4-2018

Lightweight fine-grained search over encrypted data in fog computing

Yinbin MIAO

Jianfeng MA

Ximeng LIU
Singapore Management University, xmliu@smu.edu.sg

Jian WENG

Hongwei LI

See next page for additional authors

DOI: https://doi.org/10.1109/TSC.2018.2823309

Citation

MIAO, Yinbin; MA, Jianfeng; LIU, Ximeng; WENG, Jian; LI, Hongwei; and Li, Hui. Lightweight fine-grained search over encrypted data in fog computing. (2018). *IEEE Transactions on Services Computing*. Research Collection School Of Information Systems.
Available at: https://ink.library.smu.edu.sg/sis_research/3978

Author
Yinbin MIAO, Jianfeng MA, Ximeng LIU, Jian WENG, Hongwei LI, and Hui Li

Published in IEEE Transactions on
Services Computing, 2018 April,
http://doi.org/10.1109/TSC.2018.2823309

Lightweight Fine-Grained Search over Encrypted Data in Fog Computing

Yinbin Miao, Jianfeng Ma, Ximeng Liu, Jian Weng, Hongwei Li, and Hui Li

Abstract—Fog computing, as an extension of cloud computing, outsources the encrypted sensitive data to multiple fog nodes on the edge of Internet of Things (IoT) to decrease latency and network congestion. However, the existing ciphertext retrieval schemes rarely focus on the fog computing environment and most of them still impose high computational and storage overhead on resource-limited end users. In this paper, we first present a Lightweight Fine-Grained ciphertexts Search (LFGS) system in fog computing by extending Ciphertext-Policy Attribute-Based Encryption (CP-ABE) and Searchable Encryption (SE) technologies, which can achieve fine-grained access control and keyword search simultaneously. The LFGS can shift partial computational and storage overhead from end users to chosen fog nodes. Furthermore, the basic LFGS system is improved to support conjunctive keyword search and attribute update to avoid returning irrelevant search results and illegal accesses. The formal security analysis shows that the LFGS system can resist Chosen-Keyword Attack (CKA) and Chosen-Plaintext Attack (CPA), and the simulation using a real-world dataset demonstrates that the LFGS system is efficient and feasible in practice.

Index Terms—Fog computing, attribute-based encryption, searchable encryption, conjunctive keyword search, attribute update.

1 Introduction

The promising cloud computing [1] paradigm can provide on-demand services with elastic resources and enable cloud clients to relieve the high storage and computation costs [2] locally. However, the prevalence of Internet of Things (IoT) applications [3] poses a huge challenge to the centralized cloud computing paradigm which incurs unbearable transmission latency and degraded services between user requests and cloud responses. Besides, large amounts of data generated from the IoT applications are often stored in the cloud. To decrease latency and network congestion, a fog computing paradigm [4] which is an extension of cloud computing services to network edge has been a relatively recent research topic. In fog computing, the fog nodes inserted into the middle of cloud and end users (or IoT devices) can provide various services (i.e., data computation, data storage, etc.) for resource-limited end users (i.e., sensor nodes, mobile terminals, etc.), note that fog nodes are much closer to end users than cloud, which is shown in Fig. 1. When sensitive data (i.e., text, image, video, etc.) [5], [6], [7] are outsourced to honest-but-curious fog nodes which are similar to public cloud platform, the data security and privacy concerns [8] still impede the adoption of fog computing as data owners lose the physical control over their data in fog nodes or cloud.

- Y. Miao, J. Ma and H. Li (Hui Li) are with the Department of Cyber Engineering, Xidian University, Xi'an 710071, China. E-mail: ybmiao@xidian.edu.cn, jfma@mail.xidian.edu.cn, lihui@mail.xidian.edu.cn, xhli1@mail.xidian.edu.cn.
- X. Liu is with the Department of Information Systems, Singapore Management University, 80 Stamford Road, Singapore. Email: xmliu@smu.edu.sg.
- J. Weng is with the College of Information Science and Technology, College of Cyber Security, Jinan University, Guangzhou 510632, China. Email: cryptjweng@gmail.com.
- H. Li (Hongwei Li) is with the Department of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610051, China. Email: hongweili@uestc.edu.cn.

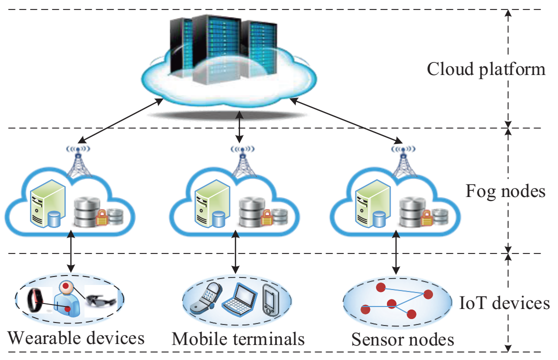

Fig. 1. The infrastructure of fog computing.

To mitigate the data privacy leakage risks, data encryption is an efficient mechanism to protect data confidentiality, but it makes the information retrieval over encrypted data extremely difficult. Moreover, the encrypted data should be amenable to access control. For example, Identity-Based Encryption (IBE) [9], [10] and Attribute-Based Encryption (ABE) [11], [12] can protect data security by providing coarse-grained and fine-grained access control mechanisms, respectively. In addition to data security concerns, achieving effective keyword search over encrypted data[1] and fine-grained access control are also the vital features in actual scenarios. Searchable Encryption (SE) technology [13], [14], which enables data users to securely search and selectively retrieve records of interest over encrypted data according to user-specified keywords, has been extensively explored. To further furnish fine-grained access control in the preceding SE solutions, the promising Ciphertext-Policy Attribute-Based Keyword Search (CP-ABKS) [15], [16] has gained much attention in both industrial and academic fields. In

1. This paper focuses on textual data.

CP-ABKS schemes, a certain end user can decrypt ciphertexts of interest if and only if his attribute set satisfies the access policy embedded into ciphertexts and his submitted trapdoor matches the indexes simultaneously.

Although CP-ABKS is a most useful cryptographic tool to achieve both fine-grained access control and keyword search functionalities, the computational and storage costs of existing CP-ABKS schemes are approximately proportional to the complexity of access policy, which greatly hinder the uses of resource-limited IoT devices. Hence, it is critical to keep operations on all end users lightweight in practice. When we take the cloud-fog-user architecture in fog computing into consideration, each fog node in fog computing can be treated as a proxy to conduct partial computation in place of IoT devices, which leaves less computation for the IoT devices to generate trapdoor and decrypt relevant ciphertexts.

Based on CP-ABKS scheme, we first devise a basic **L**ightweight **F**ine-**G**rained **S**earch (**LFGS**) over encrypted data system in fog computing in order to achieve keyword search over encrypted data and fine-grained access control in multiple end users setting as well as avoid latency and network congestion in traditional cloud computing paradigm. In addition to utilizing CP-ABKS technique, LFGS system also lightens the computational and storage burden of end users by cooperating with fog nodes. However, in dynamic applications, the roles of end users may change. As a result, the malicious end user can access the unauthorized ciphertexts by exploiting his outdated secret key. Furthermore, the single keyword search will return many irrelevant search results and then incur the waste of computation and bandwidth resources. As a further contribution, we extend the basic LFGS system to support attribute update [17], [18] and conjunctive keyword search [19], [20]. Thus, the extended LFGS can not only support the fine-grained keyword search (including single keyword search and conjunctive keyword search) and attribute update but also significantly reduce the end users computational burden with the help of fog nodes. To prevent the file and trapdoor privacy from being eavesdropped by some attacks (i.e., replay attack [21] and man-in-the-middle attack [22], etc.) and show the practicality of LFGS system in actual scenarios, we give the formal security analysis and comprehensive performance analysis. Specifically, the main contributions of our paper are shown as follows:

- ***Fine-grained keyword search***. LFGS system gains one-to-many rather than one-to-one encryption and specifies flexible access control over shared data, which eliminates some inherent drawbacks of existing public key cryptography schemes. Besides, LFGS system enables end users to search ciphertexts of interest according to queried keyword.
- ***Lightweight computation on end users***. With the help of fog nodes, LFGS system reliefs the large computational burden from data owners or end users, which means that partial computation including files encryption, trapdoor generation and ciphertexts decryption is offloaded to fog nodes without any loss of data confidentiality.
- ***Attribute update***. The extended LFGS system supports attribute update and just needs to update the keys and ciphertexts associated with the updated attributes, which not only avoids illegal accesses using outdated keys but also imposes less computational overhead on this system.
- ***Conjunctive keyword search***. The extended LFGS system allows end users to issue multiple keywords in a single search query so that it can improve the user search experience as the conjunctive keyword search can narrow down the search scope and quickly locate the results of interest.
- ***Security and practicability***. Formal security shows that LFGS system is selectively secure against Chosen-Keyword Attack (CKA) and Chosen Plaintext Attack (CPA). The extensive experiments demonstrate that LFGS system is efficient and feasible in a broad range of applications.

The remainder of our paper is structured as follows. Section 2 reviews some previous work associated with LFGS system. Section 3 gives some preliminaries used as the basis of LFGS system. In section 4, we first present the problem formulations including system model, overview of LFGS system and security model. Then, we demonstrate the concrete construction of LFGS system in section 5 and analyze the security and performance in section 6. Finally, we draw a conclusion remark in section 7.

2 Related Work

In cloud computing environment, SE provides a fundamental solution to issue search queries over encrypted data according to specified keywords. Song et al. [13] gave the first SE scheme which required litter communication, but the computational overhead was linear in the size of search query. To tackle this problem, Boneh et al. [14] presented a Public key Encryption with Keyword Search (PEKS) scheme. After that, many SE schemes enriched with different features had been proposed, such as single keyword search, multiple keywords search, fuzzy keyword search, verifiable keyword search, ranked keyword search. Compared with single keyword search [23], [24], multiple keyword search [25], [26] can quickly locate the results of interest and greatly decrease the waste of computation and bandwidth resources. As there is no tolerance of minor types and format inconsistencies in exact keyword search queries, fuzzy keyword search [27] enhances the system usability by leveraging Wildcard, Locality-Sensitive Hashing (LSH) or Bloom Filter (BL) techniques. As the cloud is a semi-trusted third-party which will execute a fraction of search operations and return a fraction of search results to save computation resources or hide data corruption accidents, verifiable keyword search [28] can verify whether the results are correct or not. To further narrow down the search results, ranked keyword search [29], [30] can return the results in the order of relevance scores to keywords.

However, the aforementioned SE schemes cannot enable data owner to grant fine-grained search capabilities to end users. To achieve fine-grained access control, two categories of ABE schemes, namely Key-Policy ABE (KP-ABE) [11] and Ciphertext-Policy ABE (CP-ABE) [12] were proposed. Besides, the cloud computing mainly provides resources distributed in the core network far from end users, which will cause high network delay and congestion. Then, fog computing, which is not a replacement but an extension of cloud computing for the emergence of IoT applications, extends the cloud elastic resources to the edge of the network (i.e., portable devices, wireless sensors and other IoT devices, etc.) with limited storage and processing capacities. To fill the gap between cloud computing and fog computing, Zuo et al. [31] proposed a practical CP-ABE scheme with outsourced decryption in fog computing environment, but this scheme still suffered from key-delegation abuse issue. Along this direction, Jiang et al. [32] presented a traceable CP-ABE scheme which can provide protection against key exposure. However, the attribute update in existing CP-ABE schemes is still an important issue when the roles of end users are changed. To this end, Zhang et al. [18] proposed an efficient access control CP-ABE scheme with outsourcing capability and attribute update for fog computing.

Unfortunately, one of the main efficiency defects in existing CP-ABE schemes is that encryption or decryption operations involve time-consuming pairing operations and a number of other operations increasing with the complexity of access policy. Hence, it is critical to largely eliminate computational costs for resource-limited end users in fine-grained keyword search system. For example, Lai et al. [33] proposed a verifiable decryption scheme which incurs less computational overhead for end users to recover the transformed ciphertexts; Malluhi et al. [34] showed an improved CP-ABE scheme with optimized ciphertext size and fast decryption, which can be applied in the context of lightweight IoT devices. However, all aforementioned CP-ABE schemes still cannot support keyword search.

Motivated by this issue, Zheng et al. [15] showed two Attribute-Based Keyword Search (ABKS) schemes (i.e., Key-Policy ABKS (KP-ABKS) and Ciphertext-Policy ABKS (CP-ABKS)) by using KP-ABE and CP-ABE, respectively. Note that CP-ABKS [35], [17] has been considered as the practical cryptographic primitive for the provision of fine-grained access control over ciphertexts. Then, Sun et al. [16] demonstrated an owner-enforced CP-ABKS scheme in a challenging multi-contributor scenario by exploiting proxy re-encryption and lazy encryption techniques. Although the two CP-ABKS schemes outlined above can check the correctness of search results, these schemes have high false-positive rate caused by BL. Then, Fan et al. [36] gave a verifiable CP-ABKS scheme which accurately verified the correctness of search results by attaching a signature to each file, whereas this scheme still cannot be applied in fog computing environment.

Although the latest CP-ABKS scheme [37] can achieve the fine-grained search and access authorization in fog computing, its computational and storage costs increase linearly with the number of system attributes rather than the number of required attributes. To achieve practicability and feasibility in fog computing, we propose a **L**ightweight **F**ine-**G**rained **S**earch (LFGS) system which supports fine-grained access control and single keyword search.

3 Preliminaries

Here, we first present some cryptographic backgrounds as the basis of LFGS system, which includes access structure, access tree and Decisional Bilinear Diffie-Hellman (DBDH) assumption. Let G, G_T be two multiplicative cyclic groups of prime order p, g be a generator of group G and e be the bilinear map $G \times G \rightarrow G_T$ with several properties: (1) Bilinearlity. $e(g^a, g^b) = e(g,g)^{ab}$, $\forall a, b \in \mathcal{Z}_p$; (2) Non-degeneracy: $e(g,g) \neq 1$; (3) Computability. e can be efficiently computed. Given a set X, the symbol $x \in X$ is defined as choosing an element x uniformly at random from the set X, and $[1, y]$ denotes an integer set $\{1, 2, ..., y\}$. Besides, given a set S, the Lagrange coefficient is defined as $\Delta_{i,S}(0) = \prod_{j \in S, j \neq i} \frac{0-j}{i-j}$.

3.1 Access Structure

Given a set of parties $\mathcal{P} = \{P_1, P_2, \cdots, P_n\}$, the collection $\mathbb{A} \subseteq 2^{\{P_1, P_2, \cdots, P_n\}}$ is monotonic if the following condition holds. Namely, given two arbitrary sets B, C, we can gain $C \in \mathbb{A}$ if $B \in \mathbb{A}$ and $B \subseteq C$. An monotonic access structure is a collection $\mathbb{A}$ with non-empty subsets of $\mathcal{P}$, such as $\mathbb{A} \subseteq 2^{\{P_1, P_2, \cdots, P_n\}} \backslash \emptyset$. Notice that sets in $\mathbb{A}$ are called authorized entities, but the ones which do not belong to $\mathbb{A}$ are called unauthorized entities. In this paper, $\mathbb{A}$ contains all authorized attribute sets. Unless stated otherwise, the access structure outlined above represents a monotonic one.

3.2 Access Tree

Assume that $\mathcal{T}$ is an access tree used to describe access policy, and its each non-leaf node w denotes a threshold gate represented by its children nodes and threshold value. Assume that the number of children of a certain node w is num_w and its threshold value is k_w, then we have $0 < k_w \leq num_w$. Especially, the threshold gate is "AND" gate when $k_w = num_w$, and it is "OR" gate when $k_w = 1$. Notice that each leaf node w of $\mathcal{T}$ denotes an attribute and its threshold value is $k_w = 1$. To further facilitate description, some functions are shown as follows:

- $parent(w)$: The parent of non-root node w in $\mathcal{T}$;
- $att(w)$: The attribute value associated with the leaf-node w in $\mathcal{T}$;
- $index(w)$: The specified order of children of non-leaf node w in $\mathcal{T}$, which ranges from 1 to num_w;
- $\mathcal{T}_w$: The subtree of $\mathcal{T}$ rooted at the non-root node w;
- $\mathcal{T}_{root}$: The access tree rooted at the root node $root$ in $\mathcal{T}$.

Given an attribute set γ, if it matches the subtree $\mathcal{T}_w$, then we set $\mathcal{T}_w(\gamma) = 1$ which can be calculated recursively with the following steps: (1) if w is a non-leaf node, we can compute each children w' of node w, where $\mathcal{T}_w(\gamma) = 1$ holds if and only if at least k_w children return $\mathcal{T}_{w'}(\gamma) = 1$; (2) if w is leaf node, then we can gain $\mathcal{T}_w(\gamma) = 1$ if and only if $att(w) \in \gamma$.

3.3 Decisional Bilinear Diffie-Hellman Assumption

Given the bilinear map parameters (G, G_T, p, e, g) and three elements $(a', b', c') \in_R \mathcal{Z}_p^3$, if there is no PPT (Probabilistic Polynomial Time) algorithm $\mathcal{B}$ that can distinguish between the tuple $(g, g^{a'}, g^{b'}, g^{c'}, e(g,g)^{a'b'c'})$ and the tuple $(g, g^{a'}, g^{b'}, g^{c'}, \vartheta)$, we can say that the PPT algorithm $\mathcal{B}$ does not have an advantage ϵ in solving the DBDH problem when the following equation holds, where $\vartheta \in_R G_T$.

$$\left| \begin{array}{l} Pr[\mathcal{B}(g, g^{a'}, g^{b'}, g^{c'}, e(g,g)^{a'b'c'}) = 1] \\ - Pr[\mathcal{B}(g, g^{a'}, g^{b'}, g^{c'}, \vartheta) = 1] \end{array} \right| < \epsilon. \quad (1)$$

Definition 1. We say that the DBDH assumption holds in G if no PPT algorithm has a non-negligible advantage ϵ in solving the DBDH problem.

4 Problem Formulation

In this section, we give the system model, threat model, algorithms overview of LFGS system and security model, respectively.

4.1 System Model & Threat Model

In this paper we consider a ciphertexts retrieval scenario in fog computing, which mainly involves five entities, namely Key Generation Center (**KGC**), Cloud Service Provider (**CSP**), multiple Fog Nodes (**FNs**), Data Owner (**DO**) and End Users (**EUs**), which are shown in Fig. 2. It is worth noticing that the communication synchronization between FNs and CSP can be gained by a fully-trusted third-party certificate center, and DO (or EUs) and FN are synchronized via a secure channel [38], such as SSL (Secure Socket Layer) and TLS (Transport Layer Security). The specific role of each entity is given as follows:

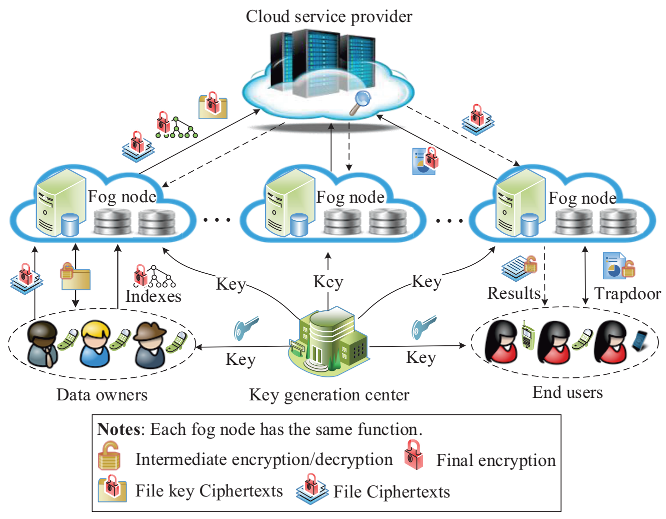

Fig. 2. The infrastructure of fog computing.

- Key generator center (KGC): KGC is responsible for generating system parameters and distributing secret keys to FNs and EUs, respectively. Besides, when EUs' attributes have been updated, KGC is able to update partial secret keys to avoid unauthorized access controls.
- Data owner (DO): Before outsourcing the final ciphertexts[2] to CSP, the DO needs to output file key ciphertexts and build encrypted indexes according to keyword set with cooperating with the chosen FN. Finally, the DO sends final ciphertexts to CSP via a certain FN.
- Cloud service provider (CSP): CSP has almost unlimited computation and storage capacities to undertake file remote storage tasks and conduct ciphertexts retrieval operations.
- End users (EUs): The resource-limited EUs can issue search queries according to trapdoor generated with the help of FNs. Moreover, EUs need to decrypt final ciphertexts returned by FNs.
- Fog nodes (FNs): FNs can not only generate the final ciphertexts but also output the final trapdoor on behalf of DO and EUs, respectively. Furthermore, FNs can partially decrypt returned ciphertexts to further cut down EUs' computational costs.

In this paper, we assume that KGC is a fully trusted entity, while the CSP and FNs are honest-but-curious third-parties which honestly execute the pre-defined protocols but are curious to deduce sensitive information from stored ciphertexts and trapdoors, which can help them acquire additional information. Depending on what information the CSP and FNs know, we adopt the following two threat models [29]. The malicious EUs may collude with each other to access unauthorized ciphertexts, whereas EUs cannot collude with FNs in LFGS system.

- *Known ciphertext model*. In this model, the CSP and FNs can obtain the encrypted files, indexes and trapdoors.
- *Known background model*. In this stronger model, the CSP and FNs are supposed to possess more knowledge (i.e., correlation relationship of given trapdoors, dataset related statistical information, etc.) than that in the known ciphertext model.

4.2 Overview of LFGS System

LFGS system is a tuple of several algorithms, namely **Setup**, **KeyGen**, **Enc**, **Trap**, **Search** and **Dec**, which is shown in Fig. 3.

Besides, we give a general description for the LFGS system in Fig. 4. In the step ①, the DO first sends the access policy to the chosen FN, then the FN returns the encrypted access policy described by a access tree to DO, finally the DO outputs the final ciphertexts (i.e., file content ciphertexts, file key ciphertexts and index ciphertexts, etc.) and returns to CSP via the chosen FN. When the EU wants to issue the search query, the KGC generates the secret keys for the EU and his chosen FN according to his attributes with the step ②. After gaining his secret key, the EU sends his attributes to the chosen FN, and the FN outputs the intermediate trapdoor (or search token) on condition that the EU is a legal entity; then, the EU further generates the intermediate trapdoor and sends it to the FN; at last, the FN outputs the final trapdoor and returns it to CSP, which is demonstrated by the step ③. In the step ④, the CSP issues the search operation and returns the relevant search results to the FN. After gaining the search results, the FN first conducts the majority of operations and then returns the

2. The ciphertexts include file ciphertexts, file key ciphertexts and index ciphertexts, where file ciphertexts are achieved with symmetric encryption algorithm(i.e., AES, DES, etc.), both file key and index ciphertexts are gained with CP-ABE mechanism.

LFGS system definition

The overview of LFGS system is presented as follows:

- **Setup**(1^k): Given the security parameters k and attribute set $\mathcal{U}$, KGC outputs the public key PK and master key MSK.
- **KeyGen**(PK, MSK, S): Given the attributes S, KGC generates the public/secret pairs (PK_{FN}, SK_{FN}), (PK_{EU}, SK_{EU}) for FNs and EUs, respectively.
- **Enc**$(PK, \mathcal{W}, \mathcal{F}, \mathcal{T}, \Gamma)$: Given the file set $\mathcal{F}$, keyword set $\mathcal{W}$ and access policy Γ, a certain DO generates the file content ciphertexts $\{c_\tau\}$, file key ciphertexts $\{CT_\tau^*\}$ and encrypted indexes $\{I_\tau\}$ with the help of a chosen FN and outsourced ciphertexts to CSP via the FN.
- **Trap**$(W', S, SK_{FN}, SK_{EU}, PK_{FN}^*, PK_{EU}^*)$: When a certain EU wants to issue search query according to his submitted keyword W', the EU outputs the final trapdoor $T_{W'}$ by outsourcing the partial computation to the chosen FN and sends $T_{W'}$ to CSP via the FN.
- **Search**$(\{c_\tau\}, \{CT_\tau^*\}, I, S, T_{W'}, \Gamma)$: After gaining attributes S and trapdoor $T_{W'}$, the CSP conducts the search operation and returns the relevant search results $\{c'_\pi\}$ on the condition that the EU is a legal entity.
- **Dec**$(\{c'_\pi\}, \{CT'_\pi\}, SK_{FN}, SK_{EU}, PK, \mathcal{T}, \Gamma)$: After receiving the search results $\{c'_\pi\}$, the chosen FN issues partial decryption operations before the EU can gain the final plaintext.

Fig. 3. Overview of LFGS system

intermediate results to the EU, finally the EU decrypts the ciphertexts without high computational and storage burden, which is shown by the step ⑤. As for the specific interacting processes in different algorithms in LFGS system, we will give a detailed introduction in Section 5.

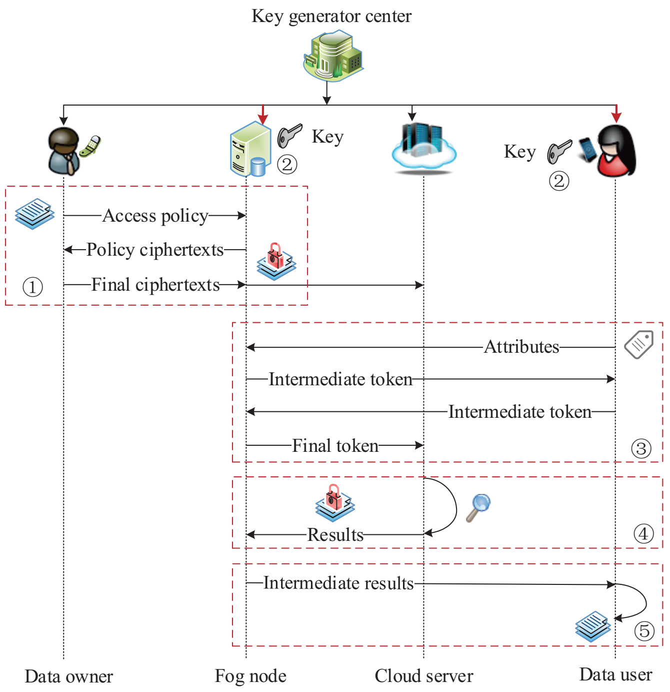

Fig. 4. Framework of LFGS system.

4.3 Security Model

We now present the chosen plaintext security of LFGS system in fog computing environment. Assume that there exists a Probabilistic Polynomial Time (PPT) adversary $\mathcal{A}$ which can send a challenge access structure Γ^* to the challenger $\mathcal{C}$ in the following security game.

- *Initialization*. $\mathcal{A}$ chooses a challenging access structure Γ^* and then sends Γ^* to $\mathcal{C}$.
- *Setup*. $\mathcal{C}$ conducts the **Setup** algorithm and sends the public parameters PP to $\mathcal{A}$.
- *Phase 1*. To issue secret key query for an attribute set S, $\mathcal{A}$ can adaptively sends any S to $\mathcal{C}$, but the restriction is that all submitted attribute sets cannot satisfy Γ^*. With regard to each attribute set S, $\mathcal{C}$ performs the **KeyGen** algorithm to output the final secret key SK and then sends SK to $\mathcal{A}$.
- *Challenge*. $\mathcal{A}$ picks two messages m_0, m_1 with equal length before sending them to $\mathcal{C}$, then $\mathcal{C}$ selects a random bit $\kappa \in \{0, 1\}$ and then calls the **Enc** algorithm to encrypt m_κ. Finally, $\mathcal{C}$ returns the challenging ciphertexts C^* to $\mathcal{A}$.
- *Phase 2*. $\mathcal{A}$ is allowed to issue secret key queries for other attribute sets, but there still exists a restriction that none of the aforementioned secret keys can decrypt the challenging ciphertexts C^*.
- *Guess*. $\mathcal{A}$ returns a guess bit $\kappa^* \in \{0, 1\}$. If $\kappa^* = \kappa$, $\mathcal{A}$ wins the security game; otherwise, $\mathcal{A}$ fails. The $\mathcal{A}$'s advantage in breaking the above security game is defined as $Adv_{\mathcal{A}}(1^k) = |Pr[\kappa^* = \kappa] - \frac{1}{2}| < \epsilon$.

Definition 2. LFGS system can achieve CPA security if there exist no PPT adversaries which can break the above security game with a non-negligible advantage ϵ under the DBDH assumption.

Besides, LFGS system can achieve selectively secure against CKA in the following security game between $\mathcal{A}$ and $\mathcal{C}$.

- *Initialization*. $\mathcal{A}$ first selects a challenging access structure Γ^* and then sends Γ^* to $\mathcal{C}$.
- *Setup*. $\mathcal{C}$ first performs the **Setup** algorithm and then sends the public parameters PP to $\mathcal{A}$.
- *Phase 1*. $\mathcal{A}$ first selects any attribute set S and keyword W, then $\mathcal{A}$ repeatedly issues the following three trapdoor generation queries.

- **Trap-I**(W, S): $\mathcal{A}$ submits the tuple (W, S) to a certain FN, then the FN returns a intermediate trapdoor value $T_{W,1}$.
- **Trap-II**(W, S): $\mathcal{A}$ submits the tuple (W, S) to a certain EU, then the EU returns a intermediate trapdoor value $T_{W,2}$.
- **Trap-III**(W, S): $\mathcal{A}$ asks for the final trapdoor T_W by submitting a tuple (W, S) to the FN chosen in **Trap-I**(W, S).

- *Challenge*. $\mathcal{A}$ first submits two challenging keywords W_0, W_1 with the same length, then $\mathcal{C}$ chooses a random bit $\kappa \in \{0, 1\}$ and calls the **Enc** algorithm to generate the index I_{W_κ} with Γ^*. Finally, $\mathcal{C}$ sends I_{W_κ} to $\mathcal{A}$.
- *Phase 2*. $\mathcal{A}$ repeatedly the process in *Phase 1*, but the restriction is that the two keywords W_0, W_1 cannot be queried in *Phase 2*.
- *Guess*. $\mathcal{A}$ returns a guess bit $\kappa^* \in \{0, 1\}$. If $\kappa^* = \kappa$, $\mathcal{A}$ wins the security game; otherwise, $\mathcal{A}$ fails. The $\mathcal{A}$'s advantage in breaking the above security game is defined as $Adv_{\mathcal{A}}(1^k) = |Pr[\kappa^* = \kappa] - \frac{1}{2}| < \epsilon$.

Definition 3. LFGS system is selectively secure against CKA if there exist no PPT adversaries which can break the above security game with a non-negligible advantage ϵ under the DBDH assumption.

5 Construction of LFGS System

Before presenting the concrete construction of LFGS system in fog computing environment, we first show some notations used in our construction in TABLE 1. With low latency and mobility provided by fog computing, resources-limited EUs and DO can outsource partial computational burden to FNs, leaving a fraction of operations to be conducted by themselves. Besides, LFGS system gains fine-grained owner-enforced access authorizations over encrypted files. Compared with prior CP-ABKS schemes, LFGS system can not only achieve fine-grained access control but also alleviate computational burden on EUs by adding a middle layer called FN. Before presenting the detailed construction, we first give some notation definitions used in LFGS system in TABLE. For simplicity, we assume that there are n attributes $\mathcal{U} = \{att_1, \cdots, att_n\}$ in LFGS system. In the following, we show the main algorithms in LFGS system, namely **Setup**, **KeyGen**, **Enc**, **Trap**, **Search**, **Dec**.

TABLE 1
Notation descriptions in LFGS system

Notations	Descriptions
$\mathcal{U} = \{att_1, \cdots, att_n\}$	System attribute set
PK_{FN}, PK_{EU}	Public keys of FNs and EUs
$S = \{att_j^*\}$	EU's attribute set
$(K_1, K_2, K_3, r, \{K_{j,2}, K_{j,3}\})$	FN's secret key
$(K_0, u, \{K_{j,1}\})$	EU's secret key
$\mathcal{F} = \{F_1, \cdots, F_m\}$	File set
$\mathcal{W} = \{W\}$	Keyword set
$k_\tau(\tau \in [1, m])$	Encryption key of F_τ
$CT_\tau = (C_1, C_2, \{C_l\}_{l\in\mathcal{L}})$	Intermediate ciphertext of k_τ
$CT_\tau^* = (CC_\tau, C', C_1', C_2', \{C_l\})$	Final ciphertext of k_τ
$I_\tau = (I_0, I_1, \{I_{l,1}, I_{l,2}\}_{l\in\mathcal{L}})$	Index of F_τ
$T_{W'} = (T_0', T_1', T_{EU}^*, \{T_{j,1}', T_{j,2}'\})$	Trapdoor for keyword W'

Setup(1^k): Given the security parameter k and public bilinear parameters $\mathcal{PP} = (G, G_T, e)$, the algorithm issued by KGC chooses two generators $g_0, g_1 \in G$, two elements $x, y \in_R \mathcal{Z}_p^2$ and a hash function $H_1 : \{0, 1\}^* \to_R \mathcal{Z}_p^*$ which maps any string to a random element in $\mathcal{Z}_p$; then, KGC selects $t_i \in_R \mathcal{Z}_p^*$ for each attribute $att_i \in \mathcal{U}(i \in [1, n])$ and computes $A_i = g_0^{t_i}$, $B = g_0^x$, $Y = e(g_0, g_0)^y$; finally, KGC returns the public key PK and master key MSK with Eq. 2.

$$\begin{aligned} PK &= (\mathcal{PP}, g_0, g_1, B, Y, H_1, \{A_i\}); \\ MSK &= (x, y, \{t_i\}). \end{aligned} \tag{2}$$

KeyGen(PK, MSK, S): When a certain FN joins into LFGS system, KGC first chooses an element $r \in_R \mathcal{Z}_p^*$ and outputs the FN's public key $PK_{FN} = Y^r = e(g_0, g_0)^{yr}$; then the FN is chosen as an intermediate party with a database-associated public key $PK_{FN}^* = PK_{FN}^{-s} = e(g_0, g_0)^{-syr}$ and authorized EU list (UL), where $s \in_R \mathcal{Z}_p^*$ can be shared by each outsourced file in LFGS system. When each EU in UL joins the LFGS system, the EU submits his certification associated with attribute set $S = \{att_j^*\}$ to ask for secret key. Similar to the authorized FN, KGC also selects an element $u \in_R \mathcal{Z}_p^*$ and computes $PK_{EU} = Y^u = e(g_0, g_0)^{yu}$ as the EU's public key, then KGC returns the EU's public key $PK_{EU}^* = PK_{EU}^{-s} = e(g_0, g_0)^{-syu}$ associated with the dataset specified by the DO.

With regard to secret key generation, the KGC first chooses $a_j, b_j \in_R \mathcal{Z}_p^2$ for each attribute $att_j^* \in S$ and $v, z \in_R \mathcal{Z}_p^*$. Then, KGC computes $K_{j,1} = g_0^{b_j/t_{\rho_1(j)}}$, $K_{j,2} = g_0^{(a_j-b_j)/t_{\rho_1(j)}}$, $K_{j,3} = g_0^{xv/t_{\rho_1(j)}}$, $K_0 = g_0^{y+xv}$, $K_1 = g_0^{xv}g_1^z$, $K_2 = g_0^z$, $K_3 = g_0^{y-a}$, where $a = \sum_{j=1}^{|S|} a_j$, $|S|$ denotes the number of attributes in S, the function $\rho_1(\cdot)$ maps the subscript of attribute in S to that of attribute in $\mathcal{U}$, namely $att_j^* = att_{\rho_1(j)}$. Finally, the KGS returns the secret keys $SK_{FN} = (K_1, K_2, K_3, r, \{K_{j,2}, K_{j,3}\})$, $SK_{EU} = (K_0, u, \{K_{j,1}\})$ to FN and EU, respectively.

Enc$(PK, \mathcal{W}, \mathcal{F}, \mathcal{T}, \Gamma)$: Given files $\mathcal{F} = \{F_1, \cdots, F_m\}$ and associated keyword set $\mathcal{W} = \{W\}$, DO first encrypts each file $F_\tau \in \mathcal{F}(\tau \in [1, m])$ as c_τ with the symmetric key k_τ, namely $c_\tau = E_{k_\tau}(F_\tau)$.

To encrypt each key k_τ, the DO assigns an access policy Γ described by an access tree $\mathcal{T}$ and then sends Γ to a certain FN to output file symmetric key encryption CT_τ by cooperating with the chosen FN.

- After gaining Γ, the chosen FN first chooses a polynomial q_w with degree d_w for each node w in access tree $\mathcal{T}$. Starting from the root node ν, the polynomials are selected in a top-down manner. For each node w, its threshold value is defined as $k_w = d_w + 1$. The FN picks an element $\theta \in_R \mathcal{Z}_p^*$ and then sets $q_\nu(0) = \theta$, then FN can choose other d_ν points to define d_ν completely. For any other non-root node w, the FN first sets $q_w(0) = q_{parent(w)}(index(w))$ and then defines q_w completely by utilizing other random q_w points. Let $\mathcal{L} = \{l\}$ be the leaf nodes in $\mathcal{T}$, and each leaf node l is associated with an attribute. Then the FN computes $C_1 = g_0^\theta$, $C_2 = g_1^\theta$, $C_l = g_0^{t_l q_l(0)}$.

Finally, the FN returns the intermediate ciphertext CT_τ in Eq. 3 to the DO.

$$CT_\tau = (C_1, C_2, \{C_l\}_{l\in\mathcal{L}}). \tag{3}$$

- After receiving the key ciphertext set $\{CT_\tau\}$, the DO chooses an element $h \in_R \mathcal{Z}_p^*$, and then computes $CC_\tau = k_\tau \cdot e(g_0, g_0)^{yh}$, $C' = g_0^h$, $C_1' = C_1 g_0^h$, $C_2' = C_2 g_1^h$. Finally, the DO returns the file ciphertexts $\{c_\tau\}$, file key ciphertexts $\{CT_\tau^*\}$ to CSP via the chosen FN, where CT_τ^* is defined by Eq. 4.

$$CT_\tau^* = (CC_\tau, C', C_1', C_2', \{C_l\}_{l\in\mathcal{L}}). \tag{4}$$

To generate the index for each file F_τ, the DO first selects an element $d_l \in_R \mathcal{Z}_p^*$ for each attribute in Γ, and then it computes $I_0 = Y^s$, $I_1 = g_0^s$, $I_{l,1} = A_l^{d_l H_1(W)}$, $I_{l,2} = (s - d_l)H_1(W)$, where the keyword W is included in F_τ. Finally, the DO sends the final index set $I = \{I_\tau\}$ as well as the tuple $(\{c_\tau\}, \{CT_\tau^*\})$ to CSP via the above FN, where the symbol I_τ is defined by Eq. 5.

$$I_\tau = (I_0, I_1, \{I_{l,1}, I_{l,2}\}_{l\in\mathcal{L}}). \tag{5}$$

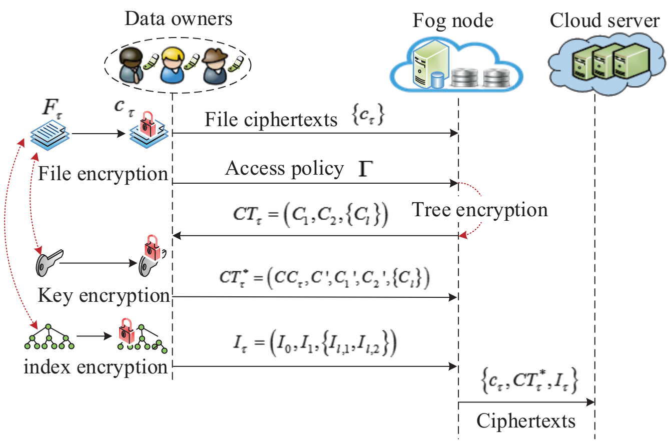
Data owners
Fog node
Cloud server
F_τ
c_τ
File ciphertexts $\{c_\tau\}$
File encryption
Access policy Γ
$CT_\tau = (C_1, C_2, \{C_l\})$
Tree encryption
Key encryption
$CT_\tau^* = (CC_\tau, C', C_1', C_2', \{C_l\})$
index encryption
$I_\tau = (I_0, I_1, \{I_{l,1}, I_{l,2}\})$
$\{c_\tau, CT_\tau^*, I_\tau\}$
Ciphertexts

Fig. 5. Interacting process in **Enc** algorithm.

Trap$(W', S, SK_{FN}, SK_{EU}, PK_{FN}^*, PK_{EU}^*)$: When an authorized EU with an attribute set S wants to issue a search query according to the keyword W' of interest, he can generate a search token (or trapdoor) with the help of a certain FN, note that the chosen FN cannot output the valid trapdoor by himself. The specific trapdoor generation process is shown as follows:

- Once gaining the requirement for trapdoor generation, the chosen FN will execute the following process if and only if the EU is in the authorized UL. The FN first picks an element $\eta \in_R \mathcal{Z}_p^*$, and then it computes $T_1 = K_3^\eta$ and $T_{j,1} = K_{j,2}^\eta$ for each attribute $att_j^* \in S$. Finally, the chosen FN returns the symbol $T_{W',1} = (T_1, \{T_{j,1}\})$ to the EU.
- After gaining $T_{W',1}$, the EU first selects $\lambda \in_R \mathcal{Z}_p^*$ and computes $T_0 = u + \lambda$, $T_1' = T_1^\lambda$, and then it outputs $T_{j,1}' = T_{j,1}^{\lambda/H_1(W')}$, $T_{j,2} = K_{j,1}^{\lambda/H_1(W')}$ for each attribute $att_j^* \in S$. Finally, the EU returns the symbol $T_{W',2} = (T_0, T_1', \{T_{j,1}', T_{j,2}\})$ to the chosen FN.
- When receiving $T_{W',2}$, the FN computes $T_0' = T_0\eta + r$ and $T_{j,2}' = T_{j,2}^\eta$ for each attribute in S. Then, the FN computes $T_{EU}^* = (PK_{EU}^*)^\eta$. Finally, the FN sends the final trapdoor $T_{W'} = (T_0', T_1', T_{EU}^*, \{T_{j,1}', T_{j,2}'\})$ and the EU's attributes S to CSP.

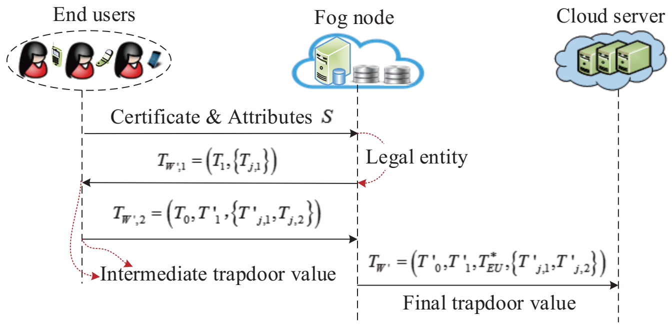
End users
Fog node
Cloud server
Certificate & Attributes S
Legal entity
$T_{W',1} = (T_1, \{T_{j,1}\})$
$T_{W',2} = (T_0, T'_1, \{T'_{j,1}, T_{j,2}\})$
Intermediate trapdoor value
$T_{W'} = (T'_0, T'_1, T_{EU}^*, \{T'_{j,1}, T'_{j,2}\})$
Final trapdoor value

Fig. 6. Interacting process in **Trap** algorithm.

Search$(\{c_\tau\}, \{CT_\tau^*\}, I, S, T_{W'}, \Gamma)$: After gaining the trapdoor, the CSP first checks whether the EU's attribute set S matches the access structure Γ. If it is true, the CSP conducts the following steps; otherwise, the CSP aborts this process.

- First, the CSP computes $I_l^* = I_{l,1} \cdot A_l^{I_{l,2}} = A_l^{s/H_1(W)}$ and $e(I_l^*, T_{j,1}')$, $e(I_l^*, T_{j,2}')$ for each attribute $att_j^* \in S$ with Eq. 6. If the equations $t_l = t_{\rho_1(j)}$, $H_1(W') = H_1(W)$ hold, then we can further gain $e(I_l^*, T_{j,1}') = e(g_0, g_0)^{s\eta\lambda(a_j - b_j)}$, $e(I_l^*, T_{j,2}') = e(g_0, g_0)^{b_j s\eta\lambda}$

$$\begin{aligned} e(I_l^*, T_{j,1}') &= e(g_0, g_0)^{\frac{t_l H_1(W) s\eta\lambda(a_j - b_j)}{H_1(W') t_{\rho_1(j)}}}; \\ e(I_l^*, T_{j,2}') &= e(g_0, g_0)^{\frac{b_j s t_l H_1(W)\eta\lambda}{H_1(W') t_{\rho_1(j)}}}. \end{aligned} \tag{6}$$

- Then, the CSP checks whether the submitted trapdoor $T_{W'}$ satisfies the indexes I with Eq. 7.

$$\begin{aligned} e(I_1, T_1') \prod_{j\in[1,|S|]} e(I_{\rho_1(j)}^*, T_{j,1}' T_{j,2}') &= \\ I_0^{T_0'} \cdot T_{EU}^* \cdot PK_{FN}^*. \end{aligned} \tag{7}$$

- Finally, if Eq. 7 holds, it indicates that $W' = W$ and then the CSP returns the corresponding search results which include file ciphertexts $\{c_\pi'\}$ and file key ciphertexts $\{CT_\pi'\}$ to the FN. Besides, we define an function $\rho_2(\cdot)$ which maps the subscript of ciphertexts c_π' (or CT_π') to that of $c_{\rho_2(\pi)}$ (or $CT_{\rho_2(\pi)}^*$), namely $c_\pi' = c_{\rho_2(\pi)}$, $CT_\pi' = CT_{\rho_2(\pi)}^*$.

Dec$(\{c_\pi'\}, \{CT_\pi'\}, SK_{FN}, SK_{EU}, PK, \mathcal{T}, \Gamma)$: To decrypt $\{c_\pi'\}$, the algorithm needs to gain the file keys $\{k_\pi'\}$ embedded in CT_π' by leveraging the following recursive algorithm. The specific decryption process involving FN and EU is shown as follows:

- First, the FN calls the recursive algorithm by taking the two cases into consideration. (1) If the node w in $\mathcal{T}$ is a leaf node and it satisfies $att(w) \in S$, then the FN computes $\varphi_w = e(K_{att(w),3}, C_w) =$

$e(g_0, g_0)^{xvq_w(0)}$; (2) If the node w is a non-leaf node, the FN first computes $\varphi_{w'}$ for each children w' of node w. Let S_w be an arbitrary ϕ_w-size child node set $\{w'\}$, if there does not exist such set, then $\varphi_{w'} = \perp$; otherwise, the FN computes $\varphi_{w'}$ with Eq. 8, where $i = index(w')$, $S'_w = \{index(w') : w' \in S_w\}$.

$$\begin{aligned}\varphi_w &= \prod_{w' \in \phi_w} \varphi_{w'}^{\Delta_{i,S'_w}(0)} \\ &= (e(g_0, g_0)^{xvq_{parent(w')}(index(w'))})^{\Delta_{i,S'_w}(0)} \\ &= e(g_0, g_0)^{xvq_w(0)}.\end{aligned} \tag{8}$$

- Then, the FN obtains $\varphi_\nu = e(g_0, g_0)^{xv\theta}$ if S matches the access structure Γ. After that, the FN computes M with Eq. 9 and $M^* = M/\varphi_\nu = e(g_0, g_0)^{xvh}$ before returning the tuple $(\{c'_\pi\}, \{CC'_\pi\}, C', M^*)$ to EU.

$$M = \frac{e(K_1, C'_1)}{e(K_2, C'_2)} = e(g_0, g_0)^{xv(\theta+h)}. \tag{9}$$

- Finally, the EU computes the each file key k'_π with Eq. 10 before decrypting each file ciphertext $c'_\pi = E_{k'_\pi}(F'_\pi)$, where $CC'_\pi = CC_{\rho_2(\pi)}$.

$$k'_\pi = \frac{CC'_\pi \cdot M^*}{e(K_0, C')} = \frac{k'_\pi \cdot e(g_0, g_0)^{yh} e(g_0, g_0)^{xvh}}{e(g_0^{y+xv}, g_0^h)}. \tag{10}$$

Remarks. To reduce the computational burden of resource-constrained cloud clients including DO and EUs, the lightweight LFGS system shifts the partial computation of cloud clients to FNs in fog computing environment, which can be applied in a broad range of applications. Although the basic LFGS system can provide fine-grained access control and keyword search simultaneously, it cannot support attribute update and conjunctive keyword. For example, if the attributes associated with roles of a certain EU have been update, the malicious EU can still access the original ciphertexts by utilizing his early secret key related to his old role. To prevent this kind of illegal accesses, the prior schemes need to update all ciphertexts, which bring high computational and storage burden. Besides, the single keyword search will return all file ciphertexts containing the queried keyword, while some of these results may not be what the EUs need, which greatly wastes the bandwidth and computation resources. Along these directions, we extend the basic LFGS system to achieve these goals in the following subsections.

5.1 Attribute Update in Extended LFGS System

Taking the dynamic scenarios into consideration, a certain attribute may change and the malicious EUs can access the unauthorized information with the outdated secret keys. Aiming to gain this feature, the extended LFGS scheme needs to support attribute update. For simplicity, we assume there exists an attribute ($att_i \rightarrow att_{\sigma \notin [1,n]}$) needed to be updated in extended LFGS system. Next, we mainly present the modified content which are different from those of basic LFGS system.

- *Key update*. The KGC first chooses an element $t_\sigma \in_R \mathcal{Z}_p^*$ and then computes $t_{i\rightarrow\sigma} = t_i/t_\sigma$ used to update the secret key of updated FN and partial ciphertexts in CSP. Then, the KGC picks another element $t_i^* \in_R \mathcal{Z}_p^*$ for attribute att_i and computes $t_{i,1} = t_i/t_i^*$, $t_{i,2} = t_i^*/t_i$, where $t_{i,1}$ is used to update the secret key of non-updated FNs, $t_{i,2}$ is utilized to update ciphertexts in CSP. Finally, the KGC updates the public key of updated attribute as $A_i^* = A_i^{t_{i,2}} = g_0^{t_i^*}$. Besides, the partial secret key of updated FN is set as $K_{i,3} = g_0^{(xv/t_i)\cdot t_{i\rightarrow\sigma}} = g_0^{xv/t_\sigma}$, and that of non-updated FN is defined as $K_{i,3} = g_0^{(xv/t_i)\cdot t_{i,1}} = g_0^{xv/t_i^*}$
- *Ciphertexts update*. To enable the FNs to execute partial decryption operation, it is required that the CSP should update the ciphertext associated with the updated attribute with two cases: (1) for updated FN, the stored ciphertext C_i is set as $C_i = g_0^{t_i q_i(0)/t_{i\rightarrow\sigma}} = g_0^{t_\sigma q_\sigma(0)}$; for non-updated FN, C_i is defined as $C_i = g_0^{t_i q_i(0)\cdot t_{i,2}} = g_0^{t_i^* q_i(0)}$

It is worth noticing that the attribute does not exert computational burden on resource-limited EUs. Furthermore, the FNs and CSP just need to update partial secret keys or ciphertexts, respectively. Last but not least, the extended LFGS system also does not need to update indexes and trapdoors which are not involved with the secret key $K_{j,3}$. Hence, our extended LFGS system is still lightweight.

5.2 Conjunctive Keyword Search in Extended LFGS System

As the single keyword search in the basic LFGS system will return many irrelevant search results, which wastes the bandwidth and computation resources. To further narrow down the search scope and quickly locate the search results, the extended LFGS system should support conjunctive keyword search. Next, we also demonstrate the modified content in the extended LFGS system.

- *Index generation*. For each file F_τ including multiple keywords, if F_τ contains the keyword $W \in \mathcal{W}$, the **Enc** algorithm first computes $I_{l,1,W} = A_l^{d_l H_1(W)}$, $I_{l,2,W} = (s - d_l)H_1(W)$; otherwise, it sets $I_{l,1,W} = I_{l,2,W} = 1$. Then, the $I_{l,1} = \{I_{l,1,W}\}$, $I_{l,2} = \{I_{l,2,W}\}$.
- *Trapdoor generation*. Given the search query involving with multiple keywords $W^* = \{W'_1, \cdots, W'_\hbar\}$, the EU first computes $T_{j,1}^* = T_{j,1}^{\lambda/\sum_{i\in[1,\hbar]} H_1(W'_i)}$, $T_{j,2} = K_{j,1}^{\lambda/\sum_{i\in[1,\hbar]} H_1(W'_i)}$, where $H_1(W'_i) \in W^* \subset \mathcal{W}$. Then, the EU returns the tuple $\{T_{j,1}^*, T_{j,2}\}$ and the queried keyword location set in $\mathcal{W}$ to the chosen FN to output the final trapdoor.

Although the conjunctive keyword search will increase the index size, the **Enc** algorithm is just one-time operation and the high storage burden is outsourced to CSP, which does not bring storage burden on DO. Besides, the hash operation $\mathcal{O}_{H_1}$ is much efficient than other operations (i.e., pairing operation, exponential operation, etc.). Hence, the extended LFGS system also incurs less computational burden on EUs.

6 Security and Performance Analysis

In this section, we first give the formal security analysis of LFGS system with following theorems, and then demonstrate its performance in terms of theoretical and practical costs.

6.1 Security Analysis

In our LFGS system, the file encryption keys and indexes are encrypted by the fully-trusted DO, and the submitted trapdoors (or search tokens) are generated by authorized EUs, note that the malicious EUs cannot collude with the CSP and FNs. Besides, the collusion attacks between malicious EUs can be avoided by the CP-ABE mechanism used in our LFGS system. Thus, the file and keyword privacy can be well protected in the *known ciphertext model*, but the access pattern (i.e., the list of returned files, etc.) and search pattern (i.e., whether the encrypted files are returned by the same keyword, etc.) privacy cannot be guaranteed in the *known background model* as it is extremely expensive since the algorithm has to "touch" the whole file set. Thus, the LFGS system does not aim to protect this kind of privacy for efficiency concerns.

Although the FNs in LFGS system are considered to honest-but-curious, these entities cannot issue replay attack as the file encryption key ciphertext $(CC_\tau, C', C_1', C_2')$, indexes $\{I_{l,1}, I_{l,2}\}$ (or trapdoor $(T_0, T_1', \{T_{j,1}', T_{j,2}\})$) are confused by DOs (or EUs) random elements $h \in_R \mathcal{Z}_p^*$, $d_l \in_R \mathcal{Z}_p^*$ (or $\lambda \in_R \mathcal{Z}_p^*$), respectively. In the Enc algorithm, the chosen FN is just in charge of encrypting access policy and cannot generate the final ciphertxts (CT_τ^*) without DOs random elements; the selected FN still cannot output the valid trapdoor $T_{W'}$ without EUs secret keys in Trap algorithm. Although the honest-but-curious FN can deduce sensitive information when treated as an intermediate communication party, it still honestly executes the predefined protocol. Thus, the LFGS system also can resist the man-in-the-middle attack.

As for outsourced files, the LFGS system requires that the malicious attackers cannot adaptively ask for the ciphertexts of arbitrary plaintext messages. That is to say, the LFGS system should achieve the CPA security which can be reduced to the DBDH assumption with the following theorem.

Theorem 1. Assume that a PPT adversary $\mathcal{A}$ can break the CPA security of LFGS system with a non-negligible advantage ϵ, then there must exist a PPT simulator $\mathcal{B}$ which can break the DBDH problem with an advantage $\epsilon/2$.

Proof: Given the public bilinear parameters (G, G_T, e, g_0), the challenger $\mathcal{C}$ chooses several elements $a', b', c' \in_R \mathcal{Z}_p^*$, $Z \in G_T$ and a random bit $\kappa \in \{0,1\}$. If $\kappa^* = 0$, the equation $Z = e(g_0, g_0)^{a'b'c'}$ holds; otherwise, Z is a random element in group G_T. Assume that $\mathcal{C}$ gives the tuple $(g_0, g_0^{a'}, g_0^{b'}, g_0^{c'}, Z)$ to $\mathcal{B}$, then $\mathcal{B}$ plays the role of $\mathcal{C}$ in the following security game:

- *Initialization.* $\mathcal{A}$ first selects a challenging access structure Γ^*, then $\mathcal{A}$ sends Γ^* to $\mathcal{B}$.
- *Setup.* $\mathcal{B}$ first picks an element $y' \in_R \mathcal{Z}_p^*$ and computes $y = y' + a'b'$, then $\mathcal{B}$ sets $Y = e(g_0, g_0)^y = e(g_0, g_0)^{y'} e(g_0, g_0)^{a'b'}$, $B = g_0^x = g_0^{a'}$, $g_1 = g_0^{\varpi}$. For each attribute $att_i \in \mathcal{U}$, $\mathcal{B}$ chooses an element $\chi_i \in_R \mathcal{Z}_p^*$ and outputs $A_i = g_0^{a'\chi_i^{-1}} = g_0^{t_i}$. If $att_i \in \Gamma^*$, the equation $t_i = a'\chi_i^{-1}$ holds; otherwise, $A_i = g_0^{\chi_i} = g_0^{t_i}$, namely $\chi_i = t_i$. Finally, $\mathcal{B}$ returns the partial public keys $(Y, B, g_1, \{A_i\}_{i\in[1,n]})$ to $\mathcal{A}$.
- *Phase 1.* $\mathcal{A}$ adaptively submits an attribute set $S \subseteq \mathcal{U}$ to $\mathcal{B}$ to issue secret key query. $\mathcal{B}$ first chooses an element $v \in_R \mathcal{Z}_p^*$ and outputs $v = v' - b'$, then $\mathcal{B}$ defines $K_0 = g_0^{y+xv} = g_0^{y'+a'v'} = g_0^{y'} B^{v'}$, $K_1 = g_0^{a'v'} g_1^z = B^{v'} g_1^z$, $K_2 = g_0^{a'z} = B^z$. For each attribute $att_j^* \in S$, if $att_j^* \in \Gamma^*$, $\mathcal{B}$ returns $K_{j,3} = g_0^{a'^{-1}\chi_{\rho_1(j)}a'v} = g_0^{a'v't^{-1}_{\rho_1(j)}} = B^{v't^{-1}_{\rho_1(j)}}$; otherwise, $\mathcal{B}$ outputs $K_{j,3} = g_0^{\chi^{-1}_{\rho_1(j)}v'a'} = B^{v't^{-1}_{\rho_1(j)}}$. Finally, $\mathcal{B}$ returns the secret key to $\mathcal{A}$.
- *Challenge.* $\mathcal{A}$ randomly submits two messages m_0, m_1 on which to be challenged to $\mathcal{B}$, $\mathcal{B}$ first sends Γ^* to the chosen FN which selects an element $\theta \in_R \mathcal{Z}_p^*$ to construct shares of θ for all corresponding attributes and returns the challenging ciphertexts $(g_0^\theta, g_1^\theta, \{C_i\})$ for all attributes $att_i \in \Gamma^*$. Then, $\mathcal{B}$ picks an element $h \in_R \mathcal{Z}_p^*$, a random bit $\kappa \in \{0,1\}$ and computes $C' = g_0^h = g_0^{c'}$, $C_1' = g_0^{c'} g_0^\theta$, $C_2' = g_1^\theta g_1^{c'} = g_0^{c'} g_0^{\theta\varpi}$, $CC_\kappa = m_\kappa e(g_0, g_0)^{yh} = m_\kappa e(g_0, g_0)^{a'b'c'+y'c'} = m_\kappa Z e(g_0, g_0)^{y'c'}$. Finally, $\mathcal{B}$ sends the tuple $(\Gamma^*, CC_\kappa, C', C_1', C_2', \{C_i\})$ to $\mathcal{A}$.
- *Phase 2.* $\mathcal{A}$ repeatedly the process in *Phase 1*, but the restriction is that the two messages m_0, m_1 cannot be challenged.
- *Guess.* $\mathcal{A}$ returns a guess bit $\kappa' \in \{0,1\}$. $\mathcal{B}$ returns $\kappa^* = 0$ indicating that $Z = e(g_0, g_0)^{a'b'c'}$ on the condition that $\kappa' = \kappa$; otherwise, $\mathcal{B}$ returns $\kappa^* = 1$ indicating that Z is a random element in group G_T.

If $Z = e(g_0, g_0)^{a'b'c'}$, then $\mathcal{A}$ can break the security game with an advantage ϵ, and the $\mathcal{B}$'s advantage $Adv^{\mathcal{B}}_{DBDH}(1^k)$ in solving the DBDH problem is $\frac{1}{2} + \epsilon$; otherwise, $Adv^{\mathcal{B}}_{DBDH}(1^k) = \frac{1}{2}$. Hence, the advantage of $\mathcal{B}$ in breaking the security game outlined above is defined as $Adv^{\mathcal{B}}_{DBDH}(1^k) = \frac{1}{2}(\frac{1}{2} + \epsilon + \frac{1}{2}) - \frac{1}{2} = \frac{\epsilon}{2}$. This completes the proof of **Theorem** 1. □

Besides, to guarantee the security of LFGS system, the malicious attackers cannot distinguish an encryption of a keyword W_0 from an encryption of a keyword W_1 for which he does not gain the corresponding trapdoor. Thus, the LFGS system should resist the chosen-keyword attack with the following theorem.

Theorem 2. If there exists a PPT adversary $\mathcal{A}$ can break the CKA security of LFGS system with a non-negligible advantage ϵ, then we can construct a simulator $\mathcal{B}$ which can solve the DBDH problem with an advantage $\epsilon/2$.

Proof: Given the security parameter k and the whole attribute set $\mathcal{U}$, the challenger $\mathcal{C}$ chooses several elements $a', b', c', \xi \in_R \mathcal{Z}_p^*$ and random bit $\kappa^* \in \{0,1\}$. If $\kappa^* = 0$, $\mathcal{C}$ sets $Z = e(g_0, g_0)^{a'b'c'}$; otherwise, $\mathcal{C}$ sets $Z = e(g_0, g_0)^{\xi}$. Then, $\mathcal{C}$ sends the tuple $\{g_0^{a'}, g_0^{b'}, g_0^{c'}, Z\}$ to $\mathcal{B}$ that will plays a challenger in the following security game.

- *Initialization.* $\mathcal{A}$ first selects a challenging access structure Γ^*, then $\mathcal{A}$ sends Γ^* to $\mathcal{B}$.

- *Setup*. $\mathcal{B}$ first sets $Y = e(g_0, g_0)^y = e(g_0, g_0)^{a'b'}$ with the equation $y = a'b'$ holds, then $\mathcal{B}$ selects $u, r \in_R \mathcal{Z}_p^*$ and sets $PK_{EU} = e(g_0, g_0)^{a'b'u}$, $PK_{FN} = e(g_0, g_0)^{a'b'r}$. Besides, for each attribute $att_i \in \mathcal{U}$, $\mathcal{B}$ picks an element $t_i \in_R \mathcal{Z}_p^*$ and computes $A_i = g_0^{t_i}$. Finally, $\mathcal{B}$ outputs the above public parameters.
- *Phase 1*. $\mathcal{A}$ can ask for different trapdoor queries by submitting any keyword W and the corresponding attribute set S which does not satisfy the challenging access structure Γ^*, and $\mathcal{B}$ responses these queries by conducting the following three operations:

  - **Trap-I**(W, S): For each attribute $att_j^* \in S$, $\mathcal{B}$ first selects $a'_j, b'_j \in_R \mathcal{Z}_p^*$ and defines $b_j = b'b'_j$, $a_j = b'a'_j$. Then, $\mathcal{B}$ sets $a = \sum a_j = \sum b'a'_j$. Besides, $\mathcal{B}$ picks an element $\eta \in_R \mathcal{Z}_p^*$ and computes $T_1 = g_0^{-\eta \sum b'a'_j}$, $T_{j,1} = g_0^{\eta(a'_j - b'_j)/t_{\rho_1(j)}} = g_0^{\eta(a_j - b_j)/t_{\rho_1(j)}}$. Finally, $\mathcal{B}$ returns the symbol $T_{W,1} = (T_1, \{T_{j,1}\})$ to $\mathcal{A}$.
  - **Trap-II**(W, S): $\mathcal{B}$ first selects an element $\lambda \in_R \mathcal{Z}_p^*$, then $\mathcal{B}$ computes $T_0 = u + \lambda$, $T'_1 = T_1^\lambda$, $T'_{j,1} = T_{j,1}^{\lambda/H_1(W)}$, $T_{j,2} = g_0^{b'_j \eta / H_1(W) t_{\rho_1(j)}}$. Finally, $\mathcal{B}$ sends the tuple $(T_0, T'_1, \{T'_{j,1}, T_{j,2}\})$ to $\mathcal{A}$.
  - **Trap-III**(W, S): $\mathcal{B}$ first computes $T'_0 = \eta T_0 + r$, $PK^*_{FN} = Z^{-r}$, $PK^*_{EU} = Z^{-u\eta}$. Then, for each attribute $att_j^* \in S$, $\mathcal{B}$ computes $T'_{j,2} = T_{j,2}^\eta$. Finally, $\mathcal{B}$ sends the tuple $(PK^*_{FN}, PK^*_{EU}, T'_0, T'_1, \{T'_{j,1}, T'_{j,2}\})$ to $\mathcal{A}$.

- *Challenge*. $\mathcal{A}$ randomly submits two keyword W_0, W_1 on which to be challenged to $\mathcal{B}$, $\mathcal{B}$ chooses a random bit $\kappa \in \{0, 1\}$ and encrypts W_κ with the challenging access structure Γ^*. $\mathcal{B}$ first sets $I_0 = Z$, $I_1 = g_0^{c'}$, then computes $I^*_{\rho_1(j)} = g_0^{c'\rho_1(j)H_1(W_\kappa)}$. Finally, $\mathcal{B}$ returns the tuple $(I_0, I_1, \{I^*_{\rho_1(j)}\})$ to $\mathcal{A}$.
- *Phase 2*. The process is as same as that in *Phase 1*, but the restriction is that queried keywords in *Phase 1* cannot be issued again in this phase.
- *Guess*. $\mathcal{A}$ outputs a guess bit $\kappa' \in \{0, 1\}$. If $\kappa' = \kappa$, $\mathcal{B}$ outputs $\overline{\kappa}^* = 0$ indicating that the equation $Z = e(g_0, g_0)^{a'b'c'}$ holds; otherwise, $\mathcal{B}$ outputs $\overline{\kappa}^* = 1$ indicating that Z is a random element in G_T.

When $\kappa^* = 1$, it indicates that $\mathcal{A}$ does not obtain any information about κ. Thus, we can obtain $Pr[\Gamma \neq \Gamma' | \kappa^* = 1] = \frac{1}{2}$. If $\mathcal{B}$ outputs $\overline{\kappa}^* = 1$ on condition that $\Gamma \neq \Gamma'$, we also gain $Pr[\kappa^* = \overline{\kappa}^* | \kappa^*] = Pr[\overline{\kappa}^* = 1 | \kappa^* = 1] = \frac{1}{2}$. When $\overline{\kappa}^* = 0$, $\mathcal{A}$ can obtain a valid encryption for keyword W_κ and can break the above security game with a non-negligible advantage ϵ. In this case, we can further have $Pr[\kappa = \kappa' | \kappa^* = 0] = \frac{1}{2} + \epsilon$. If $\mathcal{B}$ outputs $\kappa^* = \overline{\kappa}^*$ on condition that $\Gamma = \Gamma'$, we can have $Pr[\kappa^* = \overline{\kappa}^* | \kappa^* = 0] = Pr[\overline{\kappa}^* = 0 | \kappa^* = 0] = \frac{1}{2} + \epsilon$. Therefore, the $\mathcal{B}$'s advantage in breaking the DBDH problem is $Pr[\overline{\kappa}^* = \kappa^*] - \frac{1}{2} = Pr[\overline{\kappa}^* = \kappa^* | \kappa^* = 1] Pr[\kappa^* = 1] + Pr[\overline{\kappa}^* = \kappa^* | \kappa^* = 0] Pr[\kappa^* = 0] - \frac{1}{2} = \frac{\epsilon}{2}$. This completes the proof of Theorem 2. □

TABLE 2
Functional comparison in various schemes

Schemes	Function 1	Function 2	Function 3	Function 4
VABKS [15]	✓			
ABKS-UR [16]	✓	✓		
SCP-ABKS [17]	✓		✓	
m^2-ABKS [19]	✓	✓		
KSF-OABE [35]	✓			
MO-ABKS [36]	✓	✓		
HFGA [37]	✓			✓
Basic LFGS	✓			✓
Extended LFGS	✓	✓	✓	✓

"Function 1": Fine-grained keyword search;
"Function 2": Conjunctive keyword search;
"Function 3": Attribute update;
"Function 4": Fog computing.

6.2 Performance Analysis

Before analyzing the performance of LFGS system, we first compared our LFGS system with several state-of-the-art CP-ABKS schemes [15], [16], [17], [19], [35], [36], [37] in TABLE 2, and we notice that the extended LFGS system supports fine-grained keyword search[3], conjunctive keyword search, attribute update and fog computing simultaneously. As for the performance analysis of LFGS system, we mainly present its theoretical and actual performance by comparing with the state-of-the-art schemes, i.e., HFGA scheme [37], ABKS-UR scheme [16].. Note that we omit the performance analysis of extend LFGS system as the conjunctive keyword search and attribute update do not bring high computational and storage burden.

With regard to theoretical analysis, we first present the computation in TABLE 3. We mainly consider several time-consuming operations like bilinear pairing operation P, exponentiation operation E (resp., E_T) in group G (resp., G_T). Note that the computational costs of LFGS scheme are affected by the number of submitted attributes, while those of other two schemes are influenced by the number of system attributes. Because of $S \subseteq \mathcal{U}$, we can assume that $|S| < n$ in practical applications. Hence, compared with the efficient ABKS-UR scheme, the LFGS system can further reduce the EUs' computational burden. Furthermore, the LFGS system has fewer computational costs on the chosen FN than the HFGA system in fog computing environment.

Besides, we present the storage costs of aforementioned three schemes in TABLE 4, where the element lengths in $G, G_T, \mathcal{Z}_p$ are defined as $|G|$, $|G_T|$ and $|\mathcal{Z}_p|$, respectively. With the same reason shown in TABLE 3, the LFGS system still outperforms other two schemes in terms of storage costs of different algorithms.

As for the actual performance analysis, we conduct experimental simulations using real-world Enron Email Dataset[4] which includes half a million records from 150 users to evaluate the actual performance of aforementioned schemes. This public email dataset used in many SE schemes contains half a million records from about 150 users, mostly senior management of Enron, and the Enron corpus contains

3. It means that it just supports single keyword search.
4. http://www.cs.cmu.edu/~enron/

TABLE 3
Computation costs in various schemes

Algorithms	LFGS		HFGA [37]		ABKS-UR [16]	
	FN	EU	FN	EU	FN	EU
KeyGen	$(2\|S\|+3)E+2E_T$	$(\|S\|+1)E+2E_T$	$2E_T$	$(4n+1)E+E_T$	—	$(2n+1)E+E_T$
Enc	$(n+2)E$	$(n+3)E+2E_T$	—	$(2n+4)E+E_T$	—	$(n+1)E+E_T$
Trap	$(2\|S\|+1)E+E_T$	$(2\|S\|+1)E$	$(4n+1)E+E_T$	$(4n+1)E$	—	$(2n+1)E$
Search	$(\|S\|+1)P+E+E_T$		$(2n+1)P+2nE+E_T$		$(n+1)P+E_T$	
Dec	$(n+2)P$	P	$(2n+1)P+(2n+1)E$	E_T	—	—

Notes. "$|S|$": The number of submitted attributes; "n": The number of system attributes.

TABLE 4
Storage costs in various schemes

Algorithms	LFGS		HFGA [37]	
	FN	EU	FN	EU
KeyGen	$(2\|S\|+4)\|G\|+2\|G_T\|+\|\mathcal{Z}_p\|$	$(\|S\|+1)\|G\|+2\|G_T\|+\|\mathcal{Z}_p\|$	$2\|G_T\|+2\|\mathcal{Z}_p\|$	$(4n+1)\|G\|+2\|G_T\|+(2n+1)\|\mathcal{Z}_p\|$
Enc	$n\|\mathcal{Z}_p\|+(n+2)\|G\|$	$(n+4)\|G\|+(n+1)\|\mathcal{Z}_p\|+2\|G_T\|$	—	$(2n+4)\|G\|+n\|\mathcal{Z}_p\|+2\|G_T\|$
Trap	$(2\|S\|+1)\|G\|+\|G_T\|+2\|\mathcal{Z}_p\|$	$(2\|S\|+1)\|G\|+2\|\mathcal{Z}_p\|$	$(4n+1)\|G\|+\|G_T\|+\|\mathcal{Z}_p\|$	$(4n+1)\|G\|+\|\mathcal{Z}_p\|$
Search	$(\|S\|+1)\|G_T\|$		$(2n+2)\|G_T\|+2n\|G\|$	
Dec	$(n+1)\|G_T\|$	$\|G_T\|+\|\mathcal{Z}_p\|$	$(2n+1)\|G\|+(2n+1)G_T$	$\|G_T\|$

Scheme	**KeyGen**	**Enc**	**Trap**	**Search**
ABKS-UR	$\|\mathcal{Z}_p\|+(2n+1)\|G\|+\|G_T\|$	$(2n+1)\|G\|+\|G_T\|+2\|G_T\|$	$\|\mathcal{Z}_p\|+(2n+1)\|G\|$	$(n+3)\|G_T\|$

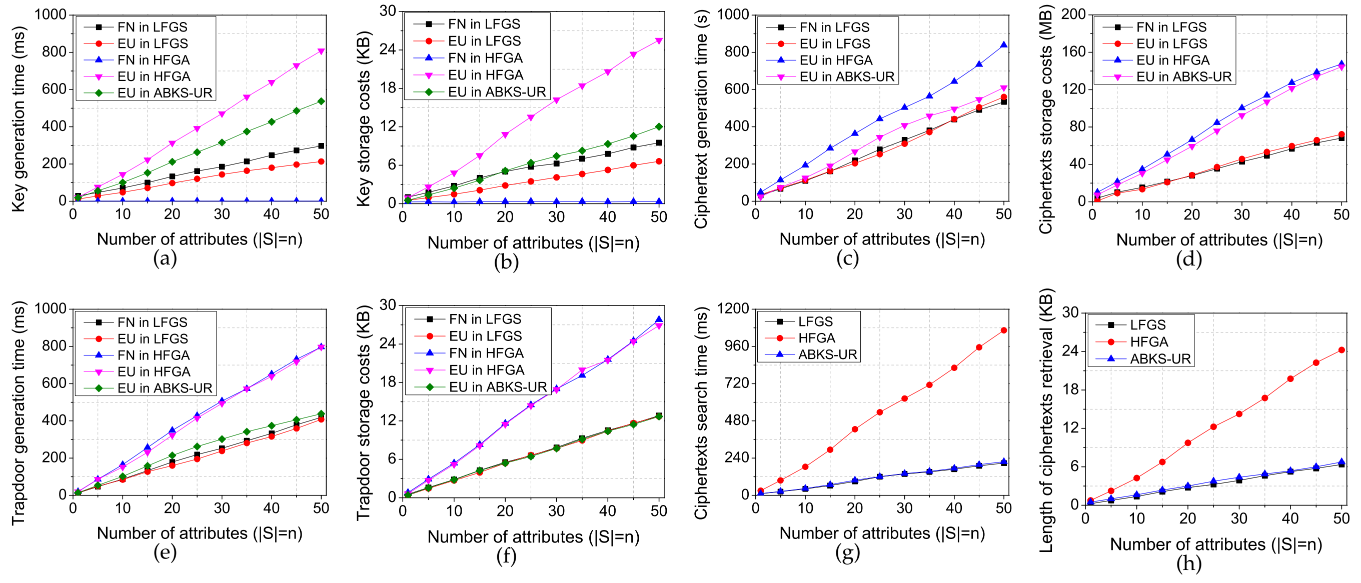

Fig. 7. Performance analysis in various schemes: (a) Computational costs in **KeyGen** algorithm; (b) Storage costs in **KeyGen** algorithm; (c) Computational costs in **Enc** algorithm; (d) Storage costs in **Enc** algorithm; (e) Computational costs in **Trap** algorithm; (f) Storage costs in **Trap** algorithm; (g) Computational costs in **Search** algorithm; (h) Storage costs in **Search** algorithm.

a total of about 0.5 MB (Megabyte) message. Experiments are implemented on an Ubuntu Server 15.04 with Intel Core i5 Processor 2.3 GHz by using C and Paring Based Cryptography (PBC) Library. In PBC Library, the Type A is denoted as $E(F_q) : y^2 = x^3 + x$, the group G and group G_T of order p are subgroups of $E(F_q)$, where the parameters p and q are equivalent to 160 bits and 512 bits, respectively. In line with prior schemes, we choose 10000 files from the public dataset and conduct the experimental tests for 100 times. For comparison, we set $|S| = n \in [1, 50]$ in the whole paper.

In Fig. 7 (a), (b), we show the actual performance of **KeyGen** algorithm in different schemes (i.e., LFGS, HFGA, ABKS-UR, etc.). As for the computational costs of key generation, the EUs in LFGS system have much less computational burden than those in HFGA and ABKS-UR schemes. This is because the theoretical costs of **KeyGen** algorithm in aforementioned three schemes are $(|S| + 1)E + 2E_T$, $(4n+1)E+E_T$, $(2n+1)E+E_T$, respectively. For example, when setting $|S| = n = 40$, the LFGS system takes 247.7 ms to generate secret keys for EUs, while the HFGS and ABKS-UR scheme take 180.3 ms, 639.6 ms to conduct the same operations, respectively. Although the HFGA scheme exerts less computational overhead on FNs than LFGS system, it brings much more computational burden to resource-limited EUs, which is impractical in actual scenarios. Besides, the result of storage costs comparison is as same as that of computational cost comparison. Hence, the LFGS system outperforms the other two schemes in terms of computational and storage costs of key generation.

In Fig. 7 (c), (d), we notice that the lightweight LFGS system can shift partial ciphertexts generation computation from EUs to the chosen FNs without loss of data confidentiality in **Enc** algorithm, which significantly reduces the computational burden of resource-constrained EUs. However, the other two schemes only enable the EUs to generate ciphertexts. Note that the computational overhead of FNs and EUs in LFGS system is slightly less than that of EUs in ABKS-UR scheme but much less than that of EUs in HFGA. Besides, the storage costs of FNs and EUs in LFGS system are much fewer than those of EUs in HFGA and ABKS-UR schemes. For example, when setting $|S| = n = 50$, the FNs and EUs in LFGS system need 533.7 s and 560.8 s to generate ciphertexts for 10000 files, respectively, while the EUs in HFGA and ABKS-UR schemes take 838.3 s and 611 s, respectively. Furthermore, the storage overhead of FNs and EUs in LFGS system is 68.2 MB (Megabytes) and 72.3 MB, respectively, and that of EUs in HFGA and ABKS-UR schemes is 147.5 MB and 144.5 MB, respectively. Therefore, the LFGS system can reduce the computational and storage costs of EUs with the help of FNs in the fog computation environment. Fortunately, the **Enc** algorithm is a one-time operation, which does not affect the search experience of EUs.

In Fig. 7 (e), (f), we demonstrate the computational and storage burden of trapdoor generation in **Trap** algorithm. The actual performance of LFGS system is similar to that of ABKS-UR scheme, and is better than that of HFGA system. Compared with the HFGA scheme, the LFGS system does not incur extra computational and storage burden when applied in fog computing. For example, when setting $|S| = n = 30$, the computational and storage costs of FNs and EUs in LFGS system are (253ms, 7.84 KB), (238 ms, 7.7 KB), respectively, while those of FNs and EUs in HFGA scheme are (507 ms, 17 KB), (494 ms, 16.9 KB), respectively. Although the ABKS-UR scheme is more efficient than HFGA scheme with regard to trapdoor generation, it does not apply to the above fog computing environment. Thus, the LFGS system not only has better performance in terms of trapdoor generation but also can gain a broad range of applications in practice.

In Fig. 7 (g), (h), we show the performance of ciphertexts retrieval in **Search** algorithm. By varying the number of attributes from 1 to 50, the computational and storage overhead of ciphertexts search is almost linear with the variable $|S| = n$. Besides, the performance of **Search** algorithm in LFGS system and ABKS-UR scheme is superior to that of HFGA scheme. For example, when setting $|S| = n = 20$, the ciphertexts search time and length of ciphertexts retrieval are 89.9 ms and 2.75 KB, respectively, while those of HFGA scheme and ABKS-UR scheme are (426 ms, 9.75 KB), (97.3 ms, 3 KB), respectively.

In Fig. 8, we just present the performance of ciphertexts decryption in LFGS system and HFGA scheme as the ABKS-UR scheme encrypts the file content by utilizing the traditional public/symmetric encryption algorithms. From TABLE 4 we know that the EUs in LFGS system and HFGA scheme just need one pairing operation P and exponentiation operation E_T to decrypt each ciphertext

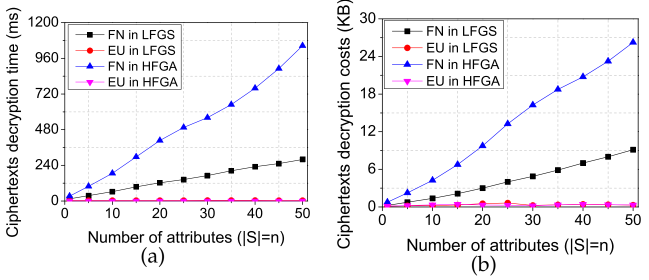

Fig. 8. Performance of **Dec** algorithm: (a) Computational costs; (b) Storage costs.

in **Dec** algorithm, and the most of computational and storage burden of ciphertexts decryption are conducted by the chosen FNs. However, the FNs in LFGS system (the computational and storage overhead is $(n+2)P$, $(n+1)|G_T|$, respectively) still has better performance than those in HFGA schemes (the computational and storage overhead is $(2n+1)P+(2n+1)E$, $(2n+1)|G|+(2n+1)G_T$, respectively). For example, when setting $|S| = n = 50$, the performance of FNs in LFGS system and HFGA scheme is (279.4 ms, 9.125 KB), (1045 ms, 26.25 KB), respectively, while that of EUs keeps almost unchanged.

To summarize, the actual performance assessments of above schemes are completely in accord with the theoretical analysis shown in TABLE 3 and TABLE 4. Compared with the HFGA scheme, the LFGS system does not incur much computational and storage burden on FNs and EUs in fog computing environment. The EUs in ABKS-UR scheme has approximately similar performance in **Enc**, **Trap** and **Search** algorithms on condition that $|S| = n$. However, in practice, the value of S is far less than that of n. Hence, the LFGS system is much more efficient than the ABKS-UR scheme as well as the HFGA scheme.

To further demonstrate the practicality of LFGS system in actual scenarios, we also conduct a series of experiments over other datasets except for Enron Email dataset, such as National Science Foundation Research Awards Abstract 1990-2003 dataset (or NSF dataset)[5] and Request For Comments database (or RFC dataset)[6] in TABLE 5 and TABLE 6, where TABLE 5 and TABLE 6 present the computational and storage costs of LFGS system and HFGA scheme in three datasets, respectively. By setting $|S| = n = 20$, we notice that the performance of LFGS system and HFGA scheme in Enron Email dataset is approximately equal to that in NSF dataset and RFC dataset. That is to say, the various datasets do not significantly affect the performance of LFGS system. Thus, our LFGS system is feasible and efficient in a broad range of applications.

7 Conclusions

In this paper, we demonstrated a Lightweight Fine-Grained Search (LFGS) system for the resource-limited EUs in fog computing. On the one hand, the basic LFGS system could greatly reduce the computational and storage burden of EUs by outsourcing partial computation and storage to the

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

TABLE 5
Computational costs in various datasets

Schemes	(LFGS, HFGA [37])								
Algorithms	**KeyGen**		**Enc**		**Trap**		**Search**	**Dec**	
Entities	FN	EU	FN	EU	FN	EU	CSP	FN	EU
Enron Dataset	(133.3, 0.924)	(97.7, 313.2)	(219716, —)	(203847, 363393)	(179, 349)	(160, 323)	(89.9, 426)	(122.8, 407)	(4.36, 0.49)
NSF Dataset	(137.1, 0.947)	(98.1, 308.4)	(219301, —)	(202721, 366592)	(187, 346)	(167, 341)	(84.2, 417)	(117.9, 403)	(4.91, 0.63)
RFC Dataset	(141.3, 1.124)	(99.2, 321.1)	(220128, —)	(208994, 367358)	(194, 353)	(179, 331)	(89.2, 421)	(125.4, 413)	(5.07, 0.62)

Notes. The symbol $(\star, \star)$ denotes the computational costs (ms) of LFGS system and HFGA scheme, respectively, where $|S| = n = 20$.

TABLE 6
Storage costs in various datasets

Schemes	(LFGS, HFGA [37])								
Algorithms	**KeyGen**		**Enc**		**Trap**		**Search**	**Dec**	
Entities	FN	EU	FN	EU	FN	EU	CSP	FN	EU
Enron Dataset	(5.02, 0.34)	(2.84, 10.83)	(28663, —)	(29186, 68095)	(5.54, 11.62)	(5.41, 11.48)	(2.75, 9.75)	(3.00, 9.75)	(0.53, 0.32)
NSF Dataset	(5.37, 0.41)	(2.93, 11.12)	(28105, —)	(28623, 67814)	(5.78, 12.15)	(5.62, 11.81)	(2.43, 9.38)	(2.83, 9.49)	(0.58, 0.43)
RFC Dataset	(5.44, 0.46)	(3.06, 11.29)	(28913, —)	(29912, 68632)	(5.83, 12.34)	(5.78, 11.95)	(2.91, 10.23)	(3.27, 9.92)	(0.62, 0.45)

Notes. The symbol $(\star, \star)$ denotes the storage costs (KB) of LFGS system and HFGA scheme, respectively, where $|S| = n = 20$.

honest-but-curious FNs without leaking sensitive information; on the other hand, the extended LFGS system could support conjunctive keyword search and attribute update to further narrow down the search scope and avoid unauthorized accesses, respectively. Furthermore, the formal security analysis showed that the LFGS system is selectively secure against CKA and CPA, and the empirical experiments using a real-world dataset illustrated the efficiency and feasibility of LFGS system in fog computing.

As a part of our future work, we will continue to concentrate on expressive search including fuzzy keyword search, semantic keyword search, and so on. Besides, the secure channel utilized in our LFGS system should be eliminated as secure channel will incur high communication burden. Hence, we still need to further improve the efficiency of LFGS system so that it can be applied in various schemes.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. 61702404, No. 61702105, No. 61672413, No. 61472310), the China Postdoctoral Science Foundation Funded Project (No. 2017M613080), the Fundamental Research Funds for the Central Universities (No. JB171504), the National High Technology Research and Development Program (863 Program) (No. 2015AA016007), the 111 project (No. B16037), the Key Program of NSFC (No. U1405255), and the Shaanxi Science & Technology Coordination & Innovation Project (No. 2016TZC-G-6-3).

References

[1] J. Li, W. Yao, Y. Zhang, H. Qian, and J. Han, "Flexible and fine-grained attribute-based data storage in cloud computing," *IEEE Transactions on Services Computing*, vol. 10, no. 5, pp. 785–796, 2017.

[2] H. Yan, J. Li, J. Han, and Y. Zhang, "A novel efficient remote data possession checking protocol in cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 1, pp. 78–88, 2017.

[3] C. Stergiou, K. E. Psannis, B. G. Kim, and B. Gupta, "Secure integration of iot and cloud computing," *Future Generation Computer Systems*, vol. 78, pp. 964–975, 2018.

[4] X. Liu, R. H. Deng, Y. Yang, H. N. Tran, and S. Zhong, "Hybrid privacy-preserving clinical decision support system in fog–cloud computing," *Future Generation Computer Systems*, vol. 78, pp. 825–837, 2018.

[5] D. Wu, F. Zhang, H. Wang, and R. Wang, "Security-oriented opportunistic data forwarding in mobile social networks," *Future Generation Computer Systems*, 2017.

[6] D. Wu, S. Si, S. Wu, and R. Wang, "Dynamic trust relationships aware data privacy protection in mobile crowd-sensing," *IEEE Internet of Things Journal*, 2017.

[7] D. Wu, J. Yan, H. Wang, D. Wu, and R. Wang, "Social attribute aware incentive mechanism for device-to-device video distribution," *IEEE Transactions on Multimedia*, vol. 19, no. 8, pp. 1908–1920, 2017.

[8] J. Zhou, Z. Cao, X. Dong, and A. V. Vasilakos, "Security and privacy for cloud-based iot: challenges," *IEEE Communications Magazine*, vol. 55, no. 1, pp. 26–33, 2017.

[9] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Proc. Annual International Cryptology Conference on Advances in Cryptology (CRYPTO'11)*. Springer, 2001, pp. 213–229.

[10] F. Guo, Y. Mu, W. Susilo, H. Hsing, D. S. Wong, and V. Varadharajan, "Optimized identity-based encryption from bilinear pairing for lightweight devices," *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 2, pp. 211–220, 2017.

[11] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. ACM conference on Computer and Communications Security (CCS'06)*. ACM, 2006, pp. 89–98.

[12] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symposium on Security and Privacy (S&P'07)*. IEEE, 2007, pp. 321–334.

[13] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symposium on Security and Privacy (S&P'00)*. IEEE, 2000, pp. 44–55.

[14] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'04)*, vol. 3027. Springer, 2004, pp. 506–522.

[15] Q. Zheng, S. Xu, and G. Ateniese, "Vabks: verifiable attribute-based keyword search over outsourced encrypted data," in *Proc. IEEE Conference on Computer Communications (INFOCOM'14)*. IEEE, 2014, pp. 522–530.

[16] W. Sun, S. Yu, W. Lou, Y. T. Hou, and H. Li, "Protecting your right: attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud," in *Proc. IEEE Conference on Computer Communications (INFOCOM'14)*. IEEE, 2014, pp. 226–234.

[17] J. Li, Y. Shi, and Y. Zhang, "Searchable ciphertext-policy attribute-based encryption with revocation in cloud storage," *International Journal of Communication Systems*, vol. 30, no. 1, 2017.

[18] P. Zhang, Z. Chen, J. K. Liu, K. Liang, and H. Liu, “An efficient access control scheme with outsourcing capability and attribute update for fog computing,” *Future Generation Computer Systems*, vol. 78, pp. 753–762, 2018.

[19] 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,” *Journal of medical systems*, vol. 40, no. 11, pp. 246:1–246:12, 2016.

[20] H. Cui, R. H. Deng, J. K. Liu, and Y. Li, “Attribute-based encryption with expressive and authorized keyword search,” in *Proc. Australasian Conference on Information Security and Privacy (ACISP’17)*. Springer, 2017, pp. 106–126.

[21] K. Yang and X. Jia, “An efficient and secure dynamic auditing protocol for data storage in cloud computing,” *IEEE transactions on parallel and distributed systems*, vol. 24, no. 9, pp. 1717–1726, 2013.

[22] I. Stojmenovic, S. Wen, X. Huang, and H. Luan, “An overview of fog computing and its security issues,” *Concurrency and Computation: Practice and Experience*, vol. 28, no. 10, pp. 2991–3005, 2016.

[23] H. Li, D. Liu, Y. Dai, T. H. Luan, and S. Yu, “Personalized search over encrypted data with efficient and secure updates in mobile clouds,” *IEEE Transactions on Emerging Topics in Computing*, vol. 6, no. 1, pp. 97–109, 2018.

[24] Y. Miao, J. Ma, and Z. Liu, “Revocable and anonymous searchable encryption in multi-user setting,” *Concurrency and Computation: Practice and Experience*, vol. 28, no. 4, pp. 1204–1218, 2016.

[25] Y. Miao, J. Liu, and J. Ma, “Efficient keyword search over encrypted data in multi-cloud setting,” *Security and Communication Networks*, vol. 9, no. 16, pp. 3808–3820, 2016.

[26] Y. Yang and M. Ma, “Conjunctive keyword search with designated tester and timing enabled proxy re-encryption function for e-health clouds,” *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 4, pp. 746–759, 2016.

[27] H. Zhu, Z. Mei, B. Wu, H. Li, and Z. Cui, “Fuzzy keyword search and access control over ciphertexts in cloud computing,” in *Proc. Australasian Conference on Information Security and Privacy (ACISP’17)*. Springer, 2017, pp. 248–265.

[28] Y. Miao, J. Ma, X. Liu, Q. Jiang, J. Zhang, L. Shen, and Z. Liu, “Vcksm: Verifiable conjunctive keyword search over mobile e-health cloud in shared multi-owner settings,” *Pervasive and Mobile Computing*, vol. 40, pp. 205–219, 2017.

[29] H. Li, D. Liu, Y. Dai, T. H. Luan, and X. S. Shen, “Enabling efficient multi-keyword ranked search over encrypted mobile cloud data through blind storage,” *IEEE Transactions on Emerging Topics in Computing*, vol. 3, no. 1, pp. 127–138, 2015.

[30] C. Chen, X. Zhu, P. Shen, J. Hu, S. Guo, Z. Tari, and A. Y. Zomaya, “An efficient privacy-preserving ranked keyword search method,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 4, pp. 951–963, 2016.

[31] C. Zuo, J. Shao, G. Wei, M. Xie, and M. Ji, “Cca-secure abe with outsourced decryption for fog computing,” *Future Generation Computer Systems*, vol. 78, pp. 730–738, 2018.

[32] Y. Jiang, W. Susilo, Y. Mu, and F. Guo, “Ciphertext-policy attribute-based encryption against key-delegation abuse in fog computing,” *Future Generation Computer Systems*, vol. 78, pp. 720–729, 2018.

[33] J. Lai, R. H. Deng, C. Guan, and J. Weng, “Attribute-based encryption with verifiable outsourced decryption,” *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 8, pp. 1343–1354, 2013.

[34] 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 on Asia Conference on Computer and Communications Security (AsiaCCS’17)*. ACM, 2017, pp. 230–240.

[35] J. Li, X. Lin, Y. Zhang, and J. Han, “Ksf-oabe: outsourced attribute-based encryption with keyword search function for cloud storage,” *IEEE Transactions on Services Computing*, vol. 10, no. 5, pp. 715–725, 2017.

[36] Y. Fan and Z. Liu, “Verifiable attribute-based multi-keyword search over encrypted cloud data in multi-owner setting,” in *Proc. IEEE International Conference on Data Science in Cyberspace (DSC’17)*. IEEE, 2017, pp. 441–449.

[37] M. Xiao, J. Zhou, X. Liu, and M. Jiang, “A hybrid scheme for fine-grained search and access authorization in fog computing environment,” *Sensors*, vol. 17, no. 6, p. 1423, 2017.

[38] L. Fang, W. Susilo, C. Ge, and J. Wang, “A secure channel free public key encryption with keyword search scheme without random oracle,” in *Proc. International Conference on Cryptology and Network Security (CANS’09)*. Springer, 2009, pp. 248–258.

Yinbin Miao 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.

Jianfeng Ma received the Ph.D. degree in computer software and telecommunication engineering from Xidian University, Xi’an, China, in 1995. From 1999 to 2001, he was a research fellow with Nanyang Technological University of Singapore. He is currently a professor with the Department of Computer Science and Technology, Xidian University, Xi’an, China. His current research interests include information and network security, wireless and mobile computing systems, and computer networks.

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 postdoctoral fellow with the Department of Information System, Singapore Management University, Singapore. His research interests include applied cryptography and big data security. He is a member of the IEEE.

Jian Weng received the Ph.D. degree in computer science and engineering from Shanghai Jiao Tong University, China, in 2008. From April 2008 to March 2010, he was a post-doctoral fellow in the School of Information Systems, Singapore Management University. Currently, he is a professor with the College of Information Science and Technology, College of Cyber Security, Jinan University. His research interests include cryptography, information security, and artificial intelligence.

Hongwei Li (M’12) 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.

Hui Li (M’10) received the Ph.D. degrees from Xidian University in 1998, respectively. In 2009, he was with the Department of ECE, University of Waterloo as a visiting scholar. Since 2005, he has been a professor in the Department of Telecommunication Engineering, Xidian University, Xi’an, China. His research interests include the areas of cryptography, security of cloud computing, wireless network security, and information theory.