Keyword Searchable Encryption with Fine-Grained Forward Secrecy for Internet of Thing Data

Rang Zhou[1], Xiaosong Zhang[1(✉)], Xiaofen Wang[1], Guowu Yang[1], and Wanpeng Li[2]

[1] Center for Cyber Security, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan, China
zhour1987@sohu.com, {johnsonzxs,guowu}@uestc.edu.cn, wangxuedou@sina.com
[2] School of Computing, Mathematics and Digital Technology, Manchester Metropolitan University, Manchester M15 6BH, UK
W.Li@mmu.ac.uk

Abstract. With the incessant development and popularization of Internet of things (IoT), the amount of the data collected by IoT devices has rapidly increased. This introduces the concerns over the heavy storage overhead to such systems. In order to relief the storage burden, a popular method is to use the outsourced cloud technology. While the massive collected IoT data is outsourced to the cloud, the security and privacy of these outsourced data is therefore of critical importance, and many researches have been done in this area. In this paper, we propose a new keyword searchable encryption system with fine-grained right revocation. In the system, each IoT device's data are stored in a special document. Thus the data owner can revoke users' search rights at fine-grained document level by setting new random number in each time period. Especially, to realize search right revocation, re-encryption operations on keyword cipheretexts are not needed in our scheme. Then, we instantiate a valid construction in practical application and discuss the security properties in the construction. Our performance evaluations show that the proposed construction is efficient.

Keywords: Searchable encryption · Data sharing
Fine-grained forward secrecy · Internet of Things

1 Introduction

Internet of Things (IoT) introduces an emerging data collection paradigm that aims to obtain IoT data from pervasive things through distributed networks. The emerging paradigm creates new growth of economy, and more researchers pay close attention to the technology. In practical IoT, massive physical devices are connected with each other through wireless or wired network to collect data, such as smart homes and cities, smart vehicle networks, industrial manufacturing and

J. Vaidya and J. Li (Eds.): ICA3PP 2018, LNCS 11337, pp. 288–302, 2018.
https://doi.org/10.1007/978-3-030-05063-4_23

smart environment monitoring. Moreover, data analysis is essential to enhance IoT service, for example the process optimization in industrial manufacturing. Therefore, to meet the requirement of data analysis, data sharing is deployed in IoT. However, with the growth-up scale of IoT, it is not efficient to store and search data directly from each resource-constrained IoT device.

To handle the lack of computing and storage power in IoT device, cloud-assisted method is introduced. This provides an economic and practicable platform with rich computing and storage resource and low cost in data sharing. Each IoT device, which is controlled by data owner, uploads shared data to the cloud server in encrypted format. The cloud server receives a search query which is submitted by an authorized user, and does query match and responds the corresponding encrypted data to the user.

Moreover, for a practical data sharing application in IoT, the users, who are authorized to search and access data, always update their search right dynamically. Therefore, dynamical right update has become a concern in date sharing. The authorization operation can be completed easily, and most studies paid close attention to the right revocation in the cloud assisted system. Especially, to maintain the property of data forward security, the users could not search the encrypted data by the old trapdoors computed from the old secret keys if their search privileges are withdrawn.

In the past, user's right revocation in IoT data sharing scheme generally is not done at fine-grained level. Such schemes [19,21–23] are not suitable for many cases. For example, a user transfers between different projects inside the same company. The manager might wish to reclaim the search right of project A from the staffs who have transferred to project B. Previous data sharing schemes can not meet this need, because after the search right revocation, the user is not able to search all the documents any more. To meet the requirement, search right revocation at document level is needed, where the data owner can reclaim each users' search privilege in document level.

Our Contribution. In this paper, we aim to solve the fine-grained search right revocation problem for IoT data in keyword searchable encryption system, where forward secrecy is met. We proposed a new keyword searchable encryption scheme with fine-grained search right revocation at document level. The main contributions include:

- We analyze the security requirements of public key keyword searchable encryption scheme with dynamic right revocation in an IoT data sharing system. To achieve forward secrecy for IoT data, we propose a new keyword searchable encryption scheme, which prompts a fine-grained search right revocation at document level.
- We give a concrete construction to complete the search right revocation function for IoT data sharing system. In the construction, we design an user right revocation method without re-encryption operations on keyword ciphertexts, which is more efficient.

– We analyze the performance evaluation of our scheme. The evaluation result shows that the scheme is practical for IoT applications.

1.1 Related Work

Since cloud-assisted technology is provided to improve IoT, the researchers are focus on data management stored in cloud server server. Moreover, data privacy becomes the concern in IoT data storage system. To ensure data security, the data owner needs to encrypt the data before sending it to the cloud server. In order to crease the quality of service, data manager needs to share IoT data with authorized users for data analysis. Keyword searchable encryption, which meets the requirement in data management, is proposed for secure data sharing. Therefore, the design of lightweight keyword searchable encryption construction is a challenge in IoT data sharing application.

To reduce the cost of massive key management in searchable symmetric encryption construction [1], the concept of public key encryption with keyword search (PEKS) is introduced by [2]. Moreover, multi-key searchable encryption and forward secrecy searchable encryption are presented to complete the functions of fine-grained document sharing and user right revocation in practical system.

Multi-key Searchable Encryption. The multi-key searchable encryption framework is designed by Popa et al. [5], and the first web application are built on Mylar in [6]. Only one trapdoor is provided to server, and keyword search match is completed by in different documents, which is encrypted by different keys. Moreover, an new security model for multi-owner searchable encryption are proposed by Tang [7] in this framework. Liu et al. [8] design a scheme in data sharing. Unfortunately, low performance problem is inevitable, because the trapdoor size is linear with the number of documents, in their scheme. To complete provable security, Rompay et al. [9] constructed a new scheme on proxy method. However, heavy overhead does not be ignored on proxy.

Combining with the study of [10], Cui et al. [11] proposed a new key-aggregate searchable encryption scheme, where an aggregation method on file keys is introduced to compute authorization key generation. One user generates a trapdoor to complete keyword search in all authorized file set to this user. However, Kiayias et al. [12] design two key guessing attacks to the study of [11]. Li et al. [14] and Liu et al. [15] proposed their two improved schemes to maintain data verification and multi-owner functions. However, in [14] and [15], the similar drawbacks as in [11] can be found. Kiayias et al. [12] proposed their improved scheme for the study of [11], and more communication and computation are inevitable. To reduce the computation and communication cost, [13] designs a new scheme.

Forward Secrecy Searchable Encryption. In recent studies [16–18], to achieve the function of fine-grained access control in searchable encryption, many schemes, in different scenario, are proposed based on attribute-based encryption. The study [17] implements user revocation in a practical multi-user and multi-owner scenario. In [18], a key-policy and a ciphertext-policy key searchable

encryption scheme, where the data owner manages users' search right, are proposed. To meet fine-grained search right management, Shi et al. [16] proposed an attribute-based keyword searchable encryption scheme.

However, all above schemes are constructed applying attribute based encryption, and the search right revocation is achieved from the attribute revocation, where re-encryption is needed to generate new keyword ciphertexts.

To adapt to more practical application, researchers introduced the random number method in computing the keys in each discrete time period without attribute-based encryption. The first scheme [19] is constructed using BLS short signature [20], and the keyword encryption is separated two phases. The first one is completed using a corresponding complementary key maintained by the server. The other one is executed by data sender to generate keyword ciphertexts. Dong et al. [21,22] separated a search key to two parts, where one is users' secret keys and the other one is re-encryption key stored on the server. To reduce the communication and computation cost on proxy, Wang et al. [23] designed a new forward secrecy searchable encryption without proxy, which is similar with the study of [22]. However, this scheme is used in the peer-to-peer scenario. Moreover, the above constructions can not provide the user revocation in fine-grained right management at document level.

2 Preliminaries

2.1 Bilinear Pairing

Bilinear Map. Let two multiplicative cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ be of the same prime order p, and g, h be the generators of $\mathbb{G}_1$. A bilinear pairing e is a map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ with the following properties:

1. Bilinearity: $e(g^{r_1}, h^{r_2}) = e(g, h)^{r_1 r_2}$ for all $g, h \in \mathbb{G}_1$ and $r_1, r_2 \in \mathbb{Z}_p^*$.
2. Non-degeneracy: $e(g, g) \neq 1$.
3. Computability: for any $g, h \in \mathbb{G}_1$, $e(g, h)$ can be computed efficiently.

2.2 Complexity Assumptions

Computational Diffie-Hellman (CDH) Assumption. Let $\mathbb{G}_1$ be bilinear groups of prime order p, given $g, g^{Z_1}, g^{Z_2} \in \mathbb{G}_1$ as input, it is infeasible to compute $g^{Z_1 Z_2} \in \mathbb{G}_1$, where $Z_1, Z_2 \in \mathbb{Z}_p^*$.

Variational Computational Diffie-Hellman (CDH) Assumption. Let $\mathbb{G}_1$ be bilinear groups of prime order p, given $g, g^{Z_3}, g^{Z_1 Z_5}, g^{Z_1 Z_4}, g^{Z_2 Z_4}, g^{Z_1 Z_3 Z_5} \in \mathbb{G}_1$ as input, it is infeasible to compute $g^{Z_2 Z_4 Z_5} \in \mathbb{G}_1$, where $Z_1, Z_2, Z_3, Z_4, Z_5 \in \mathbb{Z}_p^*$.

3 System Model

3.1 System Architecture

The Fig. 1 shows the system architecture, which is consisted of data owner, cloud server, user and IoT nodes. The system role of each party is described as follows.

Data Owner. The data owner is keys generator and data manager in the system. The data owner maintains a users list to generate and distribute all authorized keys to each user and encryption secret key to IoT nodes. Users' search right managements are maintained to achieve the function of fine-grain right revocation.

Cloud Server. The cloud server provides the storage and search service for IoT data management. Especially, the cloud server is "honest but curious", which completes search queries honestly and does not modify stored information maliciously. Moreover, it does not collude with other parties to guess the keyword information from ciphertexts and search queries.

Users. The users are registered in the data owner's list and receive authorized keys from the data owner. The users can generate query trapdoor to search the data on the cloud server.

IoT Nodes. The IoT nodes are data collection nodes for a IoT system. Real-time data to the cloud server for data storage are handled and sent to the cloud server by the IoT nodes. To maintain the data privacy, the IoT nodes achieve the function of encryption for collection data. Moreover, in most actual scenes, encryption operations are executed in resource constrained IoT nodes. Therefore, designed scheme is focus on less computation cost in the system for IoT nodes.

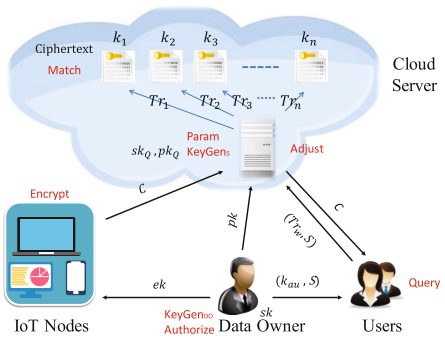


Fig. 1. The fine-grained revocation searchable encryption system

3.2 System Definition

Definition 1. *As shown in Fig. 1, the keyword searchable encryption system definition with revocation capability in data sharing consists of the following eight algorithms:*

- **Param**(ξ)*: The algorithm takes security parameter* ξ *as input, and generates system global parameter* $\mathcal{GP}$*.*
- **KeyGen**$_{\mathbf{S}}(\mathcal{GP})$*: The cloud server takes system parameter* $\mathcal{GP}$ *as input, and outputs server's public and secret key pair* (pk_Q, sk_Q)*.*
- **KeyGen**$_{\mathbf{DO}}(\mathcal{GP}, \tau)$*: The data owner takes system parameter* $\mathcal{GP}$*, time period* τ_b *as input, and outputs his/her public and secret key pair* (pk, sk) *and IoT node secret key* ek*.*
- **Authorize**$(\mathcal{GP}, \tau, sk, S)$*: The data owner takes the system parameter* $\mathcal{GP}$*, time period* τ_b*, data owner's private key* sk *and authorized document set* S *as input, and outputs authorization key* k_{au}*. The data owner sends* (k_{au}, S) *to each corresponding user through a secure channel.*
- **Encrypt**$(\mathcal{GP}, F_i, pk_Q, ek, w)$*: For a document* F_i $(i \in \{1, \cdots, n\})$*, IoT node takes system parameter* $\mathcal{GP}$*, IoT node secret key* ek*, server's public key* pk_Q*, the document number* F_i*, keywords* w *as input, and generates ciphertexts* C*.*
- **Query**$(\mathcal{GP}, \tau, k_{au}, pk, sk_Q, w)$*: An authorized user takes system parameter* $\mathcal{GP}$*, time period* τ*, authorization key* k_{au}*, data owner's public key* pk*, server's public key* pk_Q*, a keyword* w *as input, and generates query trapdoor* Tr_w*.*
- **Adjust**$(\mathcal{GP}, \tau, pk, S, Tr_w)$*: The cloud server takes system parameter* $\mathcal{GP}$*, data owner's public key* PK*, authorized document set* S*, query trapdoor* $Tr_w = \mathsf{Query}(\mathcal{GP}, k_{au}, w)$ *as input, and outputs each adjust trapdoor* Tr_i *for each* F_i *in* S*.*
- **Match**$(\mathcal{GP}, \tau, pk, sk_Q, S, Tr_i, C)$*: A deterministic algorithm runs by the cloud server, which takes system parameter* $\mathcal{GP}$*, time period* τ*, data owner's public key* pk*, server's private key* sk_Q*, authorized document set* S*, an adjust trapdoor* $Tr_i = \mathsf{Adjust}(\mathcal{GP}, pk, S, Tr_w)$*, a ciphertext* $C = \mathsf{Encrypt}(\mathcal{GP}, F_i, pk_Q, sk, w)$ *as input, and outputs a symbol "True" if* C *contains* w*; Otherwise, "False".*

3.3 Security Requirement

To maintain the security of keyword searchable encryption system, keyword confidentiality and trapdoor privacy must be considered. Further, a keyword searchable encryption construction with user search right revocation function has the ability to distinguish the unrevoked users and revoked users. The correctness is satisfied for unrevoked users and the forward secrecy is maintained for revoked user.

Correctness. A keyword searchable encryption system is correct if it satisfies that each authorized user who has the authorized key can perform a successful keyword search.

Keyword Confidentiality. A keyword searchable encryption system maintains keyword confidentiality if it satisfies that only the authorized users can complete the keyword search, and unauthorized users are incapable of learning the privacy information of the stored keyword ciphertexts.

Query Privacy. A keyword searchable encryption system maintains query privacy if it satisfies that only the authorized users can generate a trapdoor from a keyword, and unauthorized users and the honest-but-curious cloud server are incapable to determine a keyword from the submitted query trapdoor.

Forward Secrecy. A keyword searchable encryption system is forward secrecy if it satisfies that the data owner can delete a user and revoke his ability from the system. Moreover, for each revoked user, the data owner can support more fine-grained search right revocation, which is corresponding to every document.

4 The Designed Scheme and Security Analysis

4.1 The Designed Scheme

Param(ξ)**.** The algorithm works as follows:

1. Take the security parameter ξ as input and generate a bilinear group parameters $(p, \mathbb{G}_1, \mathbb{G}_2, e)$;
2. Set the maximum number of documents as n for a data owner and the keyword space as m.
3. Choose a generator $g \in \mathbb{G}_1$ and a collision resistant hash function $H : \{0,1\}^* \to \mathbb{Z}_p^*$.

The system parameters are published as $(p, \mathbb{G}_1, \mathbb{G}_2, e, g, n, m, H : \{0,1\}^* \to \mathbb{Z}_p^*)$.

KeyGen$_{\mathbf{S}}$. The cloud server randomly chooses a random secret key $\beta_1 \in \mathbb{Z}_p^*$ and computes $u = g^{\beta_1} \in \mathbb{G}_1$. The server's private key and public key are $(sk_Q, pk_Q) = (\beta_1, u)$.

KeyGen$_{\mathbf{DO}}$ $(\mathcal{GP})$**.** At time period τ_b $(b = 1, \cdots, \rho)$, the data owner randomly chooses $d_b \in \mathbb{Z}_q^*$. The algorithm performs the following steps:

1. Randomly choose an element $\alpha \in \mathbb{Z}_p^*$, and compute secret keys $g_i = g^{(\alpha)^i} \in \mathbb{G}_1$ for $i = (1, 2, \ldots, n)$.
2. Randomly choose secret keys $\beta_2, \gamma_1, \gamma_2 \in \mathbb{Z}_p^*$, and compute the public parameters $v = g^{\beta_2} \in \mathbb{G}_1$, $h_{1,i,b} = g_i^{\gamma_1 \cdot d_b} \in \mathbb{G}_1$ for $i = (1, 2, \ldots, n)$, and $h_{2,i,b} = g_i^{\gamma_2 \cdot d_b} \in \mathbb{G}_1$ for $i = (1, 2, \ldots, n, n+1, \ldots, 2n)$.
3. Compute IoT node secret key $ek = (ek_1, ek_2) = (u^{\gamma_1}, v^{\gamma_1})$.
4. Destroy α.

The data owner's private key $sk = (\beta_2, \gamma_1, \gamma_2, \{g_i\}_{i=1,2,\ldots,n})$ is kept secretly and public key $pk = (v, \{h_{1,i,b}\}_{i=1,2,\ldots,n}, \{h_{2,i,b}\}_{i=1,2,\ldots,n,n+1,\ldots,2n})$ is stored on cloud

server, respectively. Moreover, the data owner distributes the secret key ek to each IoT node.

Authorize(sk, S)**.** The data owner takes document subset $S \subseteq \{1, \ldots, n\}$ as input, computes the authorized key: $k_{au,b} = \prod_{j \in S} g_{n+1-j}^{\beta_2 \cdot d_b}$. The data owner securely sends $(k_{au,b}, S)$ to users.

Encrypt(pk_Q, pk, ek, F_i, l)**.** Each encryption node encrypts keyword w_l, $(l \in \{1, \ldots, m\})$ to the corresponding document $F_i, (i \in \{1, \ldots, n\})$, and uploads the ciphertexts to the cloud server. The encrypt node randomly chooses $t_{i,l} \in \mathbb{Z}_p^*$ and computes ciphertext C as:

$$\begin{aligned} C &= (c_{1,i,l}, c_{2,i,l}, c_{3,i,w_l}) \\ &= (ek_1^{t_{i,l}}, ek_2^{t_{i,l}}, (v^{H(w_l)} h_{2,i})^{t_{i,l}}) \\ &= (g^{\gamma_1 \beta_1 t_{i,l}}, g^{\beta_2 \gamma_1 t_{i,l}}, (g^{\beta_2 H(w_l)} g_i^{\gamma_2})^{t_{i,l}}) \end{aligned}$$

Query$(k_{au,b}, u, v, w_l)$**.** User chooses a random $x \in Z_p^*$, and generates query trapdoor $Tr_b = (Tr_{1,b}, Tr_2) = ({k_{au,b}}^{H(w_l)} v^x, u^x)$. The user sends (Tr_b, S) to the cloud server.

Adjust(pk, i, S, Tr)**.** The cloud server runs the adjust algorithm to compute the discrete trapdoors $Tr_{1,i}$ for each document F_i as:

$$Tr_{1,i,b} = Tr_{1,b} \cdot \prod_{j \in S, j \neq i} h_{2,(n+1-j+i),b} = Tr_{1,b} \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i}^{\gamma_2 \cdot d_b}.$$

Match$(Tr_{1,i,b}, Tr_2, S, pk, sk_Q, C)$**.** The cloud server does keyword search match as follows:

1. Compute $pub_b = \prod_{j \in S} h_{1,(n+1-j),b} = \prod_{j \in S} g_{n+1-j}^{\gamma_1 \cdot d_b}$ for the subset S;
2. Check the equation:

$$\frac{e(pub_b, c_{3,i,w_l})^{\beta_1} \cdot e(c_{2,i,l}, Tr_2)}{e(Tr_{1,i,b}, c_{1,i,l})} \stackrel{?}{=} e(h_{2,n+1,b}, c_{1,i,l})$$

If the result holds, outputs "True". Otherwise, "False".

If only one document F_j is authorized in S, $k_{au} = g_j^{\beta_2 d_b}$. The cloud server does not run the **Adjust** algorithm.

4.2 Security Analysis

Assuming that the public cloud server is "honest-but-curious" and does not colludes with the the revoked users. We analyze the security properties of our scheme including correctness, keyword confidentiality, query privacy and forward secrecy.

Theorem 1. Correctness: *Each authorized user is able to retrieve the encrypted documents, which are authorized to search.*

Proof. We show the correctness of our construction in the time period τ_b ($b = 1, \cdots, \rho$) as

$$
\begin{aligned}
& \frac{e(pub_b, c_{3,i,w_l})^{\beta_1} \cdot e(c_{2,i,l}, Tr_2)}{e(Tr_{1,i,b}, c_{1,i,l})} \\
= & \frac{e(\prod_{j\in S} g_{n+1-j}^{\gamma_1 \cdot d_b}, (g^{\beta_2 H(w_l)} \cdot g_i^{\gamma_2})^{t_{i,l}})^{\beta_1} \cdot e(g^{\beta_2\gamma_1 t_{i,l}}, g^{\beta_1 x})}{e(Tr_{1,b} \cdot \prod_{j\in S, j\neq i} g_{n+1-j+i}^{\gamma_2 \cdot d_b}, g^{\gamma_1\beta_1 t_{i,l}})} \\
= & \frac{e(\prod_{j\in S} g_{n+1-j}^{\gamma_1 \cdot d_b}, g^{\beta_2 H(w_l)\beta_1 t_{i,l}}) \cdot e(\prod_{j\in S} g_{n+1-j}^{\gamma_1 \cdot d_b}, g_i^{\gamma_2\beta_1 t_{i,l}}) \cdot e(g^{\beta_2\gamma_1 t_{i,l}}, g^{\beta_1 x})}{e((\prod_{j\in S} g_{n+1-j}^{\beta_2 \cdot d_b})^{H(w_l)} \cdot v^x \cdot \prod_{j\in S, j\neq i} g_{n+1-j+i}^{\gamma_2 \cdot d_b}, g^{\gamma_1\beta_1 t_{i,l}})} \\
= & \frac{e(\prod_{j\in S} g_{n+1-j}^{\gamma_1 \cdot d_b}, g^{\beta_2 H(w_l)\beta_1 t_{i,l}}) \cdot e(\prod_{j\in S} g_{n+1-j}^{\gamma_1 \cdot d_b}, g_i^{\gamma_2\beta_1 t_{i,l}}) \cdot e(g^{\beta_2\gamma_1 t_{i,l}}, g^{\beta_1 x})}{e((\prod_{j\in S} g_{n+1-j}^{\beta_2 \cdot d_b})^{H(w_l)}, g^{\gamma_1\beta_1 t_{i,l}}) \cdot e(g^{\beta_2 x}, g^{\gamma_1\beta_1 t_{i,l}}) \cdot e(\prod_{j\in S, j\neq i} g_{n+1-j+i}^{\gamma_2 \cdot d_b}, g^{\gamma_1\beta_1 t_{i,l}})} \\
= & \frac{e(\prod_{j\in S} g_{n+1-j+i}, g)^{\gamma_1 \cdot d_b \gamma_2 \beta_1 t_{i,l}}}{e(\prod_{j\in S, j\neq i} g_{n+1-j+i}, g)^{\gamma_2 \cdot d_b \gamma_1 \beta_1 t_{i,l}}} \\
= & \ e(g_{n+1}, g)^{\gamma_1\gamma_2 \cdot d_b \beta_1 t_{i,l}} \\
= & \ e(g_{n+1}^{\gamma_2 \cdot d_b}, g^{\gamma_1\beta_1 t_{i,l}}) \\
= & \ e(h_{2,n+1,b}, c_{1,i,l}).
\end{aligned}
$$

Theorem 2. Keyword Confidentiality: *The proposed scheme is security on keyword confidentiality to resist the attack from unauthorized users.*

Proof. The unauthorized users are curious to the keyword in keyword ciphertexts C and become attacker A_1. It may obtain some information to launch an attack. A_1 can obtain the stored information including public parameters, other documents search keys $k_j (i \neq j)$, keyword ciphertexts C.

Assuming that the unauthorized users want to guess the keyword w_θ from keyword ciphertexts $C = (c_{1,i,\theta}, c_{2,i,\theta}, c_{3,i,w_\theta}) = (u^{\gamma_1 t_{i,\theta}}, g^{\beta_2\gamma_1 t_{i,\theta}}, (g^{\beta_2 H(w_\theta)} g_i^{\gamma_2})^{t_{i,\theta}})$ of document F_i.

- A_1 retrieves the partial number $(g_i^{\gamma_2})^{t_{i,\theta}}$ from C. A_1 maintains $u^{t_{i,\theta}\gamma_1}, v^{t_{i,\theta}\gamma_1}$, $u, v, g_i^{\gamma_2}, g_i^{\gamma_1}$ and wants to obtain $g_i^{\gamma_2 t_{i,\theta}}$. $u, v, g_i \in \mathbb{G}_1$ and $v = u^{z_1}, g_i = u^{z_2}$, where $z_1, z_2 \in \mathbb{Z}_p^*$. A_1 maintains $u^{t_{i,\theta}\gamma_1}, u^{z_1 t_{i,\theta}\gamma_1}, u, u^{z_1}, u^{z_2\gamma_2}, u^{z_2\gamma_1}$ and wants to obtain $u^{z_2\gamma_2 t_{i,\theta}}$. Therefore, if A_1 can obtain the value of $g_i^{\gamma_2 t_{i,\theta}}$ in this case, A_1 can solve Variational Computational Diffie-Hellman problem.
- A_1 retrieves the partial number $(g^{\beta_2 H(w_\theta)})^{t_{i,\theta}} = (g^{\beta_2 t_{i,\theta}})^{H(w_\theta)}$ from C. A_1 needs the value of $g^{\beta_2 t_{i,\theta}} = v^{t_{i,\theta}}$. A_1 maintains $v^{t_{i,\theta}\gamma_1}, v$ and wants to obtain $v^{t_{i,\theta}}$. $v \in \mathbb{G}_1$ and $v = z^{\gamma_1^{-1}}$, where $z \in \mathbb{G}_1$. A_1 maintains $z^{t_{i,\theta}}, z^{\gamma_1^{-1}}$ and wants to obtain $z^{t_{i,\theta}\gamma_1^{-1}}$. Therefore, if A_1 can obtain the value of $v^{t_{i,\theta}}$ in this case, A_1 can solve Computational Diffie-Hellman problem.

Therefore, the attacker A_1 does not distinguish w_θ to achieve the attack goal.

Theorem 3. Query Privacy: *The proposed scheme is security on query trapdoor privacy to resist the attack from honest-but-curious cloud server and unauthorized users, who do not have search right of attacked document F_i.*

Proof. (1) The honest-but-curious cloud server is curious to the keyword information in query trapdoor and becomes an attacker A_2. It may obtain some information to launch an attack. A_2 can obtain the stored information including public parameters, server secret key β_1, other documents search keys $k_j (i \neq j)$, submitted query trapdoor Tr_b.

Assuming that the server wants to guess the keyword w_θ from trapdoor $Tr_b = (Tr_{1,b}, Tr_2) = ({k_{au,b}}^{H(w_\theta)} v^x, u^x)$, where S is the authorized search set and $F_i \in S$, and becomes a attacker A_2. A_2 does the guess attacks as following:

- A_2 retrieve the partial number v^x from the Tr_b. A_2 maintains u, u^x, v and wants to obtain v^x. $u, v \in \mathbb{G}_1$ and $v = u^z$, where $z \in \mathbb{Z}_p^*$. A_1 maintains u, u^x, u^z and wants to obtain u^{xz}. Therefore, if A_2 can obtain the value of v^x in this case, A_2 can solve Computational Diffie-Hellman problem.
- A_2 computes the ${k_{au,b}}^{H(w_\theta)}$ from the secret key $g_{n+1-i}^{\beta_2 \cdot d_b}$. However, for F_i, A_2 only has a negligible probability to get the secret search key $g_{n+1-i}^{\beta_2 \cdot d_b}$, and the data owner's private keys $\beta_2, \gamma_1, \gamma_2$. Moreover, A_2 computes $Tr_{1,i,b} = Tr_{1,b} \cdot \prod_{j \in S, j \neq i} h_{2,(n+1-j+i),b}$, and get the discrete trapdoor $Tr_{1,i,b}$ for the file F_i. The computation is executed by the cloud server and leak no any information to A_2 to determine w_θ in the query trapdoor.

Therefore,A_2 does not distinguish w_θ to achieve the attack goal.

(2) The unauthorized users are curious to the keyword in submitted trapdoor and become attacker A_1. It may obtain some information to launch an attack. A_1 can obtain the stored information including public parameters, other documents search keys $k_j (i \neq j)$, submitted query trapdoor Tr_b.

Comparing with A_2, A_1 has weak capability because of the lack of server secret key. Therefore, A_1 does not achieve the attack goal.

Theorem 4. Forward Secrecy: *To maintain the fine-grained forward secrecy, the system manages the search right for each document. Each revoked user can not retrieve the special encrypted documents, which are revoked from his search right.*

Proof. In the time period τ_{bb}, the revoked users could not get the new short time authorized key from our scheme. Therefore, he only has the ability to generate and send the old trapdoor $Tr_b = (Tr_{1,b}, Tr_2)$ from the old short time authorized key in time period τ_b $(b \neq bb)$. The server does the adjust and match algorithms as follows:

1. Compute $Tr'_{1,i,bb} = Tr_{1,b} \cdot \prod_{j \in S, j \neq i} h_{2,(n+1-j+i),bb} = Tr_{1,b} \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i}^{\gamma_2 \cdot d_{bb}}$
2. Compute the $pub_{bb} = \prod_{j \in S} h_{1,(n+1-j),bb} = \prod_{j \in S} g_{n+1-j}^{\gamma_1 \cdot d_{bb}}$ based on subset S.
3. Test the equation:

$$\frac{e(pub_{bb}, c_{3,i,w_l})^{\beta_1} \cdot e(c_{2,i,l}, Tr_2)}{e(Tr'_{1,i,bb}, c_{1,i,l})} \stackrel{?}{=} e(h_{2,n+1,bb}, c_{1,i,l})$$

Due to $Tr'_{1,i,bb} \neq Tr_{1,i,bb}$, the test equation does not hold.

From the above analysis, we show that the revoked users could not search the specific encrypted document by submitting the trapdoor generated from an old short time authorized key. Therefore, the forward secrecy is achieved in our scheme.

5 Performance Analysis

5.1 Implementation Details

The performance is constructed by the basic cryptographic operations in pairing computation. Two different settings are considered: one is in JAVA on smart phone, which has a 64-bit 8 core CPU processor (4 core processor runs at 1.5 GHz and 4 core processors runs at 1.2 GHz), 3 GB RAM with Android 5.1.1. The other is in C on computer, which has Intel Core i3-2120 CPU @3.30 GHz, 4.00 GB RAM with windows7 64-bits operation system. JPBC and PBC library [3] are used to implement the cryptographic operations for smart phone and computer, respectively. We choose the type A elliptic curve, which is shown as $E : y^2 = x^3 + x$. To maintain the security and efficiency, our experiment is conducted as $|Z_p| = 160$ bits, $|\mathbb{G}_1| = 1024$ bits and $|\mathbb{G}_2| = 1024$ bits. Some useful experiment results [4] about pair computation are shown in Table 1.

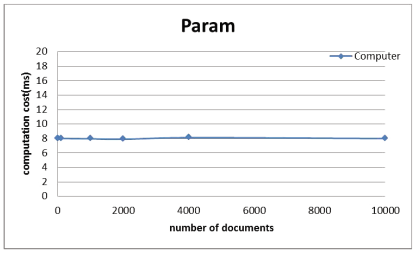
Table 1. The computation time on different platforms (ms)

Computation	Smart phone	Computer
Bilinear pairing $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$	195.11	18.03
Exponentiation on group $\mathbb{G}_1$	90.12	9.18
Exponentiation on group $\mathbb{G}_2$	33.4	2.78

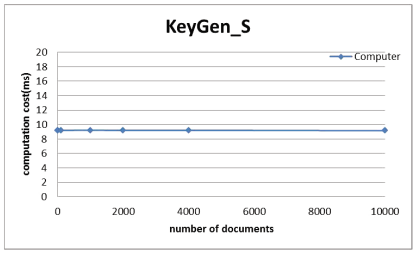
5.2 Experiment Evaluation

In the IoT data sharing system simulation, the cloud server is considered as computer. Moreover, two different simulations are implemented for data owners on smart phone and computer. Smart phone and computer are instantiated on user, encryption node and data owner. **Param**, $\mathbf{KeyGen_S}$, **Adjust** and **Match** run on the cloud server. $\mathbf{KeyGen_{DO}}$ and **Authorize** run on the data owner and **Encrypt** runs on encryption node. **Query** runs on user. Next, we discuss the performance evaluation of our construction on cloud server, data owner, encryption node and user.

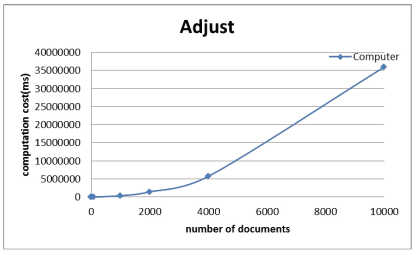
The simulation results show in Fig. 2. The time cost of algorithms on cloud server, data owner, encryption node and user are matched to the Fig. 2(a)–(d), Fig. 2(e) and (f), Fig. 2(g) and Fig. 2(h), respectively. The evaluation analysis is shown as follows:



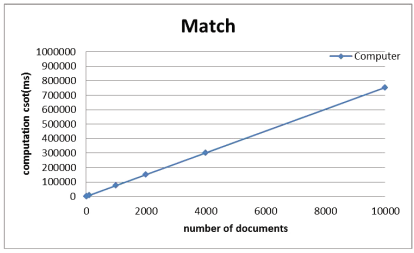
(a) Time cost of **Param**



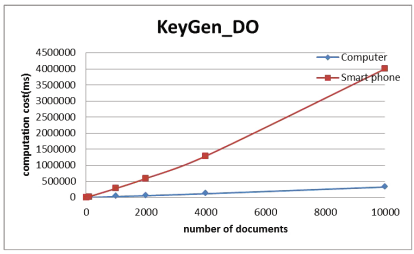
(b) Time cost of $\mathbf{KeyGen_S}$



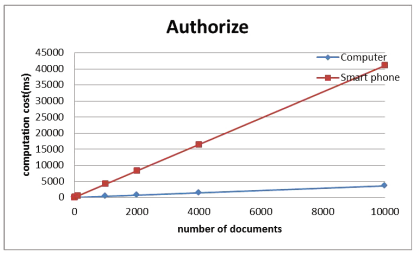
(c) Time cost of **Adjust**



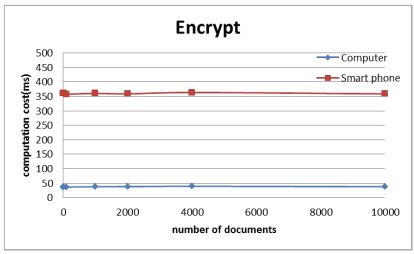
(d) Time cost of **Match**



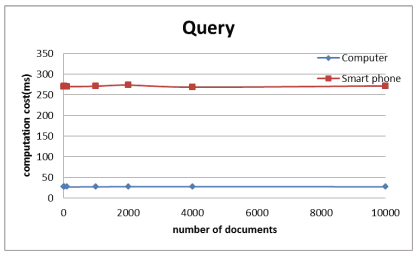
(e) Time cost of $\mathbf{KeyGen_{DO}}$



(f) Time cost of **Authorize**



(g) Time cost of **Encrypt**



(h) Time cost of **Query**

Fig. 2. The execution time of the algorithm in the system

– **Param:** Fig. 2(a) shows that the time cost of **Param** is a constant size. Moreover, the operations are consisted of bilinear group generation and hash function setting.

- **$\mathbf{KeyGen_S}$:** Fig. 2(b) shows that the time cost of $\mathbf{KeyGen_S}$ is a constant size. For instance, 9.22 ms is used on computer. Moreover, the operation only contains once exponentiation on group $\mathbb{G}_1$.
- **Adjust:** Fig. 2(c) shows that the time cost of **Adjust** is linear in the number of authorized search documents for one submitted keyword search query. For instance, when S contains 1000 authorized documents, 359640 ms is used on computer.
- **Match:** Fig. 2(d) shows that the time cost of **Match** is linear in the number of authorized search documents for one submitted keyword search query. For instance, when S contains 1000 authorized documents, 77360 ms is used on computer.
- **$\mathbf{KeyGen_{DO}}$:** Fig. 2(e) shows that the time cost of $\mathbf{KeyGen_{DO}}$ is linear in the maximum number of documents. For instance, when $n = 1000$, 28067 ms and 283620 ms are used on computer and smart phone, respectively. Therefore, the smart phone and computer meet the computation cost requirement in the system, because the $\mathbf{KeyGen_{DO}}$ runs in system idle excepting the first key generation phase.
- **Authorzie:** Fig. 2(f) shows that the time cost of **Authorzie** is linear in the number of authorized search documents for one user. For instance, when the set S contains 1000 authorized documents, 368.82 ms and 4186.02 ms are used on computer and smart phone, respectively.
- **Encrypt:** Fig. 2(g) shows that the time cost of **Encrypt** is linear in the number of keywords for each document. Moreover, for each keyword, the time cost of **Encrypt** is a constant size. For instance, 37.5 ms and 360.4 ms are used on computer and smart phone, respectively. Moreover, the operation only contains fourth exponentiations on group $\mathbb{G}_1$ for each encryption node. Therefore, **Encrypt** can be efficiently executed by resource constrained encryption nodes in IoT.
- **Query:** Fig. 2(h) shows that the time cost of **Trapdoor** is a constant size. For instance, 27.39 ms and 271.03 ms are used on computer and smart phone, respectively. Moreover, the operation only contains third exponentiations on group $\mathbb{G}_1$ for every query.

6 Conclusion

In this paper, we propose a fine-grained search privilege revocation construction to maintain forward secrecy for IoT data. Our scheme achieves the proposal of user search right revocation at document level. We implement a practical construction and our evaluation shows that our scheme is efficient.

Acknowledgement. This work is supported by the National Key Research and Development Program under Grant 2017YFB0802300, National Natural Science Foundation of China under grant No. 61502086 and 61572115; the Sichuan Provincial Major Frontier Issues (2016JY0007); the Guangxi Key Laboratory of Trusted Software (No. PF16116X); the foundation from Guangxi Colleges and Universities Key Laboratory of Cloud Computing and Complex Systems (No. YF16202).

References

1. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: 2000 IEEE Symposium on Security and Privacy, pp. 44–55. IEEE Computer Society Press, May 2000
2. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_30
3. PBC library. https://crypto.stanford.edu/pbc/
4. Yang, Y., Liu, X., Deng, R.H., Li, Y.: Lightweight sharable and traceable secure mobile health system. IEEE Trans. Dependable Secur. Comput. **99**, 1–1 (2017)
5. Popa, R.A., Zeldovich, N.: Multi-key searchable encryption, Cryptology ePrint Archive, Report 2013/508 (2013)
6. Popa, R.A., Stark, E., Valdez, S., Helfer, J., Zeldovich, N., Balakrishnan, H.: Building web applications on top of encrypted data using Mylar. In: Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2014, pp. 157-172 (2014)
7. Tang, Q.: Nothing is for free: security in searching shared and encrypted data. IEEE Trans. Inf. Forensics Secur. **9**(11), 1943–1952 (2014)
8. Liu, Z., Li, J., Chen, X., Yang, J., Jia, C.: TMDS: thin-model data sharing scheme supporting keyword search in cloud storage. In: Susilo, W., Mu, Y. (eds.) ACISP 2014. LNCS, vol. 8544, pp. 115–130. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08344-5_8
9. Van Rompay, C., Molva, R., Önen, M.: Multi-user searchable encryption in the cloud. In: Lopez, J., Mitchell, C.J. (eds.) ISC 2015. LNCS, vol. 9290, pp. 299–316. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23318-5_17
10. Chu, C.-K., Chow, S.S.M., Tzeng, W.-G., Zhou, J., Deng, R.H.: Key-aggregate cryptosystem for scalable data sharing in cloud storage. IEEE Trans. Parallel Distrib. Syst. **25**(2), 468–477 (2014)
11. Cui, B., Liu, Z., Wang, L.: Key-aggregate searchable encryption for group data sharing via cloud storage. IEEE Trans. Comput. **65**(8), 2374–2385 (2016)
12. Kiayias, A., Oksuz, O., Russell, A., Tang, Q., Wang, B.: Efficient encrypted keyword search for multi-user data sharing. In: Askoxylakis, I., Ioannidis, S., Katsikas, S., Meadows, C. (eds.) ESORICS 2016. LNCS, vol. 9878, pp. 173–195. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45744-4_9
13. Zhou, R., Zhang, X., Du, X., Wang, X., Yang, G., Mohsen, G.: File-centric multi-key aggregate keyword searchable encryption for industrial internet of things. IEEE Trans. Ind. Inform. **14**(8), 3648–3658 (2018)
14. Li, T., Liu, Z., Li, P., Jia, C., Jiang, Z.L., Li, J.: Verifiable searchable encryption with aggregate keys for data sharing in outsourcing storage. In: Liu, J.K., Steinfeld, R. (eds.) ACISP 2016. LNCS, vol. 9723, pp. 153–169. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40367-0_10
15. Liu, Z., Li, T., Li, P., Jia, C., Li, J.: Verifiable searchable encryption with aggregate keys for data sharing system. Future Gener. Comput. Syst. **78**, 778–788 (2018)
16. Shi, J., Lai, J., Li, Y., Deng, R.H., Weng, J.: Authorized keyword search on encrypted data. In: Kutyłowski, M., Vaidya, J. (eds.) ESORICS 2014. LNCS, vol. 8712, pp. 419–435. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11203-9_24

17. Sun, W., Yu, S., Lou, W., Hou, Y.T.: Protecting your right: verifiable attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud. IEEE Trans. Parallel Distrib. Syst. **27**(4), 1187–1198 (2016)
18. Zheng, Q., Shouhuai, X., Ateniese, G.: VABKS: verifiable attribute-based keyword search over outsourced encrypted data. In: 2014 Proceedings IEEE, INFOCOM, pp. 522–530 (2014)
19. Bao, F., Deng, R.H., Ding, X., Yang, Y.: Private query on encrypted data in multi-user settings. In: Chen, L., Mu, Y., Susilo, W. (eds.) ISPEC 2008. LNCS, vol. 4991, pp. 71–85. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-79104-1_6
20. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45682-1_30
21. Dong, C., Russello, G., Dulay, N.: Shared and searchable encrypted data for untrusted servers. In: Atluri, V. (ed.) DBSec 2008. LNCS, vol. 5094, pp. 127–143. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70567-3_10
22. Dong, C., Russello, G., Dulay, N.: Shared and searchable encrypted data for untrusted servers. J. Comput. Secur. **19**(3), 367–397 (2011)
23. Wang, X., Mu, Y., Chen, R., Zhang, X.: Secure channel free ID-based searchable encryption for peer-to-peer group. J. Comput. Sci. Technol. **31**(5), 1012–1027 (2016)