Optimized Verifiable Fine-Grained Keyword Search in Dynamic Multi-Owner Settings

Yinbin Miao, Robert H. Deng, *Fellow, IEEE*, Kim-Kwang Raymond Choo, *Senior Member, IEEE*, Ximeng Liu, Jianting Ning, and Hongwei Li

Abstract—Ciphertext-Policy Attribute-Based Keyword Search (CP-ABKS) schemes support both fine-grained access control and keyword-based ciphertext retrieval, which make these schemes attractive for resource-constrained users (i.e., mobile or wearable devices, sensor nodes, etc.) to store, share and search encrypted data in the public cloud. However, ciphertext length and decryption overhead in the existing CP-ABKS schemes grow with the complexity of access policies or the number of data users' attributes. Moreover, such schemes generally do not consider the practical multi-owner setting (e.g., each file needs to be signed by multiple data owners before being uploaded to the cloud server) or prevent malicious cloud servers from returning incorrect search results. To overcome these limitations, in this paper we first design an optimized Verifiable Fine-grained Keyword Search scheme in the static Multi-owner setting (termed as basic VFKSM), which achieves short ciphertext length, fast ciphertext transformation, accelerated search process, and authentic search result verification. Then, we extend the basic VFKSM to support multi-keyword search and multi-owner update (also called as extended VFKSM). Finally, we prove that the basic (or extended) VFKSM resists the Chosen-Keyword Attack (CKA) and external Keyword-Guessing Attack (KGA). We also evaluate the performance of these schemes using various public datasets.

Index Terms—Ciphertext length, decryption overhead, multi-owner setting, multi-keyword search, search result verification

1 INTRODUCTION

Cloud computing, as a relatively mature computing paradigm, is expected to continue to play an indispensable role in our data-driven society. For example, in January 2019, Amazon reported that "Amazon's cloud-computing division said revenue jumped 45 percent in the fourth quarter as the company continued to cement its lead over Microsoft and Google, and sales at Amazon Web Services climbed to \$7.43 billion from \$5.11 billion a year ago".[1] Both organizational and individual users can either outsource their data or computationally intensive tasks to cloud service providers on a pay-as-you-go or subscription-based model. Unsurprisingly, potential security and privacy risks associated with outsourcing user data and intellectual property to the cloud, as well as potential security solutions, have been extensively studied. One naive approach is to use conventional encryption mechanism to ensure the confidentiality of both data-at-rest and data-in-transit. However, such an approach limits users' capability to share and search over encrypted data, thereby affecting user search experience. This limitation motivates the design of Ciphertext-Policy Attribute-Based Keyword Search (CP-ABKS) schemes.

In addition to offering keyword-based search over encrypted data, CP-ABKS schemes [1], [2], [3], [4] achieve fine-grained access control over encrypted data. In contrast to classical Searchable Encryption (SE) solutions [5], [6], [7], CP-ABKS avoids coarse-grained access control, costly key management or unnecessary resource wastage due to multiple ciphertext copies. However, CP-ABKS schemes are still not widely deployed in practice, as computation and storage costs of existing CP-ABKS schemes such as those described in [2], [3] grow linearly with the complexity of access policies or the number of data users' attributes. Such costs can be prohibitively expensive for deployments on smartphones and Internet of Things (IoT) devices (e.g., wearable and implantable sensors in Wireless Body Area Networks (WBANs) [8]).

Ciphertext-Policy Attribute-Base Encryption (CP-ABE) [9], [10], [11], [12], a core building block of CP-ABKS, achieves one-to-many encryption by attaching access policies and attribute sets to ciphertexts and secret keys, respectively. However, one of its main limitations is that decryption overhead[2] of

1. https://www.cnbc.com/2019/01/31/aws-earnings-q4-2018.html

2. The computation time of decrypting ciphertexts.

- Y. Miao is with the School of Cyber Engineering, Xidian University, Xi'an 710071, China, and also with State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China. E-mail: ybmiao@xidian.edu.cn.
- R.H. Deng is with the School of Information Systems, Singapore Management University, 80 Stamford Road, Singapore 178902. E-mail: robertdeng@smu.edu.sg.
- K.-K.R. Choo is with the Department of Information Systems and Cyber Security, The University of Texas at San Antonio, San Antonio, TX 78249. E-mail: raymond.choo@fulbrightmail.org.
- X. Liu is with the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, Fujian 350116, China, and also with Fujian Provincial Key Laboratory of Information Security of Network Systems, Fuzhou, Fujian 350116, China. E-mail: snbnix@gmail.com.
- J. Ning is with the Department of Computer Science, National University of Singapore, 21 Lower Kent Ridge Road, Singapore 119077. E-mail: jtning88@gmail.com.
- H. Li is with the Department of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610051, China. E-mail: hongweili@uestc.edu.cn.

Manuscript received 25 Feb. 2019; revised 1 Aug. 2019; accepted 6 Sept. 2019. Date of publication 0 . 0000; date of current version 0 . 0000.
(Corresponding author: Jianting Ning.)
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TDSC.2019.2940573

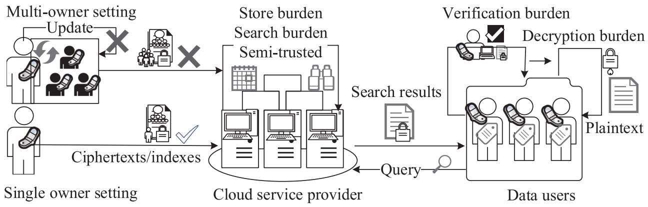


Fig. 1. Example limitations in existing schemes.

CP-ABE schemes increases linearly with the complexity of access policies or the number of data users' attributes. To minimize such decryption overhead imposed on data users, CP-ABE with outsourced decryption mechanism employs the transformation key to translate the original CP-ABE ciphertext into one that can be decrypted with one exponentiation operation. However, these schemes [13] cannot guarantee the correctness of transformation executed by the malicious cloud server. This partly contributes to the study of CP-ABE with verifiable outsourced decryption solutions [14], [15]. Another drawback is that the ciphertext length[3] increases linearly with the complexity of each access policy, since the Boolean formula describing access policy is usually designed according to Linear Secret-Sharing Schemes (LSSS) [10]. However, the size of Boolean formula is generally much bigger than the number of clauses in DNF form [16]. For instance, given an access policy "('*Department A*' and '*Professor*') or ('*Department B*' and '*Associate professor*' and '*Key program*')", the size of the Boolean formula is 5, while the number of clauses in DNF form is 2.

To achieve short ciphertext length, fast ciphertext decryption, and search result verification,[4] a lightweight CP-ABKS scheme [18] was recently designed. However, the scheme [18] still incurs additional computation and storage costs on resource-limited devices for trapdoor generation, ciphertext retrieval and search result verification. Furthermore, in common multi-owner applications [19] (i.e., government document issuance system, Personal Health Record (PHR) system, etc.) in which each file/record is co-owned by multiple data owners, data owners may leave or join this group frequently. In contrast to the single owner setting, the multi-owner setting can provide enhanced and flexible access control. For example, the issuance of key government files should be signed by multiple designated officials. However, their membership may dynamically change, for example, due to turnover or change of roles/duties. There have been other attempts to address different limitations in the literature. The scheme [7] supports multi-owner setting and search result verification at the same time, but does not support multi-owner update and fine-grained keyword search. Besides, the scheme requires all data owners (should) be online in real-time.

We summarize the limitations of previous schemes in Fig. 1. As the previous CP-ABKS schemes [1], [2], [3], [4] usually use the Boolean formula to describe the access policies based on LSSS, these schemes incur long ciphertext length which increases with the complexity of access policies or the number of data users' attributes. Additionally, the computation overhead of these outsourced tasks in traditional CP-ABE schemes [14], [15] still grows with the complexity of access policies or the number of data users' attributes, which imposes heavy search burden on the cloud server. Considering the malicious cloud server that may execute a fraction of search operations and return a fraction of false search results, the existing search result verification mechanisms in prior CP-ABKS schemes [2] suffer from high false-positive rates due to the use of Bloom filters. Furthermore, some CP-ABKS schemes [19] implemented in the multi-owner setting may easily lead to significant multi-owner update overhead.

Thus, to address existing limitations, we design an optimized Verifiable Fine-grained Keyword Search scheme in the static Multi-owner setting, hereafter referred to as the basic VFKSM, and an extended scheme in the dynamic multi-owner setting, referred to as the extended VFKSM, to achieve multi-keyword search and multi-owner update. Our proposed VFKSM schemes use the DNF form to describe the access policies, thereby shorting the ciphertext length. The outsourced decryption mechanism implemented in our schemes allows the cloud server to conduct approximately constant operations by using its secret key rather than the transformation key, which eliminates the data users' decryption burden and cloud server's search burden at the same time. The modified public auditing technique facilitates accurate search result verification without incurring false-positive rate in our schemes. The threshold signature mechanism enables our schemes to be deployed in the multi-owner setting, and allows the extended VFKSM to update multiple data owners without bringing much update overhead. Compared with the previous CP-ABKS schemes, our proposed schemes have some improvements in terms of shortening ciphertext length, reducing data users' decryption overhead and cloud server's outsourced decryption overhead, providing accurate search result verification reports as well as efficient multi-owner update. Our formal security analysis proves that basic (or extended) VFKSM is secure against Chosen-Keyword Attack (CKA) and external Keyword-Guessing Attack (KGA).[5] Through empirical tests over different public datasets, the efficiency and feasibility of basic (or extended) VFKSM are verified. Specifically, the key characteristics of the proposed schemes are summarized as follows:

- *Fine-grained access control with keyword search*. VFKSM achieves fine-grained access control over encrypted cloud data by specifying expressive access policies in terms of any Boolean formula over attributes. Different from the traditional CP-ABKS scheme [18] which just supports single keyword search, the basic VFKSM and extended VFKSM can offer single keyword search and multi-keyword search, respectively, which meet query requirements flexibly. In particular, the extended VFKSM can further minimize bandwidth and communication consumption.
- *Short ciphertext length*. Unlike the previous CP-ABKS schemes [1], [2], [3], [4] in which the ciphertext

3. The length/size of ciphertexts.

4. Unlike verifiable outsourced decryption (which guarantees the cloud has correctly executed the ciphertext transformation), the goal of search result verification is to check the correctness of search results. Similar to schemes such as those in [7], [17], the malicious cloud may be financially-motivated to delete the data that are rarely or never accessed, and to return forged or false search results after completing search operations.

5. The outside KGA is stronger than CKA. However, VFKSM cannot resist off-line (or inside) KGA, which is deferred to future work.

length generally increases with the size of Boolean formula describing the access policy, the ciphertext length in our basic (or extended) VFKSM flexibly grows linearly with the number of clauses in the DNF form or the size of Boolean formula in access structure depending on which value is smaller. Moreover, the threshold signature mechanism transforms multiple signatures generated by a threshold number of data owners into a single threshold signature, which further reduces the ciphertext length.

- *Fast (outsourced) decryption*. Without creating transformation keys in outsourced decryption processes of traditional CP-ABE schemes [14], [15], the basic (or extended) VFKSM allows the cloud server to execute the most of time-consuming pairing operations by using its secret key, which leaves a fraction of lightweight operations at the data user's side. Note that the outsourced decryption overhead does not vary with the number of clauses in the DNF form or the size of access Boolean formula, which further accelerates the search process.
- *Search result verification*. The basic (or extended) VFKSM allows a public verifier to check the correctness of search results on behalf of data users without privacy leakage. This reduces the computational requirements for resource-constrained data users. Compared with the previous verifiable CP-ABKS scheme [2] which incurs high false-positive rate caused by the Bloom filter, our basic (or extended) VFKSM scheme can accurately check the correctness of search results by modifying the prominent public auditing technique.
- *Dynamic multi-owner setting*. The basic (or extended) VFKSM can be deployed in multi-owner settings by using the threshold signature technique, where the secret key of data owner-manager is shared among multiple data owners, which can realize the multi-owner access authorizations. Superior to existing scheme [19] in the multi-owner setting, our proposed schemes just require that at least a threshold number of data owners rather than all ones should be online at the same time. Besides, the extended VFKSM allows data owners to join or leave the group dynamically without incurring substantial update overhead.

In Section 2, we review the existing literature. Section 3 introduces the relevant mathematical building blocks and security assumptions. In Section 4, we present the system model, threat model and security model. Then, we present the construction of the basic VFKSM and show it can be extended to offer multi-keyword search and multi-owner update in Section 5, followed by the security and performance analysis of basic (or extended) VFKSM in Section 6. Finally, we conclude this paper and discuss its future work in Section 7.

2 Related Work

Existing solutions to achieve lightweight fine-grained keyword search over encrypted cloud data include both Attribute-Based Encryption (ABE) and SE approaches.

Attribute-Based Encryption. Since conventional public encryption mechanisms just allow a data owner to generate ciphertext for particular data users, ABE [9], [20] supporting fine-grained access control has been an area of active research. Existing ABE approaches are either Key-Policy ABE (KP-ABE) or CP-ABE. There have been attempts to enrich the functionalities of CP-ABE, such as introducing multi-authority [21], policy update [22], and outsourced decryption, as well as enhancing its security features (i.e., traceability, policy hidden [23]). These schemes return all appropriate results on the condition that the user's attributes match with the specified access policy hidden in ciphertexts. However, not all results are required by a certain data user. Thus, CP-ABKS scheme [2], [3], [19] combining CP-ABE with SE can be an effective tool to filter uninteresting results while offering fine-grained access control. For example, inspired by CP-ABE, Sun et al. presented an owner-enforced CP-ABKS scheme [2] supporting user revocation by utilizing proxy re-encryption and lazy re-encryption. The scheme achieves both search result verification and search authorization. Miao et al. designed a lightweight CP-ABKS scheme [3] by relying on fog nodes to execute the majority of encryption and decryption tasks. While such schemes facilitate fine-grained access control with keyword search, they do not achieve short ciphertext length, the *first challenge* we attempt to address in this paper.

Searchable Encryption. Since the proposal of the first SE scheme in an asymmetric setting [5], a large number of Public Encryption with Keyword Search (PEKS) schemes with different features (i.e., exact keyword search [7], [27], [28], [29], fuzzy keyword search [30], ranked keyword search [31], etc.) have been proposed in the literature. These SE schemes achieve expressive search, but they do not allow the data owner to grant flexible search capabilities in the multi-user setting. Zhang et al. proposed a ranked multi-keyword search scheme in the multi-owner model [24]. This scheme prevents malicious attackers from intercepting the secret keys and supports user revocation. However, it only supports coarse-grained access control and single keyword search. Although the CP-ABKS scheme in the multi-owner setting [25] provides both fine-grained access control and multi-keyword search, its ciphertext length and decryption overhead are affected by the complexity of access structures. Moreover, these schemes [24], [25] do not consider the dynamic owner membership in the multi-owner setting, the *second challenge* we attempt to solve in this paper.

In addition to having the fine-grained keyword search function in [3], [25], the verifiable CP-ABKS schemes presented in [2], [26] can verify the correctness of search results via Bloom filter in the cloud computing environment. However, these schemes incur significant computation and storage costs, particularly on resource-limited devices. These schemes also lead to high false-positive rates, the *third challenge* we attempt to settle in this paper.

To address three challenges described above, we present an optimized CP-ABKS scheme in the static (or dynamic) multi-owner setting based on the public verifiability technique. A comparative summary of our proposed schemes and the existing schemes is presented in Table 1.

3 Mathematical Background

In this section, we give the definitions required in the construction of VFKSM, namely, bilinear maps, access structure, LSSS matrix, and certain security assumptions.

TABLE 1
A Comparative Summary of ABE and SE Schemes

Schemes*	$\mathbb{F}_1$	$\mathbb{F}_2$	$\mathbb{F}_3$	$\mathbb{F}_4$	$\mathbb{F}_5$	$\mathbb{F}_6$
[2]	×	×	Private	Static	Multiple	CKA
[4]	×	✓	×	×	Multiple	CPA/CKA
[7]	×	×	Private	Static	Single	CKA
[13]	×	✓	×	×	×	CPA/CCA
[14]	×	✓	Private	×	×	CPA
[15]	×	✓	Private	×	×	CPA
[16]	✓	✓	×	×	×	CPA
[18]	✓	✓	Private	×	Single	CPA/CKA
[19]	×	×	×	✓	Multiple	KGA
[24]	×	×	×	Static	Single	CKA
[25]	×	×	×	Static	Multiple	CKA
[26]	×	×	Private	×	Single	CKA
Basic VFKSM	✓	✓	Public	Static	Single	CKA/KGA
Extended VFKSM	✓	✓	Public	Dynamic	Multiple	CKA/KGA

— ∗: Schemes of [7], [24] achieve only coarse-grained access control;
— $\mathbb{F}_1$: Short ciphertext length; $\mathbb{F}_2$: Fast (or outsourced) decryption; $\mathbb{F}_3$: Result verification; $\mathbb{F}_4$: Multi-owner setting; $\mathbb{F}_5$: Keyword-based search type; $\mathbb{F}_6$: Resist attack types;
— CPA: Chosen-Plaintext Attack; CCA: Chosen-Ciphertext Attack; CKA: Chosen-Keyword Attack; KGA: Keyword-Guessing Attack.

Let $\mathbb{G}, \mathbb{G}_T$ represent two finite multiplicative groups of order p, where p is a large prime. Besides, let g be the generator of $\mathbb{G}$ and $\mathbb{Z}_p$ denote the finite field, then the bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ has three features for all $h_1^*, h_2^* \in \mathbb{Z}_p$. (1) *Bilinearity*: $e(g^{h_1^*}, g^{h_2^*}) = e(g,g)^{h_1^* h_2^*}$; (2) *Non-degeneracy*: $e(g^{h_1^*}, g^{h_2^*}) = 1$ iff $h_1^* = 0$ or $h_2^* = 0$; (3) *Efficiency*: There exists an efficient algorithm to compute $e(g^{h_1^*}, g^{h_2^*})$. In addition, the symbol $[\hbar_1, \hbar_2]$ be an integer set $\{\hbar_1, \hbar_1 + 1, \dots, \hbar_2\}$, where $\hbar_1, \hbar_2$ are both integers.

3.1 Access Structure

Let $A = \{A_1, A_2, \dots, A_n\}$ denote an attribute set, then the collection $\mathbb{A} \subseteq 2^{\{A_1, A_2, \dots, A_n\}}$ is monotone for arbitrary attribute sets B, C: if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}$. We can say $\mathbb{A}$, as a collection of non-empty subsets of $\{A_1, A_2, \dots, A_n\}$ (i.e., $\mathbb{A} \subseteq 2^{\{A_1, A_2, \dots, A_n\}} \{\emptyset\}$), is a monotone access structure. The sets in $\mathbb{A}$ are termed as the authorized sets, and the ones not in $\mathbb{A}$ are called as the unauthorized sets.

3.2 LSSS Matrix

Let $\mathbb{A}$ denote an access structure on A, then there exists a LSSS matrix $\mathcal{M} \in \mathbb{Z}_p^{l \times n}$ and a function ρ that maps each row of $\mathcal{M}$ to an attribute in $\mathbb{A}$. The tuple $(\mathcal{M}, \rho)$ denotes a LSSS access policy by combining with a column vector $\vec{y} = (s, y_2, \dots, y_n)^\top \in \mathbb{Z}_p^n$, and $\vec{\lambda} = \mathcal{M} \cdot \vec{y}$, where s represents the secret value to be shared. Assume that $\mathbb{S}$ denotes an authorized set described by $(\mathcal{M}, \rho)$, and I^* represents the set of rows in $\mathcal{M}$ such that $I^* = \{\tau | \tau \in [1, l] \wedge \rho(\tau) \in \mathbb{S}\}$, then we can find such constants $\{\omega_\tau\}_{\tau \in I^*}$ satisfying $\sum_{\tau \in I^*} \omega_\tau \lambda_\tau = s$, where $\lambda_\tau = (\mathcal{M} \cdot \vec{y})_\tau$.

3.3 Security Assumptions

We review the modified Bilinear Diffie-Hellman Exponent (BDHE) assumption [16], Computational Diffie-Hellman (CDH) assumption and Discrete Logarithm (DL) assumption, respectively.

Modified-BDHE Assumption. Given the public bilinear parameters (G, G_T, p, e, g) and a tuple $\vec{Q} = (g, g^s, g^{a'}, \cdots g^{a'^q}, g^{a'^{q+2}}, \dots, g^{a'^{2q}}, g^{s(a't+a')}, g^{a't}, g^{a'^2 t}, \dots, g^{a'^q t}, g^{a'^{q+2} t}, \dots, g^{a'^{2q} t})$, where $a', t, s, q \in \mathbb{Z}_p$, the goal of modified-BDHE problem is to distinguish $P = e(g,g)^{a'^{q+1} s}$ from a random element $P = R$ in $\mathbb{G}_T$. The modified-BDHE assumption holds on condition that there exists no Probabilistic Polynomial Time (PPT) algorithm $\mathcal{A}^*$ that can break the modified-BDHE problem with non-negligible advantage ϵ.

$$\left[\begin{array}{c} Pr[\mathcal{A}^*(\vec{Q}, P = e(g,g)^{a'^{q+1} s}) = 0] \\ -Pr[\mathcal{A}^*(\vec{Q}, P = R) = 0]| \end{array} \right] \geq \epsilon.$$

CDH Assumption. Given the tuple $(g, g^{a'}, g^{b'})$, the goal of CDH problem is to output $g^{a'b'}$. The CDH assumption holds if there exists no PPT algorithm $\mathcal{A}^*$ that can solve the CDH problem with non-negligible advantage ϵ, where $a, b \in \mathbb{Z}_p$.

$$Pr[\mathcal{A}^*(g, g^{a'}, g^{b'}) = g^{a'b'}] \geq \epsilon.$$

DL Assumption. Given the tuple $(g, g^{a'})$, the goal of DL problem is to compute $a' \in \mathbb{Z}_p$. The DL assumption holds on condition that there exits no PPT algorithm $\mathcal{A}^*$ that can break the DL problem with non-negligible advantage ϵ.

$$Pr[\mathcal{A}^*(g, g^{a'}) = a'] \geq \epsilon.$$

4 Problem Formulation

In this section, we show the system model, threat model and security model, respectively.

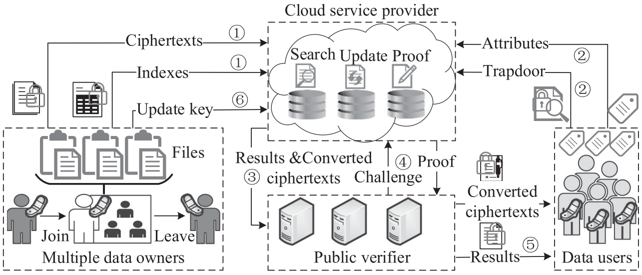


Fig. 2. The system model of VFKSM.

4.1 System & Threat Models

System Model. In the system model shown in Fig. 2, we consider a cloud-based data outsourcing scenario, which mainly consists of four types of entities, namely Data Owners (DOs), Data Users (DUs), Cloud Service Provider (CSP) and Public Verifier (PV). For example, an enterprise remotely outsources its sensitive files to the public cloud server so that its authorized staffs (DUs) have access to these data anytime and anywhere. Before being outsourced to the cloud, each file should be signed by a certain fraction of the department managers (DOs), which avoids the restriction that all department managers should be online in real-time to sign the files. To cater to organizational changes in the enterprise (e.g., change of position or employment), VFKSM should support multi-owner update. Note that the Trusted Authority (TA), which is responsible for generating secret keys for various entities and initializing system parameters, is omitted in Fig. 2. The specific role of each entity is described below:

- *Data owners*. Outsourcing of sensitive files is authorized by multiple DOs. DO manager, on behalf of multiple DOs, encrypts file[6] encryption keys by using the specified access policy and builds indexes according to the extracted keywords (Step ①). When the membership of DOs changes, DO manager outputs an update key and sends it to CSP which just needs to update parts of the ciphertexts (Step ⑥).
- *Cloud service provider*. CSP, which owns almost unlimited storage and computation capabilities, stores massive data and offers ciphertext retrieval services according to DUs' search requests (Step ②). CSP also performs the costly ciphertext transformation in the search phase, thereby leaving lightweight ciphertext decryption operations on DUs. Once DUs' attributes and trapdoors match with the access policies and indexes, respectively, CSP returns the search results along with the converted ciphertexts to PV (Step ③).
- *Public verifier*. Upon receiving the search results, PV calls the search result verification mechanism by interacting with CSP in a challenge-response mode (Step ④). If the search results are correct, PV sends them as well as the converted ciphertexts to DUs (Step ⑤). Otherwise, it rejects. Note that PV can check the correctness of search results without learning DOs' or DUs' sensitive information.
- *Data users*. Authorized DUs can issue keyword-based search queries including single keyword or multiple keywords, receive verified and transformed ciphertexts from PV, and perform lightweight decryption to obtain the desired search results in plaintext form.

Threat Model. TA and multiple DOs are completely trusted entities, but authorized DUs may collude with each other. PV is assumed to be honest-but-curious in the sense that it honestly obeys the protocols but is curious to try to obtain sensitive information. CSP is assumed to be malicious, which follows the predefined protocols to accomplish ciphertext retrieval, update and transformation tasks. However, due to data corruptions (incurred by inevitable software and hardware failures) or lose, CSP may return some false search results. Furthermore, one threat is CSP may try to forge the valid threshold signature to pass the search result verification. To avoid the leakage of DUs' secret keys, it requires that DUs cannot collude with CSP and PV as DUs' secret keys can be used to generate valid trapdoors and decrypt returned results. Finally, the proposed schemes may suffer from other threats (i.e., CKA, KGA, etc.) launched by outside attackers, which is the goal to be achieved. It is worth noticing that the potential threats (i.e., access pattern leakage, search pattern leakage, etc.) in most of existing SE schemes or CP-ABKS schemes still cannot be avoided in our proposed schemes, which are beyond the scope of our discussion.

4.2 Security Model

To prove that the basic (or extended) VFKSM is selectively secure, we use the security game defined in [10]. This probabilistic game is conducted between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, which is shown as follows:

- *Setup*. $\mathcal{C}$ calls **Setup** to output public key PK and master key MSK. Note that the initial value of list L_q that marks $\mathcal{A}$'s queries is set to null.
- *Phase 1*. $\mathcal{A}$ adaptively selects an attribute set S and sends it to $\mathcal{C}$ for secret key generation query. Then, $\mathcal{C}$ forwards the corresponding secret key sk_u to $\mathcal{A}$ by calling **KeyGen**. Note that S is added in L_q.
- *Challenge*. $\mathcal{A}$ submits a target access structure $\mathbb{A}^*$ as well as two messages M_0^*, M_1^* with equal length. Then, $\mathcal{C}$ chooses a random bit $b' \in \{0,1\}$ and executes **Enc** to obtain the ciphertext $C_{b'}^*$.
- *Phase 2*. $\mathcal{A}$ repeatedly asks the secret key queries as *Phase 1*.
- *Guess*. $\mathcal{A}$ outputs a guess bit $b'' \in \{0,1\}$. $\mathcal{A}$ wins this security game on condition that $b'' = b'$ and all attribute sets in L_q do not match with $\mathbb{A}^*$; otherwise, $\mathcal{A}$ fails. The advantage of $\mathcal{A}$ in wining this game is defined as $Adv_{\mathcal{A}} = Pr[b'' = b'] - \frac{1}{2}$.

According to the above security model, our basic (or extended) VFKSM is secure if all PPT adversaries have at most a negligible advantage to break above security game. Note that the system is selectively secure if the *Init* phase is added before the *Setup* phase, in which the selective adversary $\mathcal{A}$ commits to a challenging access structure $\mathbb{A}^*$.

Definition 1. *VFKSM is selectively secure if there exist no adversaries that can break the above selective security game with non-negligible probability.*

In the following security game regarding CKA,[7] $\mathcal{A}$ is allowed to issue queries with some restrictions. That is, $\mathcal{A}$ cannot distinguish the ciphertexts between keywords w_0 and w_1 and have the corresponding trapdoor. This security game is also performed between $\mathcal{A}$ and $\mathcal{C}$.

- *Init*. $\mathcal{A}$ declares the challenging attribute set S^*.
- *Setup*. $\mathcal{C}$ calls **Setup**(1^k) to return the public key PK and sends them to $\mathcal{A}$.
- *Phase 1*. $\mathcal{A}$ adaptively issues a polynomial bounded number of queries to the following queries:
    - *Secret key query*: $\mathcal{A}$ submits S to $\mathcal{C}$ for secret key query, where $S \neq S^*$. Then, $\mathcal{C}$ returns the associated secret key sk_u to $\mathcal{A}$ by calling **KeyGen**.
    - *Trapdoor query*: $\mathcal{A}$ submits S, w to $\mathcal{C}$. $\mathcal{C}$ outputs the associated trapdoor T_w and sends T_w to $\mathcal{A}$ by calling **Trap**.
- *Challenge*. $\mathcal{A}$ submits two challenging keywords w_0, w_1 and attributes S^*, but it requires that queries about trapdoors for w_0, w_1 or secret keys for S^* have not been issued previously. Then, $\mathcal{C}$ selects a random bit $b' \in \{0,1\}$ and returns the challenging index $I_{1,w_{b'}}^*$ to $\mathcal{A}$.
- *Phase 2*. $\mathcal{A}$ continues to issue secret key queries for S and trapdoor queries for w, but one restriction is that $S \neq S^*$, $w \neq w_0, w_1$, which has the similar process as *Phase 1*.
- *Guess*. $\mathcal{A}$ returns a guess bit $b'' \in \{0,1\}$ and wins this security game on condition that $b' = b''$.

6. These files are generally encrypted by the symmetric encryption algorithms (i.e., DES, AES, etc.)

7. Note that both outside attackers and cloud server (is also called as an inside attacker) can launch CKA, but only cloud server can issue the secret key query.

Definition 2. *VFKSM is secure against CKA if there exist no adversaries that can win the above security game with non-negligible advantage.*

As the outside KGA security game is similar to CKA security game except for *Phase 1* and *Challenge*, we just demonstrate the differences between these two security games. In *Phase 1* of KGA security game, $\mathcal{A}$ is permitted to issue the *Secret key query* as well as the *Index query* rather than the *Trapdoor query* shown in the CKA security game. In the *challenge* phase, $\mathcal{A}$ should not have asked for the index queries for challenging keywords w_0, w_1 or secret key query for S^*, and $\mathcal{C}$ returns the challenging trapdoor $T^*_{b'}$ of keyword $w_{b'}$ to $\mathcal{A}$. The detailed KGA security game is referred to schemes [32], [33].

Definition 3. *VFKSM can resist the outside KGA on condition that there exist no adversaries that can successfully attack the KGA security game.*

The threshold signature mechanism, in which the secret key is distributed among multiple DOs with the help of trusted DO manager, is also utilized in VFKSM to support the multi-owner setting. The security of VFKSM requires that there exist no adversaries that can corrupt up to a threshold number of DOs to forge valid threshold signatures. The similar security game is shown in [34], [35]. Note that our proposed schemes just utilize the idea of threshold signature, but have different threshold signature generation algorithms. One of main differences is that the threshold signatures in [34] do not involve the identities of search results, which cannot efficiently prevent CSP from forging the incorrect search results.

Definition 4. *The threshold signature in VFKSM is unforgeable if any PPT adversary can win above security game with negligible advantage.*

5 Proposed Basic & Extended VFKSM

In this section, we first describe the basic VFKSM to achieve our claimed features (i.e., fine-grained single keyword search, short ciphertext length, outsourced ciphertext decryption, convincing search result verification, static multi-owner setting, etc.), which does not impose high computation and storage burden on resource-limited cloud clients (i.e., smartphones, sensors, wearable devices, etc.). Then, we extend the basic scheme to support multi-keyword search and dynamic multi-owner setting to further enhance its adaptability and scalability in practice. Note that these two added advantages do not incur much additional computation and storage overhead.

5.1 Concrete Construction of the Basic VFKSM

In this section, we show the high-level description of the basic VFKSM in Fig. 3. To shorten the ciphertext length in most of existing CP-ABKS scheme, which grows with the complexity of access policies, the basic VFKSM uses the DNF form [16] rather than the Boolean formula designed by LSSS to describe the specified access policies, note that the number of clauses in DNF form is usually much less than the size of Boolean formula. Different from the state-of-the-art CP-ABE with outsourced decryption schemes [14], [15]

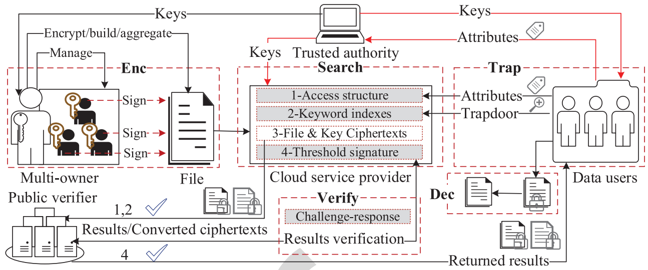


Fig. 3. Overview of the basic VFKSM.

in which the outsourced decryption overhead still increases with the complexity of access policies or the number of data users' attributes, the basic VFKSM allows CSP to execute the constant pairing operation by utilizing its secret key rather than the transformation key, which not only eliminates DUs' decryption burden but also accelerates the search process.

As the malicious CSP may execute a fraction of search tasks and return a fraction of incorrect search results, our basic VFKSM modifies the public auditing mechanism [36], [37] to guarantee the correctness of search results, and allows the public verifier rather than the private verifier to accomplish the search result verification tasks, which further relieves DUs' computation burden. Furthermore, our basic VFKSM scheme can be implemented in the static multi-owner setting. For example, a patient's Personal Health Records come from multiple doctors. The encrypted PHRs can be shared among multiple authorized DUs, but the correctness of each encrypted PHR should be guaranteed by the threshold signature derived from multiple doctors. Simply extending this search result verification mechanism in multi-owner settings easily leads to much search result verification time and storage overhead in the encryption process, which increase with the number of data owners. Moreover, the straightforward solution requires that all DOs (should) be online in real-time, which lacks practicability and availability in practice. The basic VFKSM will use the threshold signature technique to avoid this limitation, note that the secret key is distributed among multiple DOs so that each DO can generate the signature for the same file. Hence, the basic VFKSM will involve a trusted DO manage that executes secret key share distribution, file (key) encryption and indexes building on behalf of multiple DOs.

In the key generation process, TA in the basic VFKSM outputs the secret key for DO manager so that he can manage multiple DOs, encrypt files and associated symmetric keys by using the access structure, and generate the threshold signature for each file according to multiple DOs' signatures. In **Enc**, the basic VFKSM flexibly uses the access structure defined by DNF form or Boolean formula using LSSS in the worst case to generate ciphertexts, and utilizes the threshold signature technique to form a single signature for each file according to multiple DOs' signatures. In **Trap**, the basic VFKSM can facilitate the single keyword search. In **Search**, CSP not only can match DUs' attributes (or trapdoors) with the specified access policies (or indexes), but also execute ciphertext transformation by using its allocated secret key, which dramatically reduces DUs' ciphertext decryption overhead, note that the ciphertext transformation process is not

Concrete construction of the basic VFKSM

Setup(1^k). Given the public bilinear parameters $\mathcal{BP} = (\mathbb{G}, \mathbb{G}_T, e, p, g)$, TA chooses $\alpha_1, \alpha_2, \beta \in \mathbb{Z}_p$, $h \in \mathbb{G}$, and computes $e(g,g)^\alpha$, g^β, where $\alpha = \alpha_1 + \alpha_2$. Then, TA generates n random elements $h_1, \cdots, h_n \in \mathbb{G}$ for the global attribute set $A = \{A_1, \cdots, A_n\}$. Finally, TA selects two anti-collision hash functions $H_0 : \{0,1\}^* \to \mathbb{Z}_p$, $H_1 : \{0,1\}^* \to \mathbb{G}$. TA outputs the public key $PK = (\mathcal{BP}, h, H_0, H_1, h_1, \cdots, h_n, e(g,g)^\alpha, g^\beta)$ and master key $MSK = (\alpha_1, \alpha_2)$.

KeyGen$(PK, MSK, S, \mathcal{O})$. TA runs this algorithm to generate secret/public key pairs for each DU, CSP and DOs, respectively. As for each DU with attribute set $S = \{Att_1, \cdots, Att_{n'}\}$, TA selects $a, b, r \in \mathbb{Z}_p$ and computes $sk_{u,1} = g^{\alpha_2} g^{r\beta}$, $sk_{u,2} = a$, $sk_{s,1} = g^{\alpha_1} g^{r\beta}$, $sk_{s,2} = g^r$, $sk_{s,3} = b$, $pk_u = g^a$, $pk_s = g^b$. For each attribute $Att_i (i \in [1, n'])$, TA computes $sk_{s,i} = h_i^r$. Assume that there exist a DO manager that manages DO group $\mathcal{O} = \{O_1, \cdots, O_d\}$, TA distributes a secret/public key pair $(sk_o = c, pk_o = g^c)$ for DO manager, where $c \in \mathbb{Z}_p$. Then, DO manager outputs a $(\eta - 1)$-degree polynomial $f(x) = a_{\eta-1}x^{\eta-1} + \cdots + a_1 x + a_0$, where $a_j \in \mathbb{Z}_p (j \in [1, \eta - 1]), a_0 = c$, $2\eta - 1 \geq d$. Besides, DO manager chooses d points $\{(x_1, y_1), \cdots, (x_d, y_d)\}$ according to $f(x)$, where $y_{i'}$ is returned to $O_{i'} (i' \in [1, d])$ via a security channel (e.g., Secure Sockets Layer). Finally, TA marks secret/public pairs of each DU, CSP and DO manager with Eq. 1.

$$\begin{aligned}
&(sk_u, pk_u) = ((g^{\alpha_2} g^{r\beta}, a), g^a); (sk_o, pk_o) = (c, g^c);\\
&(sk_s, pk_s) = ((g^{\alpha_1} g^{r\beta}, g^r, b, \{sk_{s,i}\}), g^b).
\end{aligned} \tag{1}$$

Enc$(PK, \mathcal{F}, \mathcal{W}, \mathbb{A}, sk_o, pk_u, pk_s)$. Given the file set $\mathcal{F} = \{m\}$ and keyword set $\mathcal{W} = \{w\}$, DO manager needs to generate file ciphertexts and keyword indexes as follows:

As for each file $m \in \mathcal{F}$ with identity ID, DO manager first encrypts it as C_m by using the traditional symmetric encryption key $K \in \mathbb{G}_T$. Assume that the specified access structure $\mathbb{A}$ is expressed in DNF form and its size is $|\mathbb{A}|$, namely $\mathbb{A} = cl_1 \vee \cdots \vee cl_v$, where each clause $cl_{j'} (j' \in [1, v])$ is a set of attributes. If $v \leq |\mathbb{A}|$, DO manager selects the secret share $s \in \mathbb{Z}_p$ and outputs the key ciphertexts $C' = K \cdot e(g,g)^{s\alpha}$, $C'' = g^s$, $\hat{C} = g^{\beta s}$, $C_{j'} = \prod_{Att_i \in cl_{j'}} h_i^s$. Otherwise, DO manager describes $\mathbb{A}$ as LSSS matrix $(\mathcal{M}, \rho(\cdot))$ (see Section 3.2)[a], then he/she selects a random vector $\vec{y} = (s, y_2, \cdots, y_n)$ and computes $\lambda_\tau = (\mathcal{M} \cdot \vec{y})_\tau (\tau \in [1, l])$, finally he/she generates the ciphertext $C_\tau = g^{\beta\lambda_\tau} h_{\rho(\tau)}^{-s}$. Thus, DO manager can achieve short ciphertext length by comparing v and $|\mathbb{A}|$.

To verify the correctness of search results later, each $O_{i'}$ generates his signature $\sigma_{m,i'} = [H_1(ID) g^{H_0(C_m)}]^{y_{i'}}$. After gaining at least η (threshold value) signatures, DO manager outputs the threshold signature $\sigma_m = \prod_{i'=1}^{\eta} \sigma_{m,i'}^{L_{i'}(0)} = (H_1(ID) g^{H_0(C_m)})^c$ to shorten the ciphertext length, where $L_{i'}(0) = \prod_{1 \leq \ell \leq \eta, \ell \neq i'} \frac{-x_\ell}{x_{i'} - x_\ell}$.

Besides, DO manager also needs to build index for each keyword $w \in \mathcal{W}$. He first selects a random element $\varpi \in \mathbb{Z}_p$ and computes $I_{1,w} = g^{a\varpi} g^{H_0(w)\varpi}$, $I_2 = e(g^b, h)^\varpi$. Finally, DO manager returns the ciphertexts $CT = (C_m, \sigma_m, C', C'', \hat{C}, \{C_{j'}\})$ (note that $\{C_{j'}\}$ may be replaced by $\{C_\tau\}$) and indexes $I = (\{I_{1,w}\}, I_2)$ to CSP.

Trap(S, sk_u, pk_s, PK, w'). When intending to search encrypted files containing keyword w', DU first chooses two random elements $\theta_1, \theta_2 \in \mathbb{Z}_p$, then calculates $T_1 = (hg^{-\theta_1})^{\theta_2}$, $T_2 = g^{b\theta_1\theta_2}$, $T_{w'} = \theta_2(a + H_0(w')) \bmod p$. Finally, DU sends his attributes S and trapdoor $T = (T_1, T_2, T_{w'})$ to CSP.

Search$(PK, \mathbb{A}, CT, I, T, S, sk_s)$. CSP first checks whether DU's attribute set S satisfies $\mathbb{A}$. If not, CSP ends this process. Otherwise, CSP keeps on verifying whether the trapdoor T matches with the indexes I with the equation $e(I_{1,w}, (T_1^{sk_{s,3}} T_2)^{1/T_{w'}}) \stackrel{?}{=} I_2$. If this equation does not hold, CSP returns $\perp$. Otherwise, CSP returns the corresponding search results $\{C_m^*\}$.

To diminish the heavy decryption burden imposed on resource-limited DUs, CSP executes the partial decryption according to the following two cases. If the number of ciphertext components in CT is equal to $v + 5$, CSP computes the converted ciphertext φ by calling Eq. 2, where v denotes the number of clauses in DNF form. Otherwise, CSP is able to gain φ by computing $e(\hat{C}^{-1} \prod_{\tau \in I^*} C_\tau^{-\omega_\tau}, sk_{s,2}) e(C'', sk_{s,1} \prod_{\tau \in I^*} sk_{s,\rho(\tau)}^{-\omega_\tau})$, note that $\sum_{\tau \in I^*} \omega_\tau \lambda_\tau = s$. Finally, CSP sends the tuple $\phi = \{C_m^*, \varphi, C', C''\}$ to PV.

$$\varphi = \frac{e(C'', sk_{s,1} \prod_{Att_i \in cl_{j'}} sk_{s,i})}{e(sk_{s,2}, \hat{C}^2 \cdot C_{j'})}. \tag{2}$$

Verify(PK, ϕ). Let the number of search results $\{C_m^*\}$ be π, and the identity of each search result $C_{\tau'}^* (\tau' \in [1, \pi])$ be $ID_{\tau'}^*$. First, PV selects $b_{\tau'} \in \mathbb{Z}_p$ and sends the challenging information $\{\tau', b_{\tau'}\}$ to CSP. Then, CSP computes $\mu^* = \sum_{\tau'=1}^{\pi} b_{\tau'} H_0(C_{\tau'}^*)$, $\sigma^* = \prod_{\tau'=1}^{\pi} (\sigma_{\tau'}^*)^{b_{\tau'}}$, where $\sigma_{\tau'}^* = (H_1(ID_{\tau'}^*) g^{H_0(C_{\tau'}^*)})^c$. Then, CSP sends the proof information (μ^*, σ^*) to PV. Finally, PV claims that search results $\{C_m^*\}$ are correct if Eq. 3 holds, then forwards ϕ to DU.

$$e(\sigma^*, g) \stackrel{?}{=} e(\prod_{\tau'=1}^{\pi} H_1(ID_{\tau'}^*)^{b_{\tau'}} g^{\mu^*}, pk_o). \tag{3}$$

Dec(PK, ϕ, sk_u). To gain the file decryption key for each result C_m^*, DU first computes $K^* = C' / (e(sk_{u,1}, C'') \cdot \varphi)$, then gains the plaintext m^* by decrypting C_m^* with K^*.

a. If $v > |\mathbb{A}|$, the basic VFKSM employs the similar encryption algorithm in [11], but the basic VFKSM does not need to generate the ciphertext components $\{g^{r_\tau}\}$, which still shortens the ciphertext length.

Fig. 4. Detailed algorithms in the basic VFKSM.

affected by the complexity of access policies or the number of DUs' attributes. Before returning the search results to DUs, PV can accurately check the correctness of search results in **Verify**, based on the unforgeable threshold signature. In **Dec**, DUs just need to execute a fraction of decryption tasks as most of them have been outsourced to CSP.

The detailed design of the basic VFKSM involves seven algorithms in Fig. 4. For ease of reference, some notation descriptions are summarized in Table 2.

Remark. Unlike the traditional CP-ABKS schemes [1], [2], [3], [26] in which the ciphertext length is proportional to the number of attributes in access structure, the basic VFKSM has short ciphertext length depending on the number of clauses in DNF form (note that each clause is usually composed of several attributes). It is worth noticing that DUs, who outsource computationally intensive tasks to CSP, just take one pairing operation to gain the file decryption key in the basic VFKSM, and CSP needs two pairing operations to accomplish ciphertext transformation regardless of the number of clauses in DNF form or the size of access Boolean formula. However, the ciphertext transformation overhead in previous CP-ABKS schemes linearly increases with the

TABLE 2
Notation Descriptions in the Basic VFKSM

Notations	Descriptions	Notations	Descriptions
$A = \{A_1, \cdots, A_n\}$	Global attributes	$\mathcal{F} = \{m\}$	File set
$S = \{Att_i\}$	DU's attributes	$\mathcal{W} = \{w\}$	Keyword set
$\mathcal{O} = \{O_1, \cdots, O_d\}$	DO group	$\mathbb{A}$	Access structure
$\mathbb{A} = cl_1 \vee \cdots \vee cl_v$	DNF form	$cl_{j'}$	Each clause
$I = (\{I_{1,w}\}, I_2)$	Indexes	σ_m	Aggregate signature
$T = (T_1, T_2, T_{w'})$	Trapdoor	φ	Converted ciphertext
$\phi = \{C_m^*, \varphi, C', C''\}$	Returned results	$\{\tau', b_{\tau'}\}$	Challenge
$(\mathcal{M}, \rho(\cdot))$	LSSS matrix	(μ^*, σ^*)	Proof

complexity of access structures or the number of DUs' attributes, which seriously degrades DUs' search experience, especially for real-time applications. Besides, the basic VFKSM considers a common scenario, namely multi-owner setting, which not only avoids all DOs being online in real-time but also allows DUs to be assured that their accessed results are correct. Last but not least, the basic VFKSM utilizes the threshold signature mechanism to further shorten the ciphertext length and lessen computation costs. Although the basic VFKSM has outstanding advantages, it still cannot offer multi-keyword search and multi-owner update, which are two essential requirements for extensive deployments in practice. In later sections, we will extend the basic VFKSM to support multi-keyword search (see Section 5.2) and multi-owner update (see Section 5.3), respectively.

5.2 Extended VFKSM with Multi-Keyword Search

The basic VFKSM facilitates the single keyword search, but arouses the efficiency concerns due to the waste of computation and bandwidth resources. Moreover, the trapdoor generation cost in most of existing multi-keyword search schemes increases linearly with the number of queried keywords. As a consequence, the basic VFKSM is not capable of being widely deployed in practical platforms involving resource-constrained devices. Accordingly, an efficient multi-keyword search functionality should be equipped in the basic VFKSM to accelerate the search process without loss of data privacy. Next, we just elaborate the modified algorithms in extended VFKSM to achieve multi-keyword search, –see Fig. 5. Note that the multi-keyword search includes conjunctive keyword search and disconjunctive keyword search. The extended VFKSM supports conjunctive keyword search, which returns each result containing all queried keywords. However, each result returned by the disconjunctive keyword search mechanism contains at least one queried keyword.

5.3 Extended VFKSM with Multi-Owner Update

The basic VFKSM allows at least a threshold number of DOs to sign each file in the multi-owner setting, but is not sufficient to deal with dynamic DO group. This is because DO membership in this group may vary from time to time. For example, a DO may be cleared out from this group due to bad behaviors or added owing to well-deserved reputation. The straightforward solution is to let existing DOs generate new signatures by using the updated secret keys, which leads to unnecessary waste of computation resources. To avoid existing DOs re-computing signatures in dynamic multi-owner settings, the threshold proxy re-signature mechanism [40] will be employed to enable CSP to recompute the threshold signature on behalf of multiple DOs, which relieves DOs from the burdensome signature update tasks. Note that the scheme [40] just allows multiple proxies to transform the same signature into multiple signatures according to their respective secret shares, which incurs high computation overhead in the signature verification process. Besides, each signature has two parts, which further brings high storage overhead in the encryption process. Compared with the traditional CP-ABKS scheme [19] supporting the multi-owner setting, the extended VFKSM can efficiently support multi-owner update by regenerating the secret/public key pair for DO manager and re-signing key used for updating threshold signatures, which avoids updating the whole ciphertexts. The multi-owner update algorithm consists of two cases, –see Fig. 6.

When adding a new DO into the original DO group $\mathcal{O} = \{O_1, \ldots, O_d\}$, DO manager sets $d^* = d + 1$ and checks whether $2\eta - 1 > d^*$ holds.[8] If $2\eta - 1 > d^*$, DO manager chooses a new point (x_{d+1}, y_{d+1}) from the polynomial $f(x)$ and distributes y_{d+1} to O_{d+1}. Note that DO manager is not required to update the outsourced file signatures in this case. If $2\eta - 1 \leq d^*$, TA needs to create a new secret/public key pair $(sk_o^* = c^*, pk_o^* = g^{c^*})$ for DO manager who then generates a $(\eta^* - 1)$-degree polynomial $f^*(x) = a^*_{\eta^*-1}x^{\eta^*-1} + \ldots + a_1^* x + a_0^*$ such that $2\eta^* - 1 > d^*$, where $a_0^* = c^*$. Besides, DO manager distributes new shares $\{(x_1^*, y_1^*), \ldots, (x_{d^*}^*, y_{d^*}^*)\}$ for the new DO group. Finally, DO manager sends the re-signing key $rkey = c^*/c$ to CSP that will update each threshold signature σ_m as $\sigma_m^* = \sigma_m^{rkey}$.

When revoking a certain DO in the original group, TA also needs to distribute secret/public key $(sk_o^* = c^*, pk_o^* = g^{c^*})$ for the chosen DO manager. Then, DO manager sets $d^* = d - 1$ and chooses a new $(\eta^* - 1)$-degree polynomial $f^*(x)$. Next, DO manager will execute the similar process outlined above.

6 Analysis of Our VFKSM

The security and performance analyses of our basic (or extended) VFKSM are demonstrated in this section, respectively.

6.1 Security

As the extended VFKSM has the same security as the basic VFKSM, in this section we just prove the security of our basic VFKSM based on the security models defined in Section 4.2.

To obtain the security under the modified-BDHE assumption, we assume that clauses in DNF form are disjoint sets, which implies that each attribute should not be reused in the Boolean formula.

Theorem 1. *Suppose that $\mathbb{A}^*$ is the challenging access Boolean formula (or access structure) which can be described as $\mathbb{A}^* = cl_1^* \vee \ldots \vee cl_v^*$ and used to construct LSSS matrix $\mathcal{M}^*_{l^* \times n^*}$ and the map function ρ^*, where $\{cl^*_{j'}\}(j' \in [1, v])$ are disjoint attribute sets, VFKSM is selectively secure under modified-BDHE assumption if $\mathcal{M}$ satisfies $l^*, n^* \leq q$.*

8. This aims to guarantee that more than half of DOs should output their signatures for each file.

Modified algorithms in the extended VFKSM

Enc$(PK, \mathcal{F}, \mathcal{W}, \mathbb{A}, sk_o, pk_u, pk_s)$. As for indexes building, DO manager chooses $\varpi \in \mathbb{Z}_p$ and computes $I_1 = g^{a\varpi}$, $I_2 = e(g^b, h)^{\varpi}$, $I_w = g^{H_0(w)\varpi}$ for each keyword $w \in \mathcal{W}$. The indexes are defined as $I = (I_1, I_2, \{I_w\})$.

Trap$(S, sk_u, pk_s, PK, W', L)$. When issuing the search query for keyword set $W' = \{w'_1, \cdots, w'_z\}$, DU recalculates $T_{W'} = \theta_2(a + \sum_{k'=1}^{z} H_0(w'_{k'}))$ and sends $T = (T_1, T_2, T_{W'})$ as well as L to CSP, where L denotes the locations of queried keywords in $\mathcal{W}$. The location privacy, which is out of the scope of our discussion, can be protected by Pseudo-random permutation functions [39], [40].

Search$(PK, \mathbb{A}, CT, I, T, L, S, sk_s)$. If DU's attributes satisfy the specified access structure $\mathbb{A}$, CSP checks whether the submitted trapdoor T matches with the indexes I by utilizing $e(I_1 \cdot \prod_L I_w, (T_1^{sk_{s,3}} T_2)^{1/T_{W'}}) \stackrel{?}{=} I_2$.

Fig. 5. Extended VFKSM supporting multi-keyword search.

Proof. In this proof, we just consider the case $v \leq |\mathbb{A}|$. If $v > |\mathbb{A}|$, the basic VFKSM is still secure, which has similar security analysis demonstrated in [10]. If there exists an adversary $\mathcal{A}$ that can break the security game with non-negligible advantage $Adv_{\mathcal{A}}$, then there exists a challenger $\mathcal{C}$ that is able to solve the modified-BDHE problem.

Setup. Let the tuple $\vec{Q} = (g, g^s, g^\beta, \ldots, g^{\beta^q}, g^{\beta^{q+2}}, \cdots, g^{\beta^{2q}}, g^{s(t\beta+\beta)}, g^{t\beta}, g^{\beta^2 t}, \ldots, g^{\beta^q t}, g^{\beta^{q+2} t}, \ldots, g^{\beta^{2q} t})$ be an instance of modified-BDHE assumption, $\mathcal{A}$ submits a challenging access structure $\mathbb{A}^* = cl_1^* \vee \ldots \vee cl_v^*$, then $\mathcal{C}$ forms a LSSS matrix $(\mathcal{M}^*_{l^* \times n^*}, \rho^*)$, where $l^*, n^* \leq q$. Besides, $\mathcal{C}$ chooses $\alpha_1^*, \alpha_2^* \in \mathbb{Z}_p$ and sets $\alpha_1 = \alpha_1^* + \beta^{q+1}/4$, $\alpha_2 = \alpha_2^* + \beta^{q+1}/4$, then calculates $e(g,g)^{\alpha_1} = e(g,g)^{\alpha_1^*} e(g^{\beta/2}, g^{\beta^q/2})$, $e(g,g)^{\alpha_2} = e(g,g)^{\alpha_2^*} e(g^{\beta/2}, g^{\beta^q/2})$.

Assume that $\mathcal{C}$ finds disjoint sets $(I_1^* \ldots, I_v^*)$ of rows in $\mathcal{M}^*$ and each clause $cl_{j'}^*$ is defined as $cl_{j'}^* = \{\rho^*(\tau), \tau \in I_{j'}^*\}$, then $\mathbb{A}^*$ can be rewritten as $\mathbb{A}^* = (\wedge \rho^*(\tau))_{\tau \in I_1^*} \vee \ldots \vee (\wedge \rho^*(\tau))_{\tau \in I_v^*}$. To further program the tuple $h_1, \ldots, h_n$, $\mathcal{C}$ implicitly defines a column vector $\vec{y} = (t, t\beta, \ldots, t\beta^{n^*-1})^\top$. Let $\vec{\lambda} = \mathcal{M}^* \cdot \vec{y}$ be the vector shares, then we can have $\lambda_\tau = \sum_{i \in [1,n^*]} \mathcal{M}^*_{\tau,i} t\beta^{i-1}$. Then, $\mathcal{C}$ finds the set $\{\omega_\tau\}(\tau \in [1, l^*])$ such that $\sum_{\tau \in I_{j'}^*} \omega_\tau \lambda_\tau = t$.

For each group element $h_i (i \in [1, n])$, there must exist an index $\tau \in [1, l^*]$ such that $i = \rho^*(\tau)$. Then, $\mathcal{C}$ selects $z_i \in \mathbb{Z}_p$ and calculates $h_i = g^{z_i} g^{\omega_\tau \sum_{k^* \in [1,n^*]} \mathcal{M}^*_{\tau,k^*} t\beta^{k^*}} = g^{z_i} g^{\beta \omega_\tau \lambda_\tau}$, note that the tuple $(\mathcal{M}^*, g^{t\beta^{k^*}})$ has known to $\mathcal{C}$. Otherwise, $\mathcal{C}$ selects a random element $z_i \in \mathcal{Z}_p$ and sets $h_i = g^{z_i}$. As the element z_i is selected randomly, the tuple $\{h_1, \ldots, h_n\}$ is distributed randomly. Finally, $\mathcal{C}$ outputs the public key $PK = (g, g^\beta, e(g,g)^\alpha, h_1, \ldots, h_n)$ and returns them to $\mathcal{A}$.

Phase 1. $\mathcal{A}$ adaptively submits an attribute set S that does not satisfy $\mathcal{M}^*$, $\mathcal{C}$ responds to this query by selecting a vector $\vec{y^*} = (y_1^*, \ldots, y_{n^*}^*)$, where $y_1^* = -1$, note that $\vec{y^*} \cdot \mathcal{M}^*_\tau = 0$ for all $\rho^*(\tau) \in S$. Then, $\mathcal{C}$ selects a random element $r^* \in \mathbb{Z}_p$ and sets $r = r^* + y_1^* \beta^q + y_2^* \beta^{q-1} + \ldots + y_{n^*}^* \beta^{q-n^*+1}$, then computes $sk_{s,2} = g^{r^*} \prod_{i=1}^{n^*} (g^{\beta^{q+1-i}})^{y_i^*} = g^r$. According to the definition of element r, the unknown terms in g^α can be omitted in the process of generating secret keys. Thus, $\mathcal{C}$ is capable of outputting $sk_{s,1} = g^{\alpha_1} g^{r^*\beta} \prod_{i=1}^{n^*} (g^{\beta^{q+2-i}})^{y_i^*}$, $sk_{u,1} = g^{\alpha_2} g^{r^*\beta} \prod_{i=1}^{n^*} (g^{\beta^{q+2-i}})^{y_i^*}$. If there exist no indices $\tau \in [1, l^*]$ such that $\rho^*(\tau) = Att_i$, $\mathcal{C}$ knows z_i and calculates $h_i^r = (g^r)^{z_i}$ for each attribute $Att_i \in S$. Otherwise, $\mathcal{C}$ computes $h_i^r = (g^r)^{z_i} g^{(r^* + y_1^* \beta^q + y_2^* \beta^{q-1} + \ldots + y_{n^*}^* \beta^{q-n^*+1}) \omega_\tau \sum_{i=1}^{n^*} \mathcal{M}^*_{\tau,i} t\beta^i}$. Due to $\vec{y^*} \cdot \mathcal{M}^*_\tau = 0$, $\mathcal{C}$ cannot deduce the unknown terms in the form of $g^{\beta^{q+1} t}$, then he is not able to compute h_i^r. If Att_i does not belong to S but there exist indices $\tau \in [1, l^*]$ satisfying $\rho^*(\tau) = Att_i$, $\mathcal{C}$ still cannot calculate h_i^r because of $\vec{y^*} \cdot \mathcal{M}^*_\tau \neq 0$.

Challenge. $\mathcal{A}$ first submits two messages K_0^*, K_1^* with equal length, then $\mathcal{C}$ picks a random bit $b' \in \{0, 1\}$ and outputs $C_{b'}^* = K_0^* P \cdot e(g^s, g^{\alpha_1^*}) e(g^s, g^{\alpha_2^*})$, $C^* = g^s$, $C_{j'}^* = g^{s(\beta + t\beta)} g^{\sum_{\tau \in I_{j'}^*} s z_{\rho^*(\tau)}} = (g^\beta \prod_{\tau \in I_{j'}^*} g^{z_{\rho^*(\tau)}} g^{\beta \omega_\tau \lambda_\tau})^s = (g^\beta \prod_{\tau \in I_{j'}^*} h_{\rho^*(\tau)})^s$, where $j' \in [1, v]$. Note that $C_{b'}^*$ is in the correct form on condition that $P = e(g,g)^{\beta^{q+1} s}$.

Phases 2. $\mathcal{C}$ answers $\mathcal{A}$'s queries as the same as *Phase 1*, but one restriction is that the queried attribute set S should not satisfy $\mathcal{M}^*$.

Guess. $\mathcal{A}$ first outputs his guess bit $b'' \in \{0, 1\}$, then $\mathcal{C}$ outputs '0' to imply that $P = e(g,g)^{\beta^{q+1} s}$ on condition that $b'' = b'$. Otherwise, $\mathcal{C}$ returns '1' to show that P is a random element in $\mathbb{G}_T$.

If the equation $P = e(g,g)^{\beta^{q+1} s}$ holds, then we can say $\mathcal{C}$ successfully simulates this security game with an advantage $Pr[\mathcal{C}(\vec{Q}, P = e(g,g)^{\beta^{q+1} s}) = 0] = \frac{1}{2} + Adv_{\mathcal{A}}$. If P is randomly chosen from the group $\mathbb{G}_T$, then $K_{b'}^*$ is completely hidden from $\mathcal{A}$ and $\mathcal{C}$'s advantage in successfully simulating this security game is defined as $Pr[\mathcal{C}(\vec{Q}, P = R) = 0] = \frac{1}{2}$. That is, $\mathcal{C}$ succeeds to break the modified-BDHE problem, which conflicts the modified BDHE assumption. Hence, there exists no such $\mathcal{A}$ that can win the above security game. This completes the proof of Theorem 1. □

Apart from protecting the privacy of file encryption key, DUs also should prevent their query privacy from being leaked to outside attackers that can launch CKA. Similar to PEKS scheme [5], [41], the outside attacker cannot distinguish an index of keyword w_0 from another index of keyword w_1 as he/she is not able to obtain valid trapdoor in the security game (see Section 4.2). Based on the security of Identity-Based Encryption (IBE) [41], the basic VFKSM can resist CKA, which can be proved by Theorem 2.

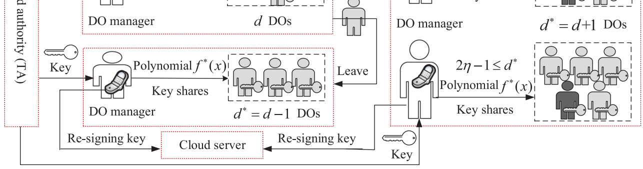


Fig. 6. Two cases in dynamic multi-owner settings.

Theorem 2. *The basic VFKSM is secure against CKA in the random oracle model on condition that there exist no adversaries that can break the security of IBE scheme [41].*

Proof. Assume that there exists an outside attack that can obtain the underlying keywords in the basic VFKSM, then there exists an adversary that is able to gain the identities in IBE scheme [41]. If the outside attacker can correctly guess queried keywords in the basic VFKSM, he/she needs to deduce the secret keys of both CSP and DUs. Therefore, basic VFKSM is not susceptible to CKA.

Before showing the main security analysis, we recall the truncated decisional Augmented Bilinear Diffie-Hellman Exponent (ABDHE) assumption. The goal of ϑ-ABDHE problem is to output $e(g,\tilde{g})^{(\tilde{a}^{\vartheta+1})}$ when given the tuple $(\tilde{g},\tilde{g}^{(\tilde{a}^{\vartheta+2})},g,g^{\tilde{a}},g^{\tilde{a}^2},\ldots,g^{(\tilde{a}^{\vartheta})})$. Clearly, the truncated ϑ-ABDHE problem is hard if the ϑ-ABDHE problem is hard. An algorithm $\mathcal{A}$ has advantage ϵ in solving the truncated ϑ-ABDHE problem if $Pr[\mathcal{A}(\tilde{g},\tilde{g}_{\vartheta+2},g,g_1,\ldots,g_\vartheta)=e(\tilde{g},g_{\vartheta+1})]\geq\epsilon$, where $g_i=g^{(\tilde{a}^i)}$, $g'_i=\tilde{g}^{(\tilde{a}^i)}$.

Assume that there exists an adversary $\mathcal{A}$ that can break the security of Gentry's IBE scheme [41], we construct a challenger $\mathcal{C}$ that can solve the truncated decisional ϑ-ABDHE problem. Given the ϑ-ABDHE instance as $(\tilde{g},\tilde{g}_{\vartheta+2},g,g_1,\ldots,g_\vartheta,Z)$, where Z denotes a random element or $e(\tilde{g},g_{\vartheta+1})$ of group $\mathbb{G}_T$, $\mathcal{C}$ executes the security game (shown in Section 4.2) as follows. It is worth noticing that we set $\tilde{g}=g^a$, $\tilde{a}=\alpha_2$ in the proof of Theorem 2.

Init. $\mathcal{A}$ declares the challenging attribute set S^*.

Setup. Apart from creating public key PK for $\mathcal{A}$ and master key MSK for itself in this phase, $\mathcal{C}$ also selects a random polynomial $p_1(x)\in\mathbb{Z}_p$ of degree ϑ and sets the public key component h as $g^{p_1(\alpha_2)/b}$.

Phase 1. When responding to the secret key queries and trapdoor queries issued by $\mathcal{A}$ in *Phase 1*, $\mathcal{C}$ first checks whether $S\stackrel{?}{=}\alpha_2=\tilde{a}$. If $S=\alpha_2$, $\mathcal{C}$ can use $\alpha_2=\tilde{a}$ to solve the truncated ϑ-ABDHE problem immediately. Otherwise, $\mathcal{C}$ defines a $(\vartheta-1)$-degree polynomial $\tilde{p}_1(x)=(p_1(x)-p_1(-H_0(w)))/(x+H_0(w))$, and then generates DU's secret key as $(a=\alpha_2,g^{\tilde{p}_1(\alpha_2)})$. Regarding the trapdoor queries, $\mathcal{C}$ chooses random elements $\theta_1,\theta_2\in\mathbb{Z}_p$, and computes $T_1=(g^{p_1(x)/b}g^{-\theta_1})^{\theta_2}$, $T_2=h^{\theta_1\theta_2b^2/p_1(x)}$, $T_w=\theta_2(p_1(S)+H_0(w))$. Finally, $\mathcal{C}$ returns the secret key and trapdoor to $\mathcal{A}$.

Challenge. $\mathcal{A}$ submits the challenging attribute set S^* and two challenging keywords w_0,w_1. Similar to *Phase 1*, if $S^*=\alpha_2$, $\mathcal{C}$ can find a solution to truncated ϑ-ABDHE problem immediately. Otherwise, $\mathcal{C}$ selects a random bit $\kappa\in\{0,1\}$, then defines two polynomials $p_2(x)=x^{\vartheta+2}$, $\tilde{p}_2(x)=(p_2(x)-p_2(-H_0(w_\kappa)))/(x+H_0(w_\kappa))$, finally computes $I_{1,w_\kappa}=\tilde{g}^{p_2(\alpha_2)-p_2(-H_0(w_\kappa))}$, $\chi=Z\cdot e(\tilde{g},\prod_{i=0}^{\vartheta}g^{t_i\tilde{a}^i})$, $I_2=e(I_{1,w_\kappa},g^{\tilde{p}_1(\alpha_2)})\chi^{p_1(-H_0(w_\kappa))}$ by setting $\varpi=(log_g\tilde{g})\tilde{p}_2(\alpha_2)$. If $Z=e(g_{\vartheta+1},\tilde{g})$, then we can gain $I_{1,w_\kappa}=g^{\varpi(\alpha_2+H_0(w_\kappa))}=g^{a\varpi}g^{H_0(w_\kappa)\varpi}$, $I_2=e(g^b,h)^{\varpi}$. Therefore, the tuple (I_{1,w_κ},I_2) is valid index for keyword w_κ on condition that $Z=e(g_{\vartheta+1},\tilde{g})$.

Phase 2. This phase is similar to *Phase 1*. The restriction is that $\mathcal{A}$ cannot issue queries for S^*,w_0,w_1.

Guess. $\mathcal{A}$ outputs a guess bit $\kappa'\in\{0,1\}$. If $\kappa'=\kappa$, $\mathcal{C}$ outputs "1" which indicates $Z=e(g_{\vartheta+1},\tilde{g})$. Otherwise, $\mathcal{C}$ outputs "0" which indicates that Z is a random element in $\mathbb{G}_T$.

According to above security game, $\mathcal{C}$ can successfully simulate this security game if $Z=e(g_{\vartheta+1},\tilde{g})$. This indicates that $\mathcal{C}$ can solve the truncated ϑ-ABDHE problem, which conflicts the truncated ϑ-ABDHE assumption. In fact, the outside attacker cannot break the security of IBE scheme [49], then the outside attacker also cannot launch CKA in the basic (or extended) VFKSM. This completes the proof of Theorem 2. The similar security proof can be referred to schemes [5], [41]. □

Additionally, most of existing CP-ABKS scheme still suffer from KGA as the keyword dictionary is always restricted to low-entropy keyword space, which makes it possible for outside attackers to know the potentially sensitive information by exhaustively guessing the candidate keywords. To avoid this kind of attacks, the basic VFKSM does not allow the outside attackers to test whether the indexes of interest match with the submitted trapdoor, and even to test whether different trapdoors are generated by the same search query. Thus, the basic VFKSM is sufficient to resist the outside KGA, which can be guaranteed by Theorem 3.

Theorem 3. *The basic VFKSM is secure against the outside KGA in the random oracle.*

Proof. To resist the outside KGA, our proposed basic VFKSM should guarantee the trapdoor indistinguishability, and does not allow the malicious attackers to test the relationship between indexes and trapdoors. It is worth noticing that the idea of index and trapdoor generation process comes from the following identity-based online/offline key encapsulation and encryption solution. The identities in this scheme are treated as keywords in basic VFKSM, and the basic VFKSM is constructed by utilizing the multiplicative bilinear map rather than the additive bilinear map. In the *Challenge* phase [42], the outside attackers cannot distinguish the ciphertexts for challenging identities based on the Bilinear Diffie-Hellman Inversion (BDHI) assumption, which solves the first issue for resisting KGA. Thus, our basic VFKSM can also satisfy the trapdoor indistinguishability. To further avoid the outside attackers testing the relationship between indexes and trapdoors in an exhaustive manner (note that keywords are often chosen from a low-entropy keyword space), the designated tester [33] is utilized to execute the matching tasks by allowing TA to distribute secret key for CSP. Thus, without the secret keys, the outside attackers cannot check whether the submitted trapdoor matches with the indexes, which avoids the possibility of keyword guessing. For example, assume that the outside attack is allowed to check the correctness of equation $e(pk_s,T_1)e(g,T_2)=e(pk_s,h^{1/(a+H_0(w'))})$, but he/she cannot guess a correct keyword without secret keys sk_u,sk_s. □

Regarding the search result verification mechanism, it requires that PV should be able to check the correctness of search results, and the adversary cannot forge valid proof information on the suspected search results. As the threshold signature contributed by at least a threshold number of

DOs can be transformed into the signature signed by DO manager based on Lagrange polynomial interpolation, PV can verify the correctness of search results by using the public key pk_o. Thus, the search result verification mechanism can be considered as the modified public auditing scheme [43], which satisfies the security requirements (i.e., soundness, completeness, etc.) of public auditing scheme. Our proposed schemes are assumed to be sound if any malicious CSP without storing the intact tuple (ID, C_m, σ_m) cannot convince PV. The completeness of our proposed schemes requires that, for all key pairs (sk_o, sk_o) and tuples (ID, C_m, σ_m) of all files $\mathcal{F} = \{m\}$, PV always decides that the search results pass the search result verification when receiving the valid proof information of CSP. To prove the unforgeability of the basic VFKSM, we will take two cases into consideration. In the first case, it is computationally feasible for the adversary to forge a valid threshold signature on each file, even this adversary can corrupt up to $(\eta - 1)$ DOs. In the second case, it is still computationally impossible for the adversary to forge valid proof information according to the whole correct threshold signatures. The unforgeability can be proved by Theorem 4.

Theorem 4. *It is computationally infeasible for an adversary to forge valid proof information on returned results under CDH and DL assumptions.*

Proof. We prove the unforgeability of basic VFKSM in terms of following two cases:

Case 1. Assume that an adversary $\mathcal{A}$ is able to forge the valid threshold signature, then we can find a solution to the CDH problem. For the (η, d)-threshold signature $(2\eta - 1 > d)$, it requires that at least η DOs output their respective signatures correctly. Even though a malicious $\mathcal{A}$ can collude with up to $(\eta - 1)$ DOs and output $(\eta - 1)$ secret/public key pairs independently, he/she still cannot generate the valid threshold signature. Next, we show a security game in which $\mathcal{A}$ is allowed to access both hash and signing oracles. Given the tuple $(g, g^{a'}, g^{b'})$, an algorithm $\mathcal{B}$, which marks the queried results of above two oracles, simulates this security game as follows.

Setup Query. $\mathcal{A}$ requests the initialization of this system, and he/she first outputs the secret/public key pairs $\{(sk_j = y_j, pk_j = g^{y_j})\}$ for $(\eta - 1)$ corrupted DOs. Then, $\mathcal{B}$ sets the same public key $pk_{rest} = g^{a'}$ for the rest of DOs and sends pk_{rest} to $\mathcal{A}$.

Hash Query. $\mathcal{A}$ first issues the hash query for the file m with identity ID. Then, $\mathcal{B}$ checks whether the tuple (m, ID) belongs to the table Tab_H. If this is true, $\mathcal{B}$ returns the corresponding result H^* to $\mathcal{A}$. Otherwise, $\mathcal{B}$ selects $\psi \in \mathbb{Z}_p$ and a random bit $\kappa \in \{0, 1\}$, note that $\kappa = 1$ with the probability ξ and '0' otherwise. If $\kappa = 1$, $\mathcal{B}$ defines $H^* = H_1(ID) g^{H_0(C_m)} = g^{\psi}$, where C_m denotes the ciphertext of file m. Otherwise, $\mathcal{B}$ sets $H^* = g^{\psi} g^{b'}$. Then, $\mathcal{B}$ sends H^* to $\mathcal{A}$. Note that $\mathcal{B}$ needs to keep the tuple (m, C_m, ID, ψ, H^*) in Tab_H. As $g, g^{\psi}, g^{b'}$ are group elements in $\mathbb{G}$, both g^{ψ} and $g^{\psi} g^{b'}$ are distributed randomly in $\mathbb{G}$. Hence, $\mathcal{A}$ is not able to distinguish the result of κ according to received hash results.

Signing Query. $\mathcal{A}$ issues the result of signing query on the tuple (m, ID), then $\mathcal{B}$ checks whether (m, ID) is an entity in table Tab_S. If this is true, $\mathcal{B}$ returns the related result σ' to $\mathcal{A}$. Otherwise, $\mathcal{B}$ sets $\sigma'_{rest} = (g^{\psi})^{a'} = (g^{a'})^{\psi}$, $\sigma'_j = (g^{\psi})^{y_j} = (g^{y_j})^{\psi} (j \in [1, \eta - 1])$, and generates the threshold signature $\sigma' = \prod_{j=1}^{\eta} (\sigma'_j)^{L_j(0)} = \prod_{j=1}^{\eta-1} (g^{y_j})^{L_j(0)\psi} (g^{a'})^{L_{rest}(0)\psi}$ on condition that $\kappa = 1$, where $L_j(0)$ is Lagrange basis polynomial. If $\kappa = 0$, $\mathcal{B}$ returns "$\perp$". Note that $\mathcal{B}$ also needs to add the tuple $(m, C_m, ID, \psi, \sigma')$ into Tab_S.

Forgery. $\mathcal{A}$ returns a forged threshold signature σ'' on (m', ID'). According to the corresponding results $g^{b'} g^{\psi'}$, $\sigma'' = \prod_{j=1}^{\eta-1} (g^{y_j})^{L_j(0)\psi'} (g^{a'})^{L_{rest}(0)\psi'}$ generated in the processes of hash query and signing query, respectively, $\mathcal{B}$ has $\sigma'' = \prod_{j=1}^{\eta-1} (g^{y_j})^{L_j(0)\psi'} (g^{b'} g^{\psi'})^{a' \cdot L_{rest}(0)}$, thereby obtaining $g^{a'b'} = (\sigma'' / (\prod_{j=1}^{\eta-1} (g^{y_j})^{L_j(0)\psi'} (g^{a'})^{L_{rest}(0)\psi'}))^{L_{rest}(0)^{-1}}$. In other words, $\mathcal{B}$ can solve the CDH problem if $\mathcal{A}$ successfully forges the threshold signature, which conflicts the CDH assumption. Hence, it is computationally infeasible for $\mathcal{A}$ to output the valid threshold signature in the basic VFKSM.

Case 2. If $\mathcal{A}$ can generate the valid proof information according to all valid threshold signatures, then we can solve the DL problem, which also conflicts the DL assumption. As the threshold signature mechanism utilized in the basic VFKSM is based on BLS signature [44], this case has similar security proof shown in [36], [37].

According to the above analysis, the basic VFKSM is computationally infeasible for $\mathcal{A}$ to forge valid threshold signature and proof information. This completes the proof of Theorem 4. □

6.2 Performance

As the extended functionalities (i.e., multi-keyword search, dynamic multi-owner setting, etc.) in extended VFKSM do not incur much extra computation overhead, we demonstrate the efficiency of our basic (or extended) VFKSM by comparing with some outstanding schemes, such as Attribute-Based encryption with efficient Verifiable Outsourced Decryption (ABVOD [15]), Lightweight Fine-Grained Search (LFGS [18]). We give the comprehensive performance analysis of schemes above from two aspects (i.e., theoretical analysis, experimental tests, etc.). We also compare the actual performance of our proposed schemes with the latest CP-ABKS scheme which supports the multi-owner setting [19] in **Search** and **Dec**. It is worth noticing that we omit the theoretical performance of scheme [19] as this scheme is less efficient than LFGS scheme.

6.2.1 Theoretical Analysis

To analyze the theoretical performance (see Table 3) of schemes listed above in terms of computation and storage overhead, we first introduce some time-consuming operations, such as modular exponentiation operation $\mathbb{E}$ (or $\mathbb{E}_T$) in group $\mathbb{G}$ (or $\mathbb{G}_T$), hash operation $\mathbb{H}_1$ which maps any string with arbitrary length into a group element in $\mathbb{G}$.[9] Then, we set the element lengths in $\mathbb{G}, \mathbb{G}_T, \mathbb{Z}_p$ as $|\mathbb{G}|$, $|\mathbb{G}_T|$ and $|\mathbb{Z}_p|$, respectively.

9. As the hash operation $\mathbb{H}_0$ which maps arbitrary string to an element in $\mathbb{Z}_p$ is much more efficient than above three operations, we omit $\mathbb{H}_0$ when analyzing the computation overhead.

TABLE 3
Theoretical Computation and Storage Analysis: A Comparative Summary

Schemes	ABVOD [15]		LFGS [18]		Basic VFKSM	
	Computation costs	Storage costs	Computation costs	Storage costs	Computation costs	Storage costs
KeyGen	$(n'+3)\mathbb{E}$	$(n'+2)\lvert\mathbb{G}\rvert+\lvert\mathbb{Z}_p\rvert$	$(n'+5)\mathbb{E}$	$(n'+4)\lvert\mathbb{G}\rvert$	$(n'+7)\mathbb{E}$	$(n'+6)\lvert\mathbb{G}\rvert+(d+3)\lvert\mathbb{Z}_p\rvert$
Enc	$(3n+1)\mathbb{E}+\mathbb{E}_T$	$(2n+1)\lvert\mathbb{G}\rvert+\lvert\mathbb{G}_T\rvert$	$(2v+5)\mathbb{E}+\mathbb{E}_T$	$(v+4)\lvert\mathbb{G}\rvert+\lvert\mathbb{G}_T\rvert$	$(v+6)\mathbb{E}+2\mathbb{E}_T+\mathbb{H}_1+\mathbb{P}$	$(v+4)\lvert\mathbb{G}\rvert+2\lvert\mathbb{G}_T\rvert$
Trap	—	—	$(n'+7)\mathbb{E}$	$(n'+6)\lvert\mathbb{G}\rvert+\lvert\mathbb{Z}_p\rvert$	$3\mathbb{E}$	$2\lvert\mathbb{G}\rvert+\lvert\mathbb{Z}_p\rvert$
Search	$(2n'+1)\mathbb{P}+n'\mathbb{E}_T$	$(2n'+1)\lvert\mathbb{G}_T\rvert$	$6\mathbb{P}$	$5\lvert\mathbb{G}_T\rvert$	$3\mathbb{P}+3\mathbb{E}$	$2\lvert\mathbb{G}_T\rvert$
Verify	$\odot$	$\odot$	$\odot$	$\odot$	$(2\pi+1)\mathbb{E}+\pi\mathbb{H}_1+2\mathbb{P}$	$(\pi+1)\lvert\mathbb{Z}_p\rvert+2\lvert\mathbb{G}\rvert+2\lvert\mathbb{G}_T\rvert$
Dec	$\mathbb{E}_T$	$\lvert\mathbb{G}_T\rvert$	$\mathbb{E}_T$	$\lvert\mathbb{G}_T\rvert$	$\mathbb{P}$	$\lvert\mathbb{G}_T\rvert$

__Notes__. n: Number of attributes in access structure; n': Number of DU's attributes; d: Number of DOs in basic VFKSM; π: Number of search results; "—": ABVOD does not support keyword search in __Trap__; $\odot$: Without consideration.

The results in Table 3 show that the performance of **Enc**, **Trap** and **Search** in basic VFKSM[10] is superior to that in the other schemes. With distributing secret/public key pairs for CSP and multiple DOs, the basic VFKSM has slightly more computation and storage overhead than ABVOD and LFGS schemes in **KeyGen**. Compared with the state-of-the-art LFGS scheme, the basic VFKSM does not increase the ciphertext length in **Enc** while supporting multi-owner settings. Moreover, the computation and storage costs of **Trap** in basic VFKSM just increase with the number of queried keywords but not the number of DU's attributes, and those of extended VFKSM are not affected by these two variables. The theoretical analysis of **Search** consists of two parts, the performance of the first part is affected by the number of queried keywords, and that of the second part is affected by the number of search files. When searching one encrypted file based on a single keyword, the performance of the basic VFKSM is better than that of the other two schemes. Although the performance of **Verify** in basic VFKSM is not much efficient than the naive checking mechanism based on file/keyword hash table, the basic VFKSM can support privacy-preserving search result verification by leveraging a public verifier, which further relieves resource-limited DUs from additional computation burden. Finally, these three schemes all have lightweight decryption overhead (about one exponentiation operation $\mathbb{E}_T$ or pairing operation $\mathbb{P}$) in **Dec**.

6.2.2 *Experimental Tests*

To appraise the actual performance of our basic VFKSM and extended VFKSM in practice, a series of empirical tests are simulated on an Ubuntu Server 18.04 with Intel Core i5-7200 CPU 2.5 GHz through C language and Paring Based Cryptography (PBC) Library. Note that the dataset used in these experiments derives from the public Enron Email Dataset,[11] which has a size of 422 MB and includes about 517431 emails from 151 users distributed in 3500 folders. Furthermore, the common Type A curve $E(F_q): y^2 = x^3 + x$, which is deemed to have 80-bit security level, is selected in PBC Library. $\mathbb{G}, \mathbb{G}_T$ of order p are subgroups of $E(F_q)$, where the lengths of p and q are 160 bits and 512 bits, respectively. Then, we have $|\mathbb{Z}_p| = 160$ bits, $|\mathbb{G}| = |\mathbb{G}_T| = 1024$ bits. The experimental results are shown in Fig. 7. It is worth noticing that we only show the performance of extended VFKSM in **Trap** and **Search** as the extended VFKSM has similar performance with the basic VFKSM in **KeyGen**, **Enc**, **Verify** and **Dec**.

From Figs. 7a and 7b, we notice that these three schemes have similar computation and storage overhead of keys generation in **KeyGen**, which linearly grows with the number of each DU's attributes ($n' \in [1, 50]$). Due to additional keys distribution for CSP and the specified DO manager (i.e., $g^{\alpha_1}g^{r\beta}$, g^b of CSP, g^c of DO manager), the basic VFKSM has slightly more key generation overhead (including keys generation time and storage costs) than ABVOD and LFGS schemes. With bringing a small amount of key generation overhead, the basic VFKSM can be easily implemented to offer ciphertext transformation on CSP, search result verification in multi-owner settings, which implies that the basic VFKSM can increase its practicability by sacrificing performance.

The overhead of ciphertext generation and storage in each file in the ABVOD scheme increases with the number of attributes ($n \in [1, 100]$) in access policy, while that of the other two schemes (namely LFGS and basic VFKSM) changes with the number of clauses ($v \in [1, 50]$) in access Boolean formula.[12] We analyze the performance of **Enc** in above three schemes by varying the number of encrypted files ($|\mathcal{F}|$) from 1 to 1000. For contrast, we set $n = 100$ and $v = 50$ in **Enc**. The experimental result shown in Fig. 7c indicates that the basic VFKSM has less ciphertexts encryption overhead than ABVOD and LFGS schemes, which has an approximately linear relationship with the value of $\mathcal{F}$. The similar conclusion regarding ciphertext storage overhead is also drawn from Fig. 7d, the only difference is that the basic VFKSM and LFGS have similar ciphertext length. We conclude that **Enc** in basic VFKSM is still efficient and feasible in practice while creating threshold signatures.

As the ABVOD scheme does not offer keyword search functionality, we just analyze the trapdoor generation time and storage costs of LFGS scheme, basic VFKSM and extended VFKSM in Figs. 7e and 7f, respectively. The performance of **Trap** in LFGS scheme increases with the number of each DU's attributes ($n' \in [1, 50]$) and the number of queried keywords ($z \in [1, 50]$), that of basic VFKSM is just influenced by the variable z, but that of extended VFKSM is almost impervious to these two variables. This is because

10. Compared with the basic VFKSM which supports single keyword search, the extended VFKSM can support the multi-keyword search, which will affect the performance of Trap and Search in actual tests. In here, we omit the theoretical analysis of extended VFKSM as the performance of **KeyGen**, **Enc**, **Verify** and **Dec** in the extended VFKSM is approximately similar to that of basic VFKSM.

11. http://www.cs.cmu.edu/∼enron/

12. If clauses are disjoint attribute sets, then $v < n$.

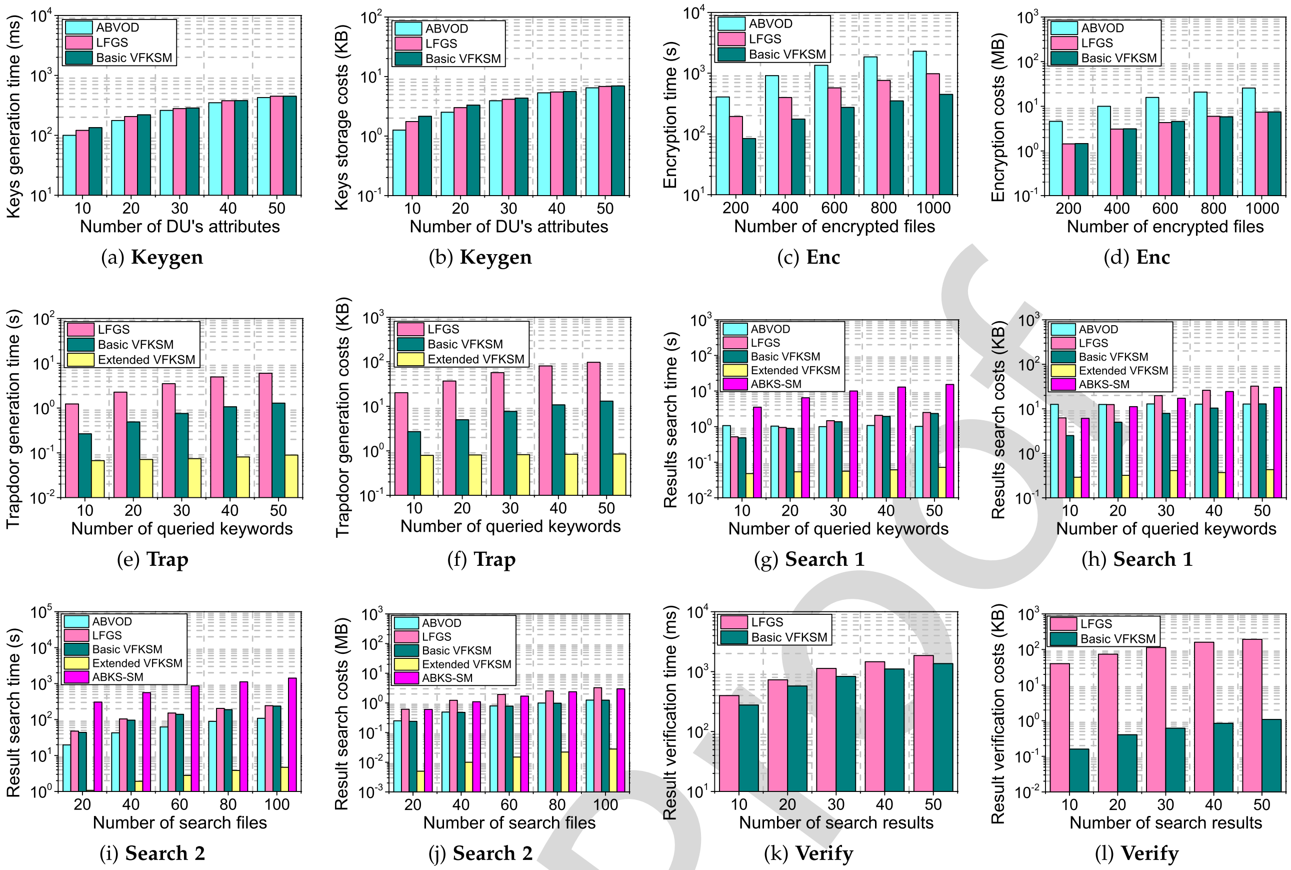


Fig. 7. Practical performance analysis in various algorithms.

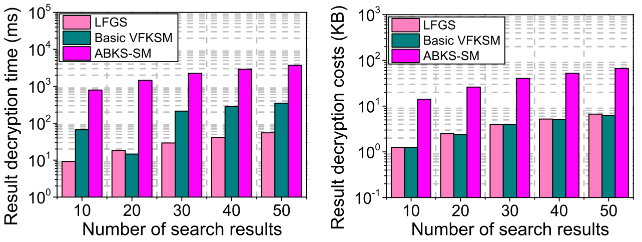
the LFGS scheme and basic VFKSM only support single keyword search, while the extended VFKSM is capable of supporting multi-keyword search and its trapdoor generation process is independent of the variable n'. Note that we set $n' = 50$ in Figs. 7e and 7f. Hence, the extended VFKSM has the best performance in terms of trapdoor generation when compared with the other two schemes, note that the basic VFKSM still outperforms the LFGS scheme as its trapdoor generation process is regardless of the variable n'.

In Figs. 7g, 7h, 7i and 7j, we take two factors (i.e., the number of queried keywords $z \in [1, 50]$, the number of search files $|\mathcal{F}'| \in [1, 100]$) into consideration, and set $n' = 50$. It is worth noticing that the performance of **Search** in the ABVOD scheme is affected by n', while that of the other three schemes is not affected by this variable, which comprises of two cases (performance of *Search 1*, performance of *Search 2*). In *Search 1*, we set $n' = 50$, $|\mathcal{F}'| = 1$, and analyze the performance of results search overhead (including results search time and search costs) in five schemes (namely ABVOD scheme, LFGS scheme, basic VFKSM, extended VFKSM and ABKS-SM [19]) by varying the value of z. In *Search 2*, we set $n' = z = 50$ and assess the performance of results search overhead by varying the value of $|\mathcal{F}'|$ from 1 to 100. As the ABVOD scheme cannot support keyword search, its results search time and cost for each encrypted file in *Search 1* do not depend on the variable z in Figs. 7g and 7h, and those of LFGS scheme, basic VFKSM and ABKS-SM scheme grow with increasing the value of z. However, the computation and storage overhead of results search in extended VFKSM is almost unaffected by variables n', z as the extended VFKSM can efficiently support multi-keyword search. In Figs. 7i and 7j, the value of z is set as 50 in *Search 2*, and the performance of results search in these schemes linearly grows with the number of search files. When executing ciphertext retrieval over each encrypted file, the ABVOD scheme, LFGS scheme, basic VFKSM and ABKS-UR scheme need about $(2n'+1)\mathbb{P} + n'\mathbb{E}_T$, $6z'\mathbb{P}$, $z'(3\mathbb{P}+3\mathbb{E})$ and $z((n'+1)\mathbb{P}+\mathbb{E}_T)$, respectively, but the extended VFKSM just needs $3\mathbb{P}+3\mathbb{E}$. Hence, the extended VFKSM is the most efficient with regard to results search when the variable $|\mathcal{F}'|$ is varied from 1 to 100. If the Pseudo-random permutation functions mentioned in the extended VFKSM scheme are

TABLE 4
Computation Time of Naive Checking Solution by Using the File/Keyword Hash Table

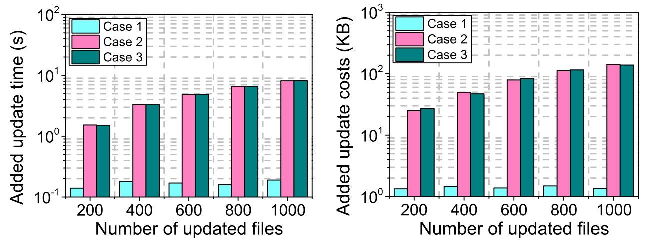
Values of $\lvert\mathcal{W}\rvert, \lvert\mathcal{F}\rvert$	$(10^3, 10^3)$	$(10^3, 10^4)$	$(10^4, 10^4)$,	$(10^4, 10^5)$	$(10^4, 10^6)$
Computation time (ms)	0.82	8.47	89.1	834	8297

Notes. *$(*, *)$ denote the values of variables $|\mathcal{W}|, |\mathcal{F}|$, respectively.*



(a) **Dec** (b) **Dec**

Fig. 8. The performance of **Dec** in various schemes.



(a) **DOs update** (b) **DOs update**

Fig. 9. The added update overhead in extended VFKSM.

utilized, the search complexity will be affected by the number of queried keywords and the number of search files rather than the size of dataset, thus our proposed schemes can be employed in the large-scale dataset if the number of queried keywords and the number of search files are small.

The ABVOD just checks whether CSP has honestly executed the outsourced decryption, but cannot check the correctness of search results. The LFGS scheme can accurately verify the correctness of search results, but this scheme still needs interactions between DO manager and DUs, which incurs high communication overhead if the number of returned results is large. Although the LFGS scheme can realize slightly efficient search result verification, it cannot be deployed in the multi-owner setting. Note that the search result verification mechanisms used in the LFGS scheme and other verifiable CP-ABKS schemes [2], [26], which just achieve private search result verification, are different from that used in our basic (or extended) VFKSM scheme. One naive result checking solution is to use the local file/keyword hash table, but this solution will result in much communication and storage overhead if the size of file/keyword hash table is large. Besides, the computation complexity of this naive checking solution is $\mathcal{O}(|\mathcal{W}||\mathcal{F}|)$, while that of the basic (or extended) VFKSM is $\mathcal{O}(\pi)$. We conduct a series of tests to show the computation time of naive checking solution in Table 4. As the size of file/keyword hash table is small, the naive checking solution is efficient than the search result verification mechanism in our basic (or extended) VFKSM, but this solution will incur high computation overhead and even storage overhead on DUs when the size of file/keyword hash table is large. However, the computation overhead of our search result verification mechanism based on the modified public auditing technique is not affected by variables $|\mathcal{F}|, |\mathcal{W}|$, which is just associated with the number of returned results (π), note that $\pi \ll |\mathcal{F}| \cdot |\mathcal{W}|$. Compared with above solutions, our basic (or extended) VFKSM can achieve accurate, public and non-interactive search result verification.

As the search result verification mechanisms in [2], [26] based on Bloom filters are related to multiple parameters (i.e., the number of authorized DUs, the number of extracted keywords, the number of hash functions used to insert DUs and keywords into Bloom filters, respectively, etc.), we just show the performance of **Verify** in the LFGS scheme and basic VFKSM schemes, note that the basic VFKSM and extended VFKSM have the same search result verification mechanism. From Fig. 8k and 8l, we notice that the computation and storage overhead of LFGS scheme and basic VFKSM grows with the number of search results (namely $\pi \in [1, 50]$), and the LFGS scheme is less efficient than our basic VFKSM in **Verify**. This is because the LFGS scheme needs to decrypt these search results before implementing search result verification. In Fig. 9a and 9b, we analyze the performance of **Dec** in the LFGS scheme, basic VFKSM and ABKS-SM scheme, which also increases with the number of search results. Although the ABKS-SM scheme can support the multi-owner setting, its result decryption time and decryption overhead are affected by the number of DOs and the number of search results. For comparison, we set the number of data owners as 10 (namely $d = 10$) in the decryption process. Our basic VFKSM is less efficient than the LFGS scheme as our basic VFKSM needs to execute one pairing operation to decrypt each search result, and the LFGS scheme needs one modular exponentiation operation. However, our basic VFKSM is still efficient than the ABKS-SM scheme as this scheme needs to interact with multiple DOs.

In contrast to the ABVOD and LFGS schemes, the basic VFKSM and extended VFKSM can be implemented in the

TABLE 5
An Example of Actual Tests in Various Schemes

Schemes	ABVOD [15]						LFGS [18]						Basic(Extended) VFKSM					
	Computation costs			Storage costs			Computation costs			Storage costs			Computation costs			Storage costs		
Datasets	Enron	NSF	RFC	Enron	NSF	RFC	Enron	NSF	RFC	Enron	NSF	RFC	Enron	NSF	RFC	Enron	NSF	RFC
KeyGen	427	413	446	6.47	6.14	7.46	449	431	463	6.82	6.74	7.17	451	429	469	6.93	6.76	7.02
Enc	2294	2203	2386	25.61	23.76	28.65	980	942	1073	7.38	6.92	7.45	447	421	458	7.48	7.29	7.61
Trap	—	—	—	—	—	—	5.97	5.84	6.02	97.61	96.78	98.12	1.29(0.09)	1.27(0.09)	1.30(0.10)	13.1(0.85)	12.36(0.81)	12.44(0.87)
Search 1	1.01	0.97	1.03	12.82	12.78	12.85	2.50	4.47	2.52	32.29	31.46	32.78	2.34(0.07)	2.31(0.06)	2.35(0.07)	12.92(0.48)	12.83(0.46)	13.12(0.49)
Search 2	108.73	107.35	109.11	1.24	1.24	1.26	242.84	241.01	244.38	3.25	3.19	3.48	236.00(4.72)	234.26(4.68)	238.11(4.75)	1.23(0.03)	1.22(0.03)	1.25(0.04)
Verify	—	—	—	—	—	—	1858	1794	1883	194	183	197	1358	1321	1453	1.08	1.02	1.19
Dec	—	—	—	—	—	—	55	48	59	6.67	6.61	6.72	349	341	363	6.25	5.89	6.41

Notes. *The units of computation and storage costs in each algorithm are defined as* $(*, *)$; *"—": Without consideration;* ***KeyGen***, ***Verify***, ***Dec***: *(ms, KB);* ***Trap***, ***Search 1***: *(s, KB);* ***Enc***, ***Search 2***: *(s, MB).*

multi-owner setting. Furthermore, the extended VFKSM supports dynamic multi-owner update such as adding or deleting a DO. Next, we analyze the added update overhead (including update time and update cost) of extended VFKSM compared with basic VFKSM in Figs. 9a and 9b. It is worth noticing that we take three cases into consideration, namely Case 1: adding a new DO and satisfying $d^* = d+1$, $2\eta - 1 > d^*$; Case 2: adding a new DO and satisfying $d^* = d+1$, $2\eta - 1 \leq d^*$; Case 3: revoking a DO. From Figs. 9a and 9b, we notice that the update overhead of Case 2 and Case 3 grows with the number of updated files (namely $|\mathcal{F}| \in [1, 1000]$), but that of Case 1 keeps almost unchanged. This because Case 1 just needs to update DO manager's secret/public key pair and distribute new share for each newly added DO, but does need to recompute each threshold signature like Case 2 and Case 3. This multi-owner update mechanism that avoids updating the whole ciphertexts is efficient in practice. For example, when updating 1000 files, the update overhead of Case 1, Case 2 and Case 3 is (0.19 s, 1.36 KB), (8.14 s, 141 KB), (8.16 s, 138 KB), respectively.

To proof the efficiency of our basic (or extended) VFKSM when being compared with ABVOD and LFGS schemes, we take some examples by assigning specific values for variables in various algorithms, note that we omit the ABKS-SM scheme as its performance is not better when compared with above schemes. Specifically, we set $n' = 50$ in **KeyGen**, $n = 100, v = 50, |\mathcal{F}| = 1000$ in **Enc**, $n' = 50, z = 50$ in **Trap**, $n' = 50, |\mathcal{F}'| = 1, z = 50$ in **Search 1**, $n' = 50, z = 50, |\mathcal{F}'| = 100$ in **Search 2**, $\pi = 50$ in both **Verify** and **Dec**. Furthermore, different datasets (i.e., Enron E-mail dataset, NSF dataset (National Science Foundation Research Awards Abstract 1990-2003 dataset),[13] RFC dataset (Request For Comments database)[14]) are employed to show the feasibility of basic VFKSM in actual scenarios. The results are given in Table 5, which are consistent with the theoretical analysis shown in Table 3. It is worth noticing that we also consider the performance of extended VFKSM in **Trap**, **Search 1** and **Search 2**.

7 CONCLUSION

Seeking to achieve short ciphertext length, fast search process, and authentic search result verification, we designed the basic VFKSM, where the ciphertext length in the encryption process and ciphertext transformation in the search process are independent of the number of attributes in the access structure. The basic VFKSM also allows the public search result verification mechanism to be deployed in the multi-owner setting, without the need for computationally expensive tasks to be performed on resource-limited devices. Furthermore, the basic VFKSM was extended to support multi-keyword search and multi-owner update. We also evaluated the security and performance of basic (or extended) VFKSM.

As previously discussed, one future research agenda is to extend VFKSM to be off-line (or inside) KGA-resilience. In addition, to ensure that the proposed VFKSM can be deployed on inexpensive devices (e.g., those with constrained computation and storage capabilities), one future research approach is to focus on expressive search queries including fuzzy or semantic keyword, and lightweight search result verification mechanism in the multi-owner setting.

13. http://kdd.ics.uci.edu/databases/nsfabs/nsfawards.html
14. http://www.ietf.org/rfc.html

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (No. 61702404, No. 61702105, No. U1804263, No. 61972094), the Fundamental Research Funds for the Central Universities (No. JB191506), the National Natural Science Foundation of Shaanxi Province (No. 2019JQ-005), the China Postdoctoral Science Foundation Funded Project (No. 2017M613080), the Singapore National Research Foundation under the NCR Award (No. NRF2018NCR-NSOE004-0001), and the AXA Research Fund.

REFERENCES

[1] Y. Miao, J. Ma, X. Liu, X. Li, Z. Liu, and H. Li, "Practical attribute-based multi-keyword search scheme in mobile crowdsourcing," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 3008–3018, Aug. 2018.
[2] W. Sun, S. Yu, W. Lou, Y. T. Hou, and 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.*, vol. 27, no. 4, pp. 1187–1198, Apr. 2016.
[3] Y. Miao, J. Ma, X. Liu, X. Li, Q. Jiang, and J. Zhang, "Attribute-based keyword search over hierarchical data in cloud computing," *IEEE Trans. Serv. Comput.*, pp. 1–14, 2017. doi: 10.1109/TSC.2017.2757467.
[4] Y. Miao, J. Ma, X. Liu, J. Weng, H. Li, and H. Li, "Lightweight fine-grained search over encrypted data in fog computing," *IEEE Trans. Serv. Comput.*, pp. 1–14, 2018. doi: 10.1109/TSC.2018.2823309.
[5] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. Int. Conf. Theory Appl. Cryptographic Tech.*, 2004, pp. 506–522.
[6] R. Chen, Y. Mu, G. Yang, F. Guo, and X. Wang, "Dual-server public-key encryption with keyword search for secure cloud storage," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 4, pp. 789–798, Apr. 2016.
[7] Y. Miao, J. Ma, X. Liu, J. Zhang, and Z. Liu, "Vkse-mo: Verifiable keyword search over encrypted data in multi-owner settings," *Sci. China Inf. Sci.*, vol. 60, no. 12, 2017, Art. no. 122105.
[8] Y. Zhang, C. Xu, H. Li, K. Yang, J. Zhou, and X. Lin, "Healthdep: An efficient and secure deduplication scheme for cloud-assisted ehealth systems," *IEEE Trans. Ind. Informat.*, vol. 14, no. 9, pp. 4101–4112, Sep. 2018.
[9] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy*, 2007, pp. 321–334.
[10] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Proc. Int. Workshop Public Key Cryptography*, 2011, pp. 53–70.
[11] Z. Wan, J. Liu, and R. H. Deng, "Hasbe: A hierarchical attribute-based solution for flexible and scalable access control in cloud computing," *IEEE Trans. Inf. Forensics Secur.*, vol. 7, no. 2, pp. 743–754, Apr. 2012.
[12] X. Fu, X. Nie, T. Wu, and F. Li, "Large universe attribute based access control with efficient decryption in cloud storage system," *J. Syst. Softw.*, vol. 135, pp. 157–164, 2018.
[13] M. Green, S. Hohenberger, B. Waters, et al., "Outsourcing the decryption of abe ciphertexts," in *Proc. USENIX Secur. Symp.*, 2011, vol. 2011, no. 3, pp. 1–16.
[14] J. Lai, R. H. Deng, C. Guan, and J. Weng, "Attribute-based encryption with verifiable outsourced decryption," *IEEE Trans. Inf. Forensics Secur.*, vol. 8, no. 8, pp. 1343–1354, Aug. 2013.
[15] B. Qin, R. H. Deng, S. Liu, and S. Ma, "Attribute-based encryption with efficient verifiable outsourced decryption," *IEEE Trans. Inf. Forensics Secur.*, vol. 10, no. 7, pp. 1384–1393, Jul. 2015.
[16] Q. M. Malluhi, A. Shikfa, and V. C. Trinh, "A ciphertext-policy attribute-based encryption scheme with optimized ciphertext size and fast decryption," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, 2017, pp. 230–240.

[17] Q. Chai and G. Gong, "Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers," in *Proc. IEEE Int. Conf. Commun.*, 2012, pp. 917–922.
[18] M. Cao, L. Wang, Z. Qin, and C. Lou, "A lightweight fine-grained search scheme over encrypted data in cloud-assisted wireless body area networks," *Wireless Commun. Mobile Comput.*, vol. 2019, 2019, Art. no. 9340808.
[19] Y. Miao, X. Liu, K.-K. R. Choo, R. H. Deng, J. Li, H. Li, and J. Ma, "Privacy-preserving attribute-based keyword search in shared multi-owner setting," *IEEE Trans. Dependable Secure Comput.*, pp. 1–15, 2019. doi: 10.1109/TDSC.2019.2897675.
[20] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2006, pp. 89–98.
[21] Z. Liu, Z. Cao, Q. Huang, D. S. Wong, and T. H. Yuen, "Fully secure multi-authority ciphertext-policy attribute-based encryption without random oracles," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2011, pp. 278–297.
[22] K. Yang, X. Jia, and K. Ren, "Secure and verifiable policy update outsourcing for big data access control in the cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 12, pp. 3461–3470, Dec. 2015.
[23] T. V. X. Phuong, G. Yang, and W. Susilo, "Hidden ciphertext policy attribute-based encryption under standard assumptions," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 1, pp. 35–45, Jan. 2016.
[24] W. Zhang, Y. Lin, S. Xiao, J. Wu, and S. Zhou, "Privacy preserving ranked multi-keyword search for multiple data owners in cloud computing," *IEEE Trans. Comput.*, vol. 65, no. 5, pp. 1566–1577, May 2016.
[25] Y. Miao, J. Ma, X. Liu, F. Wei, Z. Liu, and X. A. Wang, "m2-abks: Attribute-based multi-keyword search over encrypted personal health records in multi-owner setting," *J. Med. Syst.*, vol. 40, no. 11, 2016, Art. no. 246.
[26] Q. Zheng, S. Xu, and G. Ateniese, "Vabks: Verifiable attribute-based keyword search over outsourced encrypted data," in *Proc. IEEE Conf. Comput. Commun.*, 2014, pp. 522–530.
[27] Y. Miao, X. Liu, K.-K. R. Choo, R. H. Deng, H. Wu, and H. Li, "Fair and dynamic data sharing framework in cloud-assisted internet of everything," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 7201–7212, Aug. 2019.
[28] R. Chen, Y. Mu, G. Yang, F. Guo, X. Huang, X. Wang, and Y. Wang, "Server-aided public key encryption with keyword search," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 12, pp. 2833–2842, Dec. 2016.
[29] R. Chen, Y. Mu, G. Yang, F. Guo, and X. Wang, "A new general framework for secure public key encryption with keyword search," in *Proc. Australasian Conf. Inf. Secur. Privacy*, 2015, pp. 59–76.
[30] P. Xu, H. Jin, Q. Wu, and W. Wang, "Public-key encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack," *IEEE Trans. Comput.*, vol. 62, no. 11, pp. 2266–2277, Nov. 2013.
[31] Y. Miao, X. Liu, R. H. Deng, H. Wu, H. Li, J. Li, and D. Wu, "Hybrid keyword-field search with efficient key management for industrial internet of things," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3206–3217, Jun. 2018.
[32] P. Jiang, Y. Mu, F. Guo, X. Wang, and Q. Wen, "Online/offline ciphertext retrieval on resource constrained devices," *Comput. J.*, vol. 59, no. 7, pp. 955–969, 2016.
[33] H. S. Rhee, J. H. Park, W. Susilo, and D. H. Lee, "Improved searchable public key encryption with designated tester," in *Proc. Int. ACM Symp. Inf. Comput. Commun. Secur.*, 2009, pp. 376–379.
[34] A. Boldyreva, "Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme," in *Porc. Int. Workshop Public Key Cryptography*, 2003, pp. 31–46.
[35] H. Wang, Q. Wu, B. Qin, and J. Domingo-Ferrer, "Frr: Fair remote retrieval of outsourced private medical records in electronic health networks," *J. Biomed. Informat.*, vol. 50, pp. 226–233, 2014.
[36] B. Wang, B. Li, and H. Li, "Panda: Public auditing for shared data with efficient user revocation in the cloud," *IEEE Trans. Serv. Comput.*, vol. 8, no. 1, pp. 92–106, Jan./Feb. 2015.
[37] B. Wang, H. Li, X. Liu, X. Li, and F. Li, "Preserving identity privacy on multi-owner cloud data during public verification," *Secur. Commun. Netw.*, vol. 7, no. 11, pp. 2104–2113, 2014.
[38] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy*, 2000, pp. 44–55.
[39] S. Hu, C. Cai, Q. Wang, C. Wang, X. Luo, and K. Ren, "Searching an encrypted cloud meets blockchain: A decentralized, reliable and fair realization," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 792–800.
[40] X. Yang and C. Wang, "Threshold proxy re-signature schemes in the standard model," *Chinese J. Electron.*, vol. 19, no. 2, pp. 345–350, 2010.
[41] C. Gentry, "Practical identity-based encryption without random oracles," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Tech.*, 2006, pp. 445–464.
[42] S. S. Chow, J. K. Liu, and J. Zhou, "Identity-based online/offline key encapsulation and encryption," in *Proc. ACM Symp. Inf. Comput. Commun. Secur.*, 2011, pp. 52–60.
[43] H. Shacham and B. Waters, "Compact proofs of retrievability," *J. Cryptology*, vol. 26, no. 3, pp. 442–483, 2013.
[44] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Proc. Int. Conf. Theory Appl. Cryptology Inf. Secur.*, 2001, pp. 514–532.

Yinbin Miao (M'18) received the B.E. degree with the Department of Telecommunication Engineering from Jilin University, Changchun, China, in 2011, and Ph.D. degree with the Department of Telecommunication Engineering from Xidian University, Xi'an, China, in 2016. He is currently a Lecturer with the Department of Cyber Engineering in Xidian University, Xi'an, China. His research interests include information security and applied cryptography.

Robert H. Deng (F'16) is AXA Chair Professor of Cybersecurity and Professor of Information Systems in the School of Information Systems, Singapore Management University since 2004. His research interests include data security and privacy, multimedia security, network and system security. He served/is serving on the editorial boards of many international journals, including TIFS, TDSC. He has received the Distinguished Paper Award (NDSS 2012), Best Paper Award (CMS 2012), Best Journal Paper Award (IEEE Communications Society 2017). He is a fellow of the IEEE.

Kim-Kwang Raymond Choo (SM'15) received the Ph.D. in Information Security in 2006 from Queensland University of Technology, Australia. He currently holds the Cloud Technology Endowed Professorship at The University of Texas at San Antonio (UTSA). He is the recipient of various awards including the UTSA College of Business Col. Jean Piccione and Lt. Col. Philip Piccione Endowed Research Award for Tenured Faculty in 2018, ESORICS 2015 Best Paper Award. He is an Australian Computer Society Fellow, and an IEEE Senior Member.

Ximeng Liu (M'16) received the B.E. degree with the Department of Electronic Engineering from Xidian University, Xi'an, China, in 2010 and Ph. D. degree with the Department of Telecommunication Engineering from Xidian University, Xi'an, China in 2015. He is currently a post-doctoral fellow with the Department of Information System, Singapore Management University, Singapore. His research interests include applied cryptography and big data security.

Jianting Ning received the Ph.D. degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University in 2016. He is currently a research fellow at Department of Computer Science, National University of Singapore. His research interests include applied cryptography and information security, in particular, Public Key Encryption, Attribute-Based Encryption, and Secure Multi-party Computation.

Hongwei Li (SM'18) received the Ph.D. degree in computer software and theory from the University of Electronic Science and Technology of China, Chengdu, China, in 2008. He is currently a professor with the School of Computer Science and Engineering, University of Electronic Science and Technology of China. He has worked as a post-doctoral fellow in Department of Electrical and Computer Engineering from the University of Waterloo, Waterloo, ON, Canada, in 2012. His research interests include network security, applied cryptography, and trusted computing.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.