa, Garbled searchable symmetric encryption, in: Proc. International Conference on Financial Cryptography and Data Security (FC'14), vol. 8437, 2014, pp. 234–251, doi:10.1007/978-3-662-45472-5_15.
[10] K. Kurosawa, Y. Ohtaki, How to update documents verifiably in searchable symmetric encryption, in: Proc. International Conference on Cryptology and Network Security (CANS'13), vol. 8257, 2013, pp. 309–328, doi:10.1007/978-3-319-02937-5_17.
[11] K. Kurosawa, Y. Ohtaki, How to construct uc-secure searchable symmetric encryption scheme, IACR Cryptol. ePrint Arch. 2015 (2015) 251.
[12] H. Li, D. Liu, Y. Dai, T. Luan, S. Yu, Personalized search over encrypted data with efficient and secure updates in mobile clouds, IEEE Trans. Emerg. Top Comput. 6 (1) (2018) 97–109, doi:10.1109/TETC.2015.2511457.
[13] H. Li, D. Liu, Y. Dai, T.H. Luan, Engineering searchable encryption of mobile cloud networks: when qoe meets qop, IEEE Wireless Commun. 22 (4) (2015) 74–80, doi:10.1109/MWC.2015.7224730.
[14] H. Li, D. Liu, Y. Dai, T.H. Luan, X.S. Shen, Enabling efficient multi-keyword ranked search over encrypted mobile cloud data through blind storage, IEEE Trans. Emerg. Top. Comput. 3 (1) (2015) 127–138, doi:10.1109/TETC.2014.2371239.
[15] H. Li, Y. Yang, T.H. Luan, X. Liang, L. Zhou, X.S. Shen, Enabling fine-grained multi-keyword search supporting classified sub-dictionaries over encrypted cloud data, IEEE Trans. Dependable Secure Comput. 13 (3) (2016) 312–325, doi:10.1109/TDSC.2015.2406704.
[16] J. Li, X. Lin, Y. Zhang, J. Han, Ksf-oabe: outsourced attribute-based encryption with keyword search function for cloud storage, IEEE Trans. Serv. Comput. 10 (5) (2017) 715–725, doi:10.1109/TSC.2016.2542813.
[17] J. Li, Y. Shi, Y. Zhang, Searchable ciphertext-policy attribute-based encryption with revocation in cloud storage, Int. J. Commun. Syst. 30 (1) (2017), doi:10.1002/dac.2942.
[18] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, W. Lou, Fuzzy keyword search over encrypted data in cloud computing, in: Proc. IEEE Conference on Computer Communications (INFOCOM'10), 2010, pp. 1–5, doi:10.1109/INFCOM.2010.5462196.
[19] J. Li, W. Yao, J. Han, Y. Zhang, J. Shen, User collusion avoidance cp-abe with efficient attribute revocation for cloud storage, IEEE Syst. J. 12 (2) (2018) 1767–1777, doi:10.1109/JSYST.2017.2667679.
[20] J. Li, W. Yao, Y. Zhang, H. Qian, J. Han, Flexible and fine-grained attribute-based data storage in cloud computing, IEEE Trans. Serv. Comput. 10 (5) (2017) 785–796, doi:10.1109/TSC.2016.2520932.
[21] Y. Miao, J. Ma, X. Liu, Q. Jiang, J. Zhang, L. Shen, Z. Liu, Vcksm: verifiable conjunctive keyword search over mobile e-health cloud in shared multi-owner settings, Pervasive Mob. Comput. 40 (2017) 205–219, doi:10.1016/j.pmcj.2017.06.016.
[22] Y. Miao, J. Ma, X. Liu, X. Li, Q. Jiang, J. Zhang, Attribute-based keyword search over hierarchical data in cloud computing, IEEE Trans. Serv. Comput. PP (2017), doi:10.1109/TSC.2017.2757467. 1–1
[23] Y. Miao, J. Ma, X. Liu, X. Li, Z. Liu, H. Li, Practical attribute-based multi-keyword search scheme in mobile crowdsourcing, IEEE Internet Things J. PP (2017), doi:10.1109/JIOT.2017.2779124. 1–1
[24] Y. Miao, J. Ma, X. Liu, J. Weng, H. Li, H. Li, Lightweight fine-grained search over encrypted data in fog computing, IEEE Trans. Serv. Comput. PP (2018), doi:10.1109/TSC.2018.2823309. 1–1
[25] Y. Miao, J. Ma, Z. Liu, Revocable and anonymous searchable encryption in multi-user setting, Concurr. Comput. 28 (4) (2016) 1204–1218, doi:10.1002/cpe.3608.
[26] G.S. Poh, J.-J. Chin, W.-C. Yau, K.-K.R. Choo, M.S. Mohamad, Searchable symmetric encryption: designs and challenges, ACM Comput. Surv. 50 (3) (2017).
[27] J. Shen, D. Liu, Q. Liu, X. Sun, Y. Zhang, Secure authentication in cloud big data with hierarchical attribute authorization structure, IEEE Trans. Big Data PP (2017), doi:10.1109/TBDATA.2017.2705048.
[28] J. Shen, J. Shen, X. Chen, X. Huang, W. Susilo, An efficient public auditing protocol with novel dynamic structure for cloud data, IEEE Trans. Inf. Forensics Secur. 12 (10) (2017) 2402–2415, doi:10.1109/TIFS.2017.2705620.
[29] D.X. Song, D. Wagner, A. Perrig, Practical techniques for searches on encrypted data, in: Proc. of IEEE Symposium on Security and Privacy (S&P'00), 2000, pp. 44–55, doi:10.1109/SECPRI.2000.848445.
[30] W. Sun, X. Liu, W. Lou, Y.T. Hou, H. Li, Catch you if you lie to me: efficient verifiable conjunctive keyword search over large dynamic encrypted cloud data, in: Proc. IEEE Conference on Computer Communications (INFOCOM'15), 2015, pp. 2110–2118, doi:10.1109/INFOCOM.2015.7218596.
[31] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y.T. Hou, H. Li, Verifiable privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking, IEEE Trans. Parallel Distrib. Syst. 25 (11) (2014) 3025–3035, doi:10.1109/TPDS.2013.282.
[32] W. Sun, S. Yu, W. Lou, Y.T. Hou, H. Li, Protecting your right: verifiable attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud, IEEE Trans. Parallel Distrib. Syst. 27 (4) (2016) 1187–1198, doi:10.1109/TPDS.2014.2355202.

[33] B. Wang, B. Li, H. Li, F. Li, Certificateless public auditing for data integrity in the cloud, in: Proc. IEEE Conference on Communications and Network Security (CNS'13), 2013, pp. 136–144, doi:10.1109/CNS.2013.6682701.
[34] C. Wang, Q. Wang, K. Ren, W. Lou, Privacy-preserving public auditing for data storage security in cloud computing, in: Proc. IEEE Conference on Computer Communications (INFOCOM'10), 2010, pp. 1–9, doi:10.1109/INFCOM.2010.5462173.
[35] X.-F. Wang, Y. Mu, R. Chen, X.-S. Zhang, Secure channel free id-based searchable encryption for peer-to-peer group, J. Comput. Sci. Technol. 31 (5) (2016) 1012–1027, doi:10.1007/s11390-016-1676-9.
[36] D. Wu, Q. Liu, H. Wang, D. Wu, R. Wang, Socially aware energy-efficient mobile edge collaboration for video distribution, IEEE Trans. Multimedia 19 (10) (2017) 2197–2209, doi:10.1109/TMM.2017.2733300.
[37] D. Wu, S. Si, S. Wu, R. Wang, Dynamic trust relationships aware data privacy protection in mobile crowd-sensing, IEEE Internet Things J. PP (2017), doi:10.1109/JIOT.2017.2768073. 1–1
[38] D. Wu, J. Yan, H. Wang, D. Wu, R. Wang, Social attribute aware incentive mechanism for device-to-device video distribution, IEEE Trans. Multimedia 19 (8) (2017) 1908–1920, doi:10.1109/TMM.2017.2692648.
[39] Y. Yang, M. Ma, Conjunctive keyword search with designated tester and timing enabled proxy re-encryption function for e-health clouds, IEEE Trans. Inf. Forensics Secur. 11 (4) (2016) 746–759, doi:10.1109/TIFS.2015.2509912.
[40] R. Zhang, R. Xue, L. Liu, L. Zheng, Oblivious multi-keyword search for secure cloud storage service, in: Proc. IEEE International Conference on Web Services (ICWS'17), 2017, pp. 269–276, doi:10.1109/ICWS.2017.42.
[41] W. Zhang, Y. Lin, S. Xiao, J. Wu, S. Zhou, Privacy preserving ranked multi-keyword search for multiple data owners in cloud computing, IEEE Trans. Comput. 65 (5) (2016) 1566–1577, doi:10.1109/TC.2015.2448099.
[42] Q. Zheng, X. Li, A. Azgin, Clks: Certificateless keyword search on encrypted data, in: Proc. International Conference on Network and System Security (NSS'15), 2015, pp. 239–253, doi:10.1007/978-3-319-25645-0_16.
[43] Q. Zheng, S. Xu, G. Ateniese, Vabks: Verifiable attribute-based keyword search over outsourced encrypted data, in: Proc. IEEE Conference on Computer Communications (INFOCOM'14), 2014, pp. 522–530, doi:10.1109/INFOCOM.2014.6847976.