4-2019

Fair and dynamic data sharing framework in cloud-assisted Internet of Everything

Yinbin MIAO
Xidian University

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

Kim-Kwang Raymond CHOO
University of Texas at San Antonio

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

Hongjun WU
Nanyang Technological University

See next page for additional authors

Citation

MIAO, Yinbin; LIU, Ximeng; CHOO, Kim-Kwang Raymond; DENG, Robert H.; WU, Hongjun; and LI, Hongwei. Fair and dynamic data sharing framework in cloud-assisted Internet of Everything. (2019). *IEEE Internet of Things*. 6, (4), 7201-7212.
Available at: https://ink.library.smu.edu.sg/sis_research/4676

Fair and Dynamic Data Sharing Framework in Cloud-Assisted Internet of Everything

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

***Abstract*—Cloud-assisted Internet of Things (IoT) is increasingly prevalent in our society, for example in home and office environment; hence, it is also known as Cloud-assisted Internet of Everything (IoE). While in such a setup, data can be easily shared and disseminated (e.g., between a device such as Amazon Echo and the cloud such as Amazon AWS), there are potential security considerations that need to be addressed. Thus, a number of security solutions have been proposed. For example, Searchable Encryption (SE) has been extensively studied due to its capability to facilitate searching of encrypted data. However, threat models in most existing SE solutions rarely consider the malicious data owner and semi-trusted cloud server at the same time, particularly in dynamic applications. In a real-world deployment, disputes between above two parties may arise as either party will accuse the other of some misbehavior. Furthermore, efficient full-update operations (e.g., data modification, data insertion, data deletion) are not typically supported in the cloud-assisted IoE deployment. Therefore, in this paper, we present a Fair and Dynamic Data Sharing Framework (FairDynDSF) in the multi-owner setting. Using FairDynDSF, one can check the correctness of search results, achieve fair arbitration, multi-keyword search, and dynamic update. We also prove that FairDynDSF is secure against inside keyword guessing attack and demonstrate its efficiency by evaluating its performance using various datasets.**

***Index Terms*—Internet of Things, Internet of Everything, Searchable encryption, Full-update operation, Multi-owner setting, Fair arbitration.**

I. Introduction

INTERNET of Things (IoT) is fast, and can be found in a broad range of applications ranging from consumers (e.g., smart homes) to organizations (e.g., Industry 4.0) to governments (e.g., Internet of Battlefield/Military Things) [1], [2], [3], [4] – collectively, this can be referred to as Internet of Everything (IoE)[1]. Due to the limitations of IoT devices, such as computation and storage capacities, there is a shift towards cloud-assisted IoE/IoT where computationally intensive tasks/activities are being offloaded to the cloud [5].

Although outsourcing data to the cloud can significantly reduce local computation and storage burden, this will pose potential security risks (i.e., data privacy leakage, data integrity violation and illegal access in the complex ecosystem). The cloud provide by third-parties may arbitrarily access the sensitive data (e.g., financial documents, personal health records) or deliberately discard rarely accessed data. Encryption is typically used to ensure data confidentiality and privacy, at the expense of utility (e.g., searching on ciphertexts becomes costly and challenging). Thus, there has been interest in exploring the potential of searchable encryption (SE) technique [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], due to its capability to allow authorized (cloud) users to retrieve data of interest based on specified keywords.

To the best of our knowledge, the existing SE solutions generally assume that the cloud server is honest-but-curious, which faithfully executes search operations based on the agreed protocols but may be curious to learn potentially sensitive information [16]. However, in a real-world situation the cloud server is more likely to be semi-trusted in the sense that it may delete rarely or never accessed data in order to cut costs (e.g., reducing storage space and computation overhead), conduct partial search operations honestly and output a fraction of inaccurate search results, and so on [17]. Therefore, to determine the correctness of returned results (the *first challenging issue* that we intend to deal with), some SE schemes are designed to provide result verification [18], [19], [20]. However, such schemes deal with static data rather than dynamic data (e.g., insertion operation, modification operation, deletion operation), thereby limiting their usefulness and scalability in actual deployments, which is the *second challenging issue* that we need to solve.

Although a small number of dynamic and verifiable SE schemes have been proposed [21], [22], these schemes incur high result verification costs on the resource-constrained IoT devices. It is because security-oriented processes, such as data encryption, ciphertext decryption, and trapdoor generation, are generally computation-intensive operations and mainly executed by IoT devices. Thus, practical SE scheme should provide a lightweight data sharing mechanism deployed on (resource constrained) IoT devices.

To eliminate the high computation and storage burden on resource-constrained devices (e.g., smartphones, sensors, and wearable devices), most of verifiable SE solutions always

Y. Miao is with the School of Cyber Engineering, Xidian University, Xi'an 710071, China; State Key Laboratory of Cryptology, P.O.Box 5159, Beijing 100878, China. E-mail: ybmiao@xidian.edu.cn

X. Liu and R. H. Deng are with the Department of Information Systems, Singapore Management University, 80 Stamford Road, Singapore. Email: snbnix@gmail.com, robertdeng@smu.edu.sg

K.-K. R. Choo is with the Department of Information Systems and Cyber Security, The University of Texas at San Antonio, San Antonio, TX 78249, USA. Email: raymond.choo@fulbrightmail.org

H. Wu is with the School of Physical and Mathematical Sciences, Nanyang Technological University, 21 Nanyang Link 637371, Singapore. Email: wuhj@ntu.edu.sg

H. 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

[1]IoE is a term coined by Cisco (see https://newsroom.cisco.com/ioe and https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoE_Economy_FAQ.pdf, last accessed December 20, 2018)

delegates his/her result verification tasks to a certain public or private auditor (one that is trusted by the data owner but not necessarily by the cloud server) to avoid incurring costly verification overhead. Existing SE schemes generally assume that the data owner is honest in the security model. However, both data owner and cloud server may be motivated to behave dishonestly to obtain some compensation from the other party, which is the *third challenging issue* that we want to tackle. For instance, a greedy data owner may allege an honest cloud server to benefit financially, for example by uploading invalid records or corresponding tags and refusing to acknowledge the authenticity of verification reports [23], [24], [25]. Besides, a semi-trusted cloud server would typically execute honestly as the prescribed protocol, but may attempt to pass the result verification mechanism without offering true search results. In other words, disputes between these two parties are generally unavoidable. Thus, the need for fairness guarantee in verifiable SE schemes is urgent.

Furthermore, most of existing verifiable SE schemes are not designed to facilitate verifiable keyword search in the multi-owner setting, which is a common application in practice and the *fourth challenging issue* that we try to settle. For example, each electronic medical record (EMR) belonging to a certain patient needs to be confirmed and signed by one or more medical practitioners and/or allied healthcare professionals before being outsourced to the cloud server. Although approaches in [26], [27] consider such a setting, they do not consider decryption authorization. Precisely, the data user can decrypt retrieved search results on the condition that (s)he must be authorized by at least a threshold number of data owners. Therefore, not all data owners are required to be online at the same time.

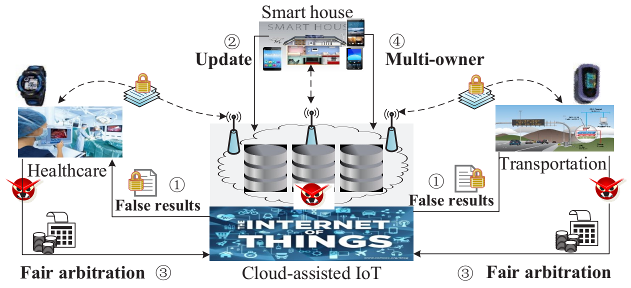


Fig. 1: Challenging issues that we attempt to address in FairDynDSF.

Up to now, the existing SE schemes can just address the discussed problems separately. To mitigate these four challenging issues (namely: ①-avoiding returning false results; ②-supporting dynamic update; ③-supporting fair arbitration; ④-supporting multi-owner setting with threshold decryption privilege) at the same time in Fig. 1, we present a Fair and Dynamic Data Sharing Framework (FairDynDSF), which aims at achieving dynamic fairness guarantee, threshold decryption privilege in the multi-owner setting and high efficiency without more costly hash operations (e.g., maps each arbitrary string into group element) in a cloud-assisted IoE environment. FairDynDSF is designed to also withstand inside keyword-guessing attacks (KGAs) while achieving secure and fair arbitration, which is shown as follows:

- *Verifiable multi-keyword search.* FairDynDSF has a well-designed result verification protocol to prevent a semi-trusted cloud server from offering inaccurate search results, and avoid results of little or no relevance being returned.
- *Multi-owner setting.* Unlike a common single-owner setting, the multi-owner setting supported by FairDynDSF allows each record to be co-accredited by multiple data owners. To be specific, multiple data owners collaboratively delegate the record signature tasks to a trusted entity, and at least a threshold number of data owners can grant the record decryption privileges.
- *Dynamic update.* FairDynDSF supports record dynamic update (e.g., modification, deletion, and insertion) without incurring significant computation overhead for ciphertext update. Specifically, FairDynDSF updates the index switcher and generates a small portion of EMR signatures and ciphertexts at a small cost (i.e., computation and communication overhead).
- *Fair guarantee.* Due to the fair guarantee feature in FairDynDSF, dishonest entities are discouraged from misbehaving (e.g., the untrue accusation from data owner, returning false search results or refusing update operations by semi-trusted cloud server).

II. Related Work

SE schemes can be categorized into symmetric and asymmetric [6], [16], and the focus of this paper is the latter as the symmetric SE schemes generally incur expensive key management and communication costs. The first asymmetric SE scheme is proposed by Boneh *et al.* [7] (also known as the Public key Encryption with Keyword Search – PEKS), which allows the data user to deliver a trapdoor and the cloud server to test this trapdoor with stored indexes.

As previously discussed, a cloud server may be selfishly motivated to minimize the operating costs, for example by deleting infrequently accessed records and producing partial search results. Thus, this results in the design of verifiable SE schemes [28], [29], [30]. For example, Liu *et al.* [28] presented a verifiable SE with aggregated keys scheme, which allows an authorized data user to create a single trapdoor and search different record sets encrypted by different keys. However, result verification incurs significant computation costs on resource-limited data users (e.g., wearable devices). To improve efficiency, Chen *et al.* [29] presented a verifiable fine-grained keyword search scheme for mobile healthcare networks. The scheme allows one to check the completeness and accuracy of search results by combining the invertible Bloom lookup table and the Merkle hash tree. However, the approach results in a high false-positive rate. Therefore, Miao *et al.* [27] designed a practical SE scheme in multi-owner settings [31], which utilizes the private audit server to guarantee the validity of search results. Unfortunately, the solutions discussed so far only achieve private verification rather than public verification.

Unlike private verification, public verification does not require a data owner to be online or incur a significant computation load. Hence, public verification has gained popularity in recent years, for example being utilized in public auditing technique for cloud storage. To further discourage a semi-trusted cloud server from discarding rarely or never accessed data, public auditing enables an arbitrary public verifier to check the integrity of cloud data on behalf of the data owner without violating data confidentiality. However, the threat models in these schemes [32], [33], [34] never consider a dishonest data owner, who may be financially motivated to cheat the cloud server. Hence, there is a need to design a fair arbitration mechanism in public auditing.

Wang *et al.* [35] devised a fair remote retrieval model, using homomorphic privately verifiable tags and fair multi-member key exchange. The model reportedly achieves high efficiency in terms of computation and communication. However, the model does not provide dispute resolution. Extending existing threat models to support fair arbitration, Küpçü *et al.* [24] presented a general-purpose arbitration solution for both static and dynamic settings. Specifically, the solution uses the fair signature exchange protocol [36], [37]. On the basis of [24], the approach of Jin *et al.* [23] provided dynamics support, public verifiability and dispute arbitration concurrently. However, these fair data sharing schemes do not support either keyword-based ciphertext retrieval or multi-owner settings.

Therefore, we seek to address the above discussed limitations (i.e., verifiable keyword search, multi-owner setting, dynamic data update, and fair arbitration) – see TABLE I.

TABLE I: A comparative summary of features in existing SE schemes

Schemes	$\mathbb{F}_1$	$\mathbb{F}_2$	$\mathbb{F}_3$	$\mathbb{F}_4$	$\mathbb{F}_5$	$\mathbb{F}_6$
[18]	Single	Private	×	×	Static	Resist CKA
[21]	Multiple	Public/Private	×	×	Dynamic	UC-secure
[23]	×	Public	×	✓	Dynamic	Sound
[24]	×	Public	×	✓	Static/Dynamic	—
[25]	Single	Private	×	✓	Static	Resist CMA
[26]	Multiple	Private	✓	×	Static	Resist CKA
[27]	Single	Private	✓	×	Static	Resist CKA
[28]	Single	Private	✓	×	Static	Controllable
[29]	Single	Public	×	✓	Static	Resist CKA
[35]	×	Public	×	✓	Static	—
FairDynDSF	Multiple	Public	✓	✓	Dynamic	Resist KGAs

Notes. $\mathbb{F}_1$: Keyword search; $\mathbb{F}_2$: Results verification; $\mathbb{F}_3$: Multi-owner setting; $\mathbb{F}_4$: Fair arbitration; $\mathbb{F}_5$: Dataset; $\mathbb{F}_6$: Security; "CKA": Chosen-Keyword Attack; "UC-secure": Universally Composable security; "CMA": Chosen-Message Attack; "KGAs": Keyword-Guessing Attacks.

III. System, Threat and Security Models

In this section, we respectively describe the system model (Section III-A), threat model (Section III-B), and security model (Section III-C).

A. System Model

The FairDynDSF system considers a cloud-assisted IoE scenario, for example in a hospital setting. In the latter, a patient's EMR is collected from multiple medical and healthcare professionals (e.g., from different hospitals in Texas) via their IoT devices and then uploaded to a public cloud server (e.g., the Texas healthcare cloud). Such a model generally comprises five entities[2], namely: Data Owners (DOs), Data Users (DUs), Public Cloud Server (PCS), Public Auditor Server (PAS), and Fair Arbitration Server (FAS) – see Fig. 2. The entities are introduced below:

- *Data owners*: Multiple DOs (e.g., medical and healthcare professionals from different departments and/or hospitals) collect diagnostic information from a patient, which is then recorded in the patient's EMR. Then, the specified DO Manager (DOM), on behalf of multiple DOs, builds indexes (based on keywords) and the index switcher, generates a signature on the EMR, and encrypts each EMR before outsourcing the final ciphertexts to HCS – see step ①. DOM can execute EMR update (e.g., modification, deletion, and insertion) by using the index switcher.
- *Public cloud server*: PCS has significant computation resources and storage space, stores and processes EMR from DOs. We assume that PCS may occasionally or deliberately delete some rarely or never accessed EMRs to save storage space, or return partial results to cheat PAS.
- *Public auditor server*: PAS periodically checks the accuracy of returned results in a challenge-response manner (see steps ③ and ④), and notifies DOM of the verification reports honestly. In addition, the correct search results are passed to the queried DU (see step ⑤).
- *Fair arbitration server*: When a dispute occurs between DOs and PCS (as shown by the dotted red line), FAS provides fair arbitration based on result verification and EMR update (as shown by the red line).
- *Data users*: A legitimate DU can search the encrypted EMRs with the valid trapdoor (see step ②). However, DU can decrypt the exact search results on the premise of gaining decryption privileges granted by at least the default threshold number of DOs (see step ⑥).

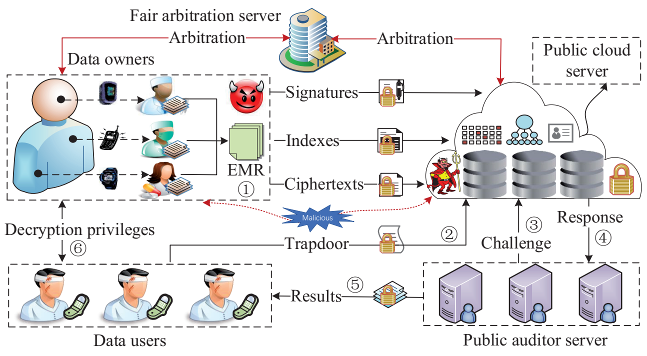


Fig. 2: System model of FairDynDSF.

[2]Note that the trusted authority tasked with the generation of global public parameters and public/secret key pairs for various entities is omitted in this system model.

B. Threat Model

The threat models in most verifiable SE schemes are not capable of handling disputes between DOs and PCS, as DOs are usually assumed to be fully trusted. However, in FairDynDSF, both DOs and PCS may falsely accuse each other. Although DOs delegate result verification tasks to PAS and trust that PAS will return the correct verification reports, a semi-trusted PCS does not trust the PAS completely. It is because DOs and PAS may collude with each other to frame the PCS for financial benefit.

Therefore, to mitigate limitations in such a simplistic threat model, FairDynDSF is designed on the assumption that an honest-but-curious FAS exists, which is a tradeoff between security and efficiency (Note that fully-trusted FAS will lead to high communication costs caused by building security channels). In other words, the FAS implements the established agreement honestly but may be curious to learn the underlying information (i.e., the indices of updated EMRs, the identity of DOM, etc.). For security concerns, FairDynDSF can utilize the random mask and signature techniques to guarantee the privacy of indices of updated EMRs and identity of DOM, respectively, which is beyond the scope of our discussion. In addition, we require FAS to be an impartial third-party and should provide fair arbitration. Authorized DUs will still need to be accredited by multiple DOs before they can recover the plaintext EMRs.

C. Security Model

Similar to previous schemes such as those of [38], [39], the security of FairDynDSF requires that the index and trapdoor indistinguishability to be guaranteed, which is based on the following two games. Here, we briefly describe these two games.

- In game 1, the adversary $\mathcal{A}$ may be the semi-trusted PCS or a malicious DU. Hence, $\mathcal{A}$ can obtain the PCS's secret key or the DU's secret key, but it cannot issue the trapdoor queries corresponding to the challenging keywords w_0, w_1 of his/her choice. FairDynDSF can ensure the index indistinguishability if there does not exist an adversary that can tell the indexes of w_0 and w_1 without obtaining the valid trapdoor.
- In game 2, $\mathcal{A}$ may be the semi-trusted PCS or a malicious DOM. $\mathcal{A}$ can obtain the PCS's secret key or the DOM's secret key. The trapdoor indistinguishability of FairDynDSF requires that $\mathcal{A}$ should not be able to distinguish the trapdoors of keyword(s) w_0 (or W_0) and w_1 (or W_1).

Definition 1. FairDynDSF is secure against the inside KGAs if there does not exist any adversary that can comprise the index and trapdoor indistinguishability with a non-negligible advantage in both games 1 and 2 described above.

Besides, we formally define other security property [40] required in FairDynDSF. This security game is conducted between the challenger $\mathcal{C}$ and $\mathcal{A}$. Note that the semi-trusted PCS acts the role of $\mathcal{A}$. The introduction of this security game is shown in **Supplemental Material A**.

Definition 2. FairDynDSF achieves signature unforgeability against a semi-trusted PCS if there does not exist an adversary that can break the above security game with a non-negligible advantage.

IV. PROPOSED FAIRDYNDSF FOR CLOUD-ASSISTED IOE

Although the scheme in [23] achieves public verification, fair arbitration, and dynamic update, the particular scheme does not achieve data sharing within a cloud-assisted IoE environment. To allow IoT devices to share data without sacrificing security, we present the construction of FairDynDSF (a dynamic and verifiable multi-keyword search with dispute arbitration scheme in the multi-owner setting). Before describing FairDynDSF, we will first introduce relevant background materials.

Let $\mathbb{Z}_p$ be the field with elements modulo p, $\mathbb{G}, \mathbb{G}_T$ be two cyclic multiplicative groups of order p. The bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ has three key properties: (a) *Bilinearity*: for $\forall h_1, h_2 \in \mathbb{G}$, $a^*, b^* \in \mathbb{Z}_p$, $e(h_1^{a^*}, h_2^{b^*}) = e(h_1, h_2)^{a^* b^*}$; (b) *Computability*: for $\forall h_1, h_2 \in \mathbb{G}$, there exists an efficient algorithm to output $e(h_1, h_2)$; (c) *Non-degeneracy*: $\exists h_1, h_2 \in \mathbb{G}$, $e(h_1, h_2) \neq 1$. Let H, h denote two collision-resistant map-to-point hash functions $H : \{0,1\}^* \rightarrow \mathbb{G}$, $h : \{0,1\}^* \rightarrow \mathbb{Z}_p$, and $[\hat{a}, \hat{b}]$ be a set of integers including $\hat{a}, \hat{b}$.

FairDynDSF comprises key generation **KeyGen**, ciphertexts uploading **Enc**, EMRs update **Update**, trapdoor generation **Trap**, ciphertexts retrieval **Search**, result verification **Verify**, fair arbitration **Arbitrate**, and decryption authorization **Dec** – see Sections IV-A to IV-H. A summary of notations used in the framework is listed in TABLE II.

TABLE II: Summary of notations

Notation	Description	Notation	Description
(pk_o, sk_o)	DOM's Keys	$\mathcal{W} = \{w_1, \cdots, w_n\}$	Keyword set
(pk_s, sk_s)	PCS's Keys	$\mathcal{D} = (m_1, \cdots, m_f)$	EMR set
(pk_u, sk_u)	DU's Keys	$I = (I_0, I_1, I_2, \{I_j\})$	Indexes for $\mathcal{W}$
$C = \{c_i\}$	EMRs ciphertexts	$S = \{s_i\}$	EMRs signatures
S_o	DOM's signature	$\mathcal{O} = \{O_1, \cdots, O_d\}$	DO group
S_s	PCS's signature	$\theta = \{(t_i, i)\}$	Index switcher
ϕ	Update request	$W' = \{w'_1, \cdots, w'_l\}$	Queried keywords
$\{z, \mu_z\}$	Challenge info	$T_{W'} = (T_{W',1}, T_{W',2})$	Trapdoor
(μ, s)	Proof info	$C^* = \{c'_1, \cdots, c'_q\}$	Search results

A. Key generation

Given global public parameters $pp = (g, g_0, u, H, h)$, where g is the generator of $\mathbb{G}$, $g_0, u \in \mathbb{G}$, FairDynDSF calls **KeyGen** to produce public/secret key pairs for DOM, DU, PCS, respectively. Note that the DO group is defined as $\mathcal{O} = \{O_1, \cdots, O_d\}$.

- **KeyGen**$_o$: Choose $a \in \mathbb{Z}_p$ and compute g^a, and the public and secret keys of DOM are defined as $pk_o = g_0^a$, $sk_o = a$, respectively.
- **KeyGen**$_u$: Select $b \in \mathbb{Z}_p$ and compute $e(g, g_0)^{1/b}$, g^b, and the public and secret keys of DU are defined as $pk_u = (pk_{u,1}, pk_{u,2})$, $sk_u = b$, where $pk_{u,1} = e(g, g_0)^{1/b}$, $pk_{u,2} = g^b$, respectively.
- **KeyGen**$_s$: Pick $c \in \mathbb{Z}_p$ and compute g^c, and the public and secret keys of PCS are defined as $pk_s = g^c$, $sk_s = c$, respectively.

B. Ciphertext uploading

Upon receiving the EMR, DOM calls **Enc** to output the keyword indexes, EMR signature, EMR ciphertext. Then, DOM uploads the final ciphertexts $CT = (I, S, C, S_o)$ to PCS, where $I = (I_0, I_1, I_2, \{I_j\})$, $C = \{c_i\}$, $S = \{s_i\}$.

- As for the keyword index I, DOM first selects $v \in \mathbb{Z}_p$ and computes $I_0 = e(g, g)^v$, $I_1 = pk_{u,2}^v = g^{bv}$, $I_2 = e(g, g_0)^{av/b}$, then calculates $I_j = g^{-vh(w_j)}$ for each keyword $w_j (j \in [1, n])$ in $\mathcal{W} = \{w_1, \cdots, w_n\}$.
- DOM encrypts each EMR $m_i (i \in [1, f])$ in $\mathcal{D} = (m_1, \cdots, m_f)$ by using a symmetric encryption algorithm Enc (e.g., AES, DES), namely: $c_i = Enc_a(m_i)$. Then, DOM randomly chooses a polynomial $f(x)$ of degree $t-1$, $f(x) = a_0 + a_1 x + \cdots + a_{t-1} x^{t-1}$, where $a_0 = a$, $\{a_0, \cdots, a_{t-1}\} \in \mathbb{Z}_p^t$, $2t - 1 \geq d$. Finally, DOM selects d points $\{(x_r, y_r)\}(r \in [1, d])$ from $f(x)$ and computes $q_r = g^{y_r}$, where $y_r = f(x_r) \in \mathbb{Z}_p$. Note that y_r is sent to each DO (O_r) via a secure channel (e.g., Secure Sockets Layer – SSL), q_r is public.
- DOM outputs the signature for each EMR m_i as $s_i = (H(t_i) u^{h(c_i)})^a$, and generates the signature $S_o = H(seq||\theta)^{sk_o}$ on the index switcher $\theta = \{(t_i, i)\}(i \in [1, f])$, where t_i represents the index of signature s_i, and i denotes the index of EMR m_i. Initially, both EMR indices and signature indices have the same sequence, $t_i = i$. seq denotes the update count pointer and its initial value is 0, which is incremented by one when DO executes each update operation.

To avoid a dishonest DOM (note that DOM does not tamper or forge EMRs) from outputting false signature s_i for each EMR m_i so that (s)he can maliciously frame an honest PCS, PCS first checks the validity of (s_i, c_i) using $e(s_i, g_0) \overset{?}{=} e(H(t_i) u^{h(c_i)}, pk_o)$ at the initial stage. If the test succeeds, then PCS is ensured that DOM has generated the accurate signature. Then, PCS generates its signature $S_s = H(seq||\theta)^{sk_s}$ on the index switcher θ. PCS also needs to check the correctness of S_o using the public key pk_o, namely $e(S_o, g_0) \overset{?}{=} e(H(seq||\theta), pk_o)$. If this verification passes, then PCS keeps S_o; otherwise, PCS turns to FAS for fair arbitration.

C. EMR update

To better understand EMR updating in the process of **Update**, we first show how to deal with modification, insertion and deletion operations in θ according to Fig. 3, which is similar to the scheme in [23]. However, the difference is that FairDynDSF should encrypt the modified or inserted EMRs so that PCS cannot access sensitive information.

- *Modification*. Assume that DOM modifies $m_k (k \in [1, f])$ into m'_k, (s)he assigns an unused signature index t'_k for m_k and calculates its new cihphertext $c'_k = Enc_a(m'_k)$ and signature $s'_k = (H(t'_k) u^{h(c'_k)})^a$. Then, (s)he updates the index switcher θ as θ'. Finally, (s)he returns the update request $\phi = \{seq', O(M), k, t'_k, c'_k, s'_k, \theta', S'_o\}$ to PCS, where $O(M)$ denotes the modification operation, $S'_o = H(seq'||\theta')^{sk_o}$, $seq' = seq + 1$. Note that seq represents the value that has been successfully updated previously.
- *Insertion*. Similar to the modification operation, DOM also allocates an unused signature index t'_k when adding a new EMR at the k-th position, as well as new ciphertext c'_k and signature s'_k. Additionally, there is an update request $\phi = \{seq', O(I), k, t'_k, c'_k, s'_k, \theta', S'_o\}$, where $O(I)$ means the insertion operation.
- *Deletion*. When DOM intends to delete EMR m_k, (s)he just needs to output the updated index switcher θ' and submits the update request $\phi = \{seq', O(D), k, \theta', S'_o\}$ to PCS, where $O(D)$ denotes the deletion operation.

After obtaining ϕ in terms of modification or insertion operation, PCS first needs to check the authenticity of c'_k, s'_k by verifying $e(s'_k, g_0) \overset{?}{=} e(H(t'_k) u^{h(c'_k)}, pk_o)$. If this equation holds, then PCS proceeds to verify the accuracy of S'_o by using his/her own updated index switcher θ' according to the update request ϕ, namely $e(S'_o, g_0) \overset{?}{=} e(H(seq'||\theta'), pk_o)$. If both verifications succeed, then PCS will replace the tuple (c_k, s_k) with a new one (c'_k, s'_k), or insert the new pair (c'_k, s'_k) into the stored ciphertexts. Note that PCS only executes the signature verification for deletion operation. If DOM has generated the signatures for each updated EMR and index switcher honestly, then PCS also returns his/her signature $S'_s = H(seq'||\theta')^{sk_s}$ on his/her own updated index switcher θ' to DOM for any subsequent dispute arbitration.

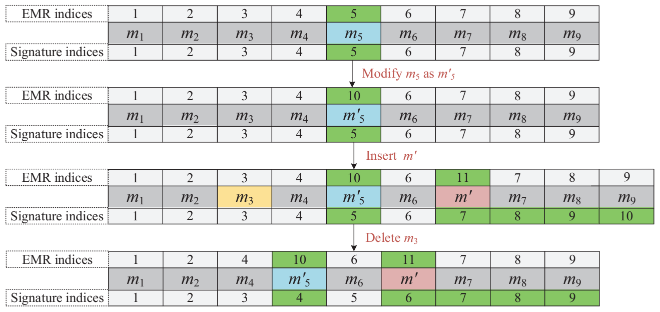


Fig. 3: The change of EMR and signature indices in **Update**.

D. Trapdoor generation

When a particular DU conducts search query $W' = \{w'_1, \cdots, w'_l\}$, (s)he calls **Trap** to generate trapdoor $T_{W'} = (T_{W',1}, T_{W',2})$. First, DU chooses $T_{W',1} = \varphi \in \mathbb{Z}_p$, then computes $T_{W',2} = (pk_o^{1/b} pk_s^{-\varphi})^{1/(b - \sum_{\tau=1}^{l} h(w'_\tau))}$, where $\tau \in [1, l]$. Finally, DU sends $T_{W'}$ along with keyword locations L of queried keywords in keyword dictionary $\mathcal{W}$ to PCS. Here, we define a function $\rho_1(\cdot)$ which maps the keyword index in W' to that in $\mathcal{W}$, namely $w'_\tau = w_{\rho_1(\tau)}$. Note that the location privacy of queried keywords can be protected using random mask techniques (e.g., pseudo-random functions) [41].

E. Ciphertexts retrieval

In this process, PCS calls **Search** to check whether the submitted trapdoor $T_{W'}$ matches with indexes T using Eq. 1. If Eq. 1 holds, then PCS returns the corresponding search results $C^* = \{c'_1, \cdots, c'_q\}$ to PAS; otherwise, PCS aborts this process.

$$e(I_1 \cdot \prod_{\tau=1}^{l} I_{\rho_1(\tau)}, T_{W',2}) I_0^{cT_{W',1}} \overset{?}{=} I_2. \tag{1}$$

F. Result verification

After obtaining the search results C^*, PAS has to test the correctness of C^* by calling **Verify**. First, PAS selects an element $\mu_z \in \mathbb{Z}_p$ for each result $c'_z(z \in [1,q])$, and sends the challenging information $\{z, \mu_z\}$ to the PAS. To pass the result verification, PAS computes $\mu = \sum_{z=1}^{q} \mu_z h(c'_z)$, $s = \prod_{z=1}^{q} (s'_z)^{\mu_z}$, and then sends proof information (μ, s) to PAS, where s'_z represents the signature for c'_z. For ease of description, we introduce another function $\rho_2(\cdot)$ which maps each index of search results to that of EMR ciphertext, namely $c'_z = c_{\rho_2(z)}$, $s'_z = s_{\rho_2(z)}$. Upon getting the proof information (μ, s), PAS checks its validity with Eq. 2. If Eq. 2 holds, then PAS sends C^* to DU; otherwise, PAS rejects.

$$e(s, g_0) \overset{?}{=} e(\prod_{z=1}^{q} H(t_{\rho_2(z)})^{\mu_z} \cdot u^{\mu}, pk_o). \tag{2}$$

PAS also needs to guarantee that PCS has performed the update request honestly. First, PAS selects a challenging set $\omega = \{z', \mu'_{z'}\}_{z' \in \Delta \cap k}$ and sends ω to PCS, where $\Delta = [1, q]$. That is to say, the challenging set includes the updated EMR. PCS returns proof information (μ', s') to PAS, which verifies the correctness of (μ', s') using $e(s', g_0) \overset{?}{=} e(\prod_{z' \in \Delta \cap k} H(t_{\rho_2(z')})^{\mu'_{z'}} \cdot u^{\mu'}, pk_o)$. PAS can continue to verify the validity of S'_s on behalf of DOM. If S'_s is valid, then it implies that PCS has indeed performed the update operations.

G. Fair arbitration

As mentioned above, both DOM and PCS may accuse each other. In FairDynDSF, PAS should rely on index switcher θ to obtain the signature indices so that (s)he can issue the result verification. The creation and update of θ are only completed by DOM, which can be potentially exploited by DOM to frame PCS. Thus, supporting fair arbitration for potential disputes is indispensable for FairDynDSF. Next, we show how FairDynDSF resolves the conflict by calling **Arbitrate**.

The straightforward solution is to send the update information (e.g., operation position, operation type, and unused signature index) to FAS for each update executed by DOM. However, this method inevitably imposes additional communication and storage costs. Thus, FairDynDSF uses the signature exchange mechanism [23], [24], [37] to guarantee the correctness of index switcher, which requires both entities to exchange their respective signatures on the latest index switcher after each dynamic operation. Note that PCS can recover the latest index switcher by employing the essential update information in ϕ. In the initial phase, the signature indices are the same as the EMR indices, namely $\{(t_i = i, i)\}$, and the signature exchange can be easily completed since the initial content of θ is known to both DOM and PCS. If the update operation has been performed successfully, not only the PAS succeeds in issuing result verification in a challenge-response mode, but also both DOM and PCS maintain the other side's signature. This means that both sides have researched an agreement on the current update. The conflict between these two parties may happen in any of the following three cases:

- PAS claims a failure of result verification in **Verify**.
- PCS gains an incorrect update request ϕ in **Update**.
- DOM obtains an invalid signature on the updated index switcher in **Arbitrate**.

As for the first dispute case, the involved FAS asks for the necessary information $\{seq_o, \theta_o, S_s\}$, $\{seq_s, \theta_s, S_o\}$ from DOM and PCS, respectively. Note that seq_o (or seq_s) denotes the value of update count pointer returned by DOM (or PCS), θ_o, θ_s represents the index switcher sent by both parties, respectively. Next, we show the arbitration results derived from FAS in Fig. 4. Note that $seq_o = seq_s$ in the first dispute case happens in the current update operation.

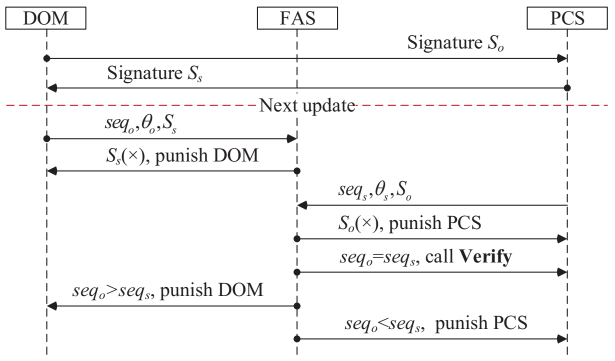

DOM
FAS
PCS
Signature S_o
Signature S_s
Next update
seq_o, θ_o, S_s
$S_s(\times)$, punish DOM
seq_s, θ_s, S_o
$S_o(\times)$, punish PCS
$seq_o = seq_s$, call **Verify**
$seq_o > seq_s$, punish DOM
$seq_o < seq_s$, punish PCS

Fig. 4: Arbitration on result verification.

To deal with the remaining dispute cases, FAS should help both parties to complete the current update and signature exchange. Besides, the signatures in the most recent successful operation should be checked again so that FAS can proceed on condition that the current update operation has not yet been completed. In these two cases, the result verification process is similar to that in the first case. That is to say, FAS decides that this party is cheating if (s)he receives an invalid signature from the corresponding party. As for the comparison between the values of update count pointer, the fair arbitration is divided into two situations (e.g., $seq_o = seq_s$, $seq_o \neq seq_s$), which is shown in Fig. 5. Note that $seq_o = seq_s$ in the last two dispute cases occurs in the last successful update operation, while the current update operation has not been accomplished. Besides, as for the case $seq_o = seq_s + 1$, there are three possibilities (i.e., (1) The DOM has inconsistent EMR and corresponding signature; (2) The PCS returns an invalid signature on updated index switcher; (3) PCS refuses to update, etc.) causing fair arbitration. With regard to the third possibility (also referred to as denial-of-update), FAS cannot decide which party should be punished, as each party may behave maliciously to the other party but honestly to FAS. For instance, the DOM sends a false updated EMR to PCS in the current update operation but returns a correct update EMR in the fair arbitration process. In this case, FAS treats $seq_o = seq_s$ so that it can accomplish the current update operation and signature exchange, which does not affect the following tasks of result verification and update operation.

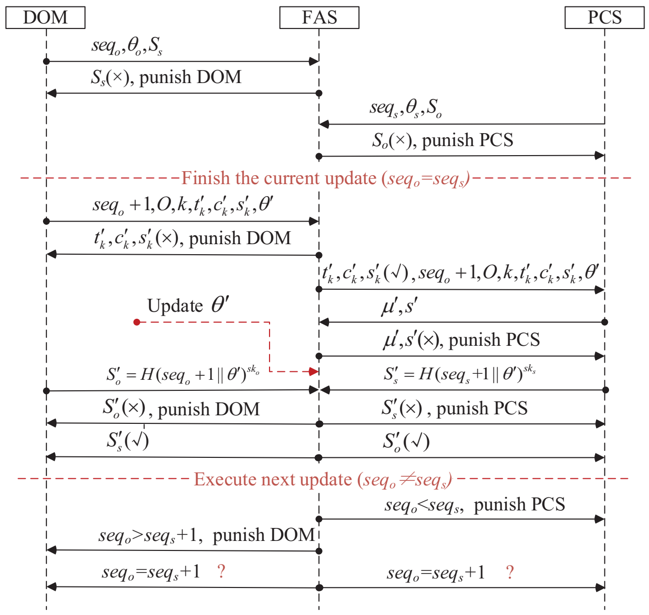

DOM
FAS
PCS
seq_o, θ_o, S_s
$S_s(\times)$, punish DOM
seq_s, θ_s, S_o
$S_o(\times)$, punish PCS
Finish the current update ($seq_o = seq_s$)
$seq_o+1, O, k, t'_k, c'_k, s'_k, \theta'$
$t'_k, c'_k, s'_k(\times)$, punish DOM
$t'_k, c'_k, s'_k(\surd), seq_o+1, O, k, t'_k, c'_k, s'_k, \theta'$
Update θ'
μ', s'
$\mu', s'(\times)$, punish PCS
$S'_o = H(seq_o+1 \| \theta')^{sk_o}$
$S'_s = H(seq_s+1 \| \theta')^{sk_s}$
$S'_o(\times)$, punish DOM
$S'_s(\times)$, punish PCS
$S'_s(\surd)$
$S'_o(\surd)$
Execute next update ($seq_o \neq seq_s$)
$seq_o < seq_s$, punish PCS
$seq_o > seq_s+1$, punish DOM
$seq_o = seq_s+1$?
$seq_o = seq_s+1$?

Fig. 5: Arbitration on the dynamic update.

H. Decryption authorization

After gaining the correct search results $C^* = \{c'_z\}$, DU still has to obtain at least t decryption authorizations from DO group $\mathcal{O}$, as the entire DOs cannot be online simultaneously. Although DOM[3] keeps the encryption key a, (s)he cannot directly give it to DU without DOs' permissions. Assume that there are t DOs are on-line, namely $\{O'_1, \cdots, O'_t\}$, and each DO ($O'_{i'}(i' \in [1,t])$) keeps the point $(x'_{i'}, y'_{i'})$ in $f(x)$, then they communicate with the DU as follows:

- Each $O'_{i'}$ first selects $y^*_{i'} \in \mathbb{Z}_p$ and computes $\hat{y}_{i'} = (y'_{i'} - y^*_{i'}) \bmod p$, $q^*_{i'} = g^{y^*_{i'}}$, where $q^*_{i'}$ is public. Then, $O'_{i'}$ sends $y^*_{i'}, \hat{y}_{i'}$ to $O'_{j'}(j' \in [1,t], j' \neq i')$, DU via SSL, respectively.
- When $O'_{j'}$ receives the element $y^*_{i'}$, (s)he first checks $q^*_{i'} \overset{?}{=} g^{y^*_{i'}}$. If this equation holds, then $O'_{j'}$ accepts $y^*_{i'}$ and forwards it to the DU via SSL; otherwise, (s)he rejects.
- Upon obtaining $\{(\hat{y}_1, y^*_1), \cdots, (\hat{y}_t, y^*_t)\}$, DU fist computes $y'_{i'} = (\hat{y}_{i'} + y^*_{i'}) \bmod p$, and then deduces $f(0) = a$ by utilizing the Lagrange interpolation method (see E-q. 3). Finally, DU decrypts these results C^* using the corresponding secret key a.

$$f(0) = \sum_{i'=1}^{t} y'_{i'} \Big(\prod_{j'=1, j' \neq i'}^{t} \frac{-y'_{j'}}{y'_{i'} - y'_{j'}} \Big) \bmod p. \tag{3}$$

[3]In some scenarios, DOs may not completely trust DOM, since DOM may distribute inconsistent secret shares to various DOs. To solve this challenge, DOM should calculate the commitments $\{g^{a_0}, \cdots, g^{a_{t-1}}\}$ on all coefficients $\{a_0, \cdots, a_{t-1}\}$ of $f(x)$ when computing each point (x_r, y_r) for each DO [42], so that each DO is able to verify the validity of (x_r, y_r) by using $g^{y_r} \overset{?}{=} \prod_{i'=0}^{t-1} g^{a_{i'} x_r^{i'}} = g^{\sum_{i'=0}^{t-1} a_{i'} x_r^{i'}} = g^{f(x_r)}$.

The straightforward way is that each DU can directly communicate with at least t (or the threshold value specified by DOM) DOs and gain their respective secret keys $\{y'_{i'}\}$. However, there exist no interactions among multiple DOs, and our FairDynDSF cannot provide flexible access control according to each DO's preference. For example, one may prefer to grant the decryption permission on the condition that his/her friends or leaders also permit without leaking his/her secret share. Hence, this decryption authorization can be deployed elastically according to the actual system requirements.

Remarks. In FairDynDSF, we consider both malicious DOM and semi-trusted PCS at the same time. In addition to realizing common functionalities (e.g., supporting multi-keyword search, checking the correctness of search results, resisting the stronger threat such as inside KGAs) required in conventional SE solutions, FairDynDSF achieves fair arbitration for dynamic update operations by using the signature exchange mechanism. Furthermore, FairDynDSF can be applied in the multi-owner setting, which achieves flexible access control over decryption authorization by using (d,t)-secret sharing mechanism. Finally, FairDynDSF is highly efficient by decreasing costly hash operations (e.g., the operation of a hash function H), in comparison to prior schemes.

V. Security and Performance Analysis

To demonstrate that the PDF is secure and efficient in practice, we show the detailed security and performance analysis, respectively.

A. Security

FairDynDSF not only withstands the inside KGAs in Theorem 1 but also guarantees the signature unforgeability in Theorem 2. Furthermore, the arbitration mechanism in FairDynDSF offers secure and fair arbitration in Theorem 3.

To resist the inside KGAs, the security of FairDynDSF requires that it should achieve both index and trapdoor indistinguishability in Theorem 1. Note that the index indistinguishability in game 1 allows $\mathcal{A}$ to enquire the private key and trapdoor ciphertext except for the trapdoor of challenging keywords w_0, w_1 of his choice, the trapdoor indistinguishability in game 2 permits $\mathcal{A}$ to enquire the secret key and index ciphertext apart from the index ciphertext corresponding to the challenging keywords w_0, w_1.

Theorem 1. FairDynDSF is secure against the inside KGAs in the random oracle model if no adversaries can break the game 1 and game 2 with non-negligible probability.

Proof: The proof of Theorem 1 depends on the following two lemmas, which protects the index indistinguishability and trapdoor indistinguishability, respectively. Note that the detailed proofs of these two lemmas can refer to schemes [38], [39].

Lemma 1. FairDynDSF realizes the index indistinguishability in game 1 on condition that the strong Decisional Diffie-Hellman (DDH) assumption [43] holds.

Lemma 2. FairDynDSF guarantees the trapdoor indistinguishability to fight the chosen-keyword attack in the random oracle model in game 2.

Given the trapdoor $T_{W'}$ based on the queried keyword set W', the semi-trusted PCS cannot output the valid keyword indexes that match with $T_{W'}$ unless he obtains DOM's secret key. Assume that PCS chooses $\tilde{a} \in \mathbb{Z}_p$ and calculates $g_0^{\tilde{a}}$, in order to create the index ciphertexts $\tilde{I} = (\tilde{I}_0, \tilde{I}_1, \tilde{I}_2, \{\tilde{I}_j\})$. According to the test equation in **Search**, the randomly chosen index ciphertexts $\tilde{I}$ satisfy the specified trapdoor $T_{W'}$ on condition that $\tilde{a} = a$, $W' = \{w'_1, \cdots, w'_l\} = \{\tilde{w}_1, \cdots, \tilde{w}_l\}$. However, the probability of choosing $\tilde{a} \in \mathbb{Z}_p$ such that $\tilde{a} = a$ is $\frac{1}{p}$, which is negligible when the value of p is large enough. Besides, the probability of finding l proper keywords from the keyword dictionary $\mathcal{W} = \{w_1, \cdots, w_n\}$ is $1/\binom{n}{l}$ $(n \gg l)$. Hence, PCS is unable to issue inside KGAs by calculating all possible keyword indexes to pass the matching mechanism.

Given the index ciphertexts $I = (I_0, I_1, I_2, \{I_j\})$, PCS still does not have the ability to generate corresponding trapdoor $\tilde{T}_{W'}$ which matches with I. Even though PCS is permitted to choose $\tilde{b} \in \mathbb{Z}_p$ and $\tilde{W}' = \{\tilde{w}'_1, \cdots, \tilde{w}'_l\}$ before outputting $\tilde{T}_{W'}$, the probability of $\tilde{b} = b$ is still negligible. Thus, PCS also cannot perform the inside KGAs by producing the valid trapdoor of possible keyword set. Besides, the equation used to match indexes and trapdoors cannot contribute to finding the valid trapdoor or executing the inside KGAs as $T_{W'}$ includes the random element. Based on the above analysis, FairDynDSF is secure against the inside KGAs in the random oracle model. This completes the proof of Theorem 1. ■

Similar to the scheme [40], we claim that the EMR signature is existentially unforgeable in Theorem 2.

Theorem 2. It is computationally infeasible to forge the legal EMR signatures in FairDynDSF on condition that the Computational Diffie-Hellman (CDH) assumption holds.

Proof: The signature unforgeability implies that $\mathcal{A}$ cannot output the valid signature. Let $\mathcal{A}$ be a $(\tilde{t}, \tilde{\epsilon})$-algorithm, which forges the signature in FairDynDSF with the time $\tilde{t}$, and the probability $\tilde{\epsilon}$, then there exists a (t^*, ϵ^*)-algorithm $\mathcal{C}$ which can break the CDH assumption, where $t^* \geq \tilde{t} + q_H\tilde{e} + q_S\tilde{e}$, $\epsilon^* \geq \tilde{\epsilon}/q_H q_P$, $\tilde{e}$ denotes the operation time of one exponentiation in $\mathbb{G}$. Note that $\mathcal{A}$ can issue at most q_H hash queries, q_S signature queries and q_P public key queries. The detailed proof of Theorem 2 is shown in **Supplemental Material B**. ■

Finally, FairDynDSF can offer secure and fair arbitration for potential disputes happening between DOM and PCS, which can be achieved by Theorem 3.

Theorem 3. The arbitration mechanism in FairDynDSF can offer secure and fair arbitration on condition that the signature is unforgeable and the result verification is correct.

Proof: First, we need to show that the arbitration mechanism is correct. The signatures signed by DOM and PCS on the $\tilde{r}$-version of index switcher $\theta_{\tilde{r}}$ are set as $S_o = H(\tilde{r}||\theta_{\tilde{r}})^{sk_o}$, $S_s = H(\tilde{r}||\theta_{\tilde{r}})^{sk_s}$, respectively. In the initial stage, the index switcher is $\{(t_i = i, i)\}$ and the value of update count pointer is 0. When DOM first uploads his EMRs to PCS, PCS must verify whether DOM has correctly created the signature for each EMR. As the initial θ_0 is public to all, FAS can quickly deal with the conflicts and accomplish the initial signature exchange. At this point, DOM and PCS both have each other's signature, namely DOM stores $S_s = H(0||\theta_0)^c$, PCS keeps $S_o = H(0||\theta_0)^a$. As for each update operation, DOM should return the update information (e.g., operation type, operation location, the newly-produced signature index) to PCS so that it can re-construct the updated index switcher $\theta_{\tilde{r}}$ and further check the correctness of DOM's signature on $\theta_{\tilde{r}}$. Due to space limitation, the detailed security analysis of Theorem 3 is demonstrated in **Supplemental Material C**.

We also need to show the fairness of the arbitration protocol. Assume that the malicious PCS intends to accuse the honest DOM, it should have the ability to forge the correct signature of DOM by utilizing the larger value of ϕ rather than the agreed one in the last completed update. Let the update count pointer and index switcher in the last successful update be seq, θ, respectively, then DOM owns $S_s(seq||\theta)$ and PCS has $S_o(seq||\theta)$. Suppose that DOM wants to accuse PCS, he should output the correct signature $S_s(seq'||\theta')$, where $seq' > seq$, $\theta' \neq \theta$. If PCS attempts to accuse DOM, it should generate the correct signature $S_o(seq''||\theta'')$, where $seq'' > seq + 1$, $\theta'' \neq \theta$. Note that θ', θ'' denote the updated index switcher. However, these two situations conflict the property of signature unforgeability, which is proved in Theorem 2. That is to say, there exist no adversaries that can forge the valid signature with a non-negligible advantage. This completes the proof of Theorem 3. ■

B. Performance

The performance of FairDynDSF is analyzed in terms of theoretical analysis and empirical tests, by comparing with the state-of-the-art solutions [29], [35], namely the Verifiable Keyword Search with Fine-grained authorization control (VKSF) scheme [29] and Fair Remote Retrieval (FRR) scheme [35]. Although the VKSF and FRR schemes allow PCS and DOM to verify each other's data, these schemes do not offer the remedial solutions to eliminate the dispute concerns in the dynamic update.

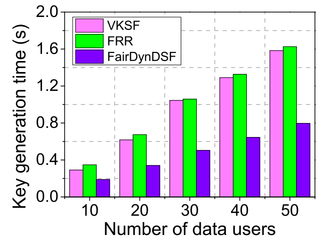
1) Theoretical analysis: As for the computation overhead of theoretical analysis, we first focus on several time-consuming operations including the pairing operation $\mathbb{P}$, modular exponentiation operations $\mathbb{E}, \mathbb{E}_T$ in $\mathbb{G}, \mathbb{G}_T$, hash operation $\mathbb{H}$ which maps each arbitrary string to a random element in $\mathbb{G}$. It is worth noticing that another hash operation, which maps each string of arbitrary length into $\mathbb{Z}_p$, is much more efficient than above operations and then omitted in theoretical analysis. To further analyze the storage overhead of FairDynDSF, we define the element lengths in $\mathbb{G}$, $\mathbb{G}_T$, $\mathbb{Z}_p$ as $|\mathbb{G}|$, $|\mathbb{G}_T|$, $|\mathbb{Z}_p|$, respectively. Next, we compare the computation and storage overhead of aforementioned three schemes in TABLE III.

It can be seen from the comparison in TABLE III, FairDynDSF is superior to other two schemes in **KeyGen**, **Trap** and **Search** algorithms in terms of computation and storage overhead, note that the FRR scheme cannot offer keyword-based ciphertext retrieval in **Trap** and **Search** algorithms. As

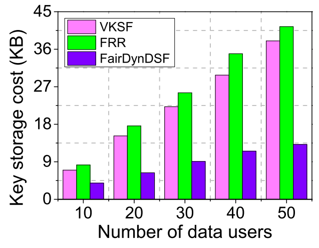
TABLE III: Theoretical computation and storage analysis: A comparative summary

Algorithms	VKSF [29]		FRR [35]		FairDynDSF	
	Computation costs	Storage costs	Computation costs	Storage costs	Computation costs	Storage costs
KeyGen	$\|\mathcal{U}\|(\mathbb{E}+\|\mathcal{S}\|\mathbb{E})$	$\|\mathcal{U}\|(\|\mathbb{G}\|+\|\mathcal{S}\|\|\mathbb{G}\|)$	$2\mathbb{E}+\|\mathcal{U}\|(\mathbb{E}+\|\mathcal{T}\|\mathbb{E})$	$2\Theta_1+\|\mathcal{U}\|(\|\mathcal{T}\|+1)\Theta_1$	$10\mathbb{E}+\|\mathcal{U}\|(\mathbb{P}+\mathbb{E}+\mathbb{E}_T)$	$\|\mathcal{U}\|\Theta_2+2\|\mathbb{Z}_p\|+10\|\mathbb{G}\|$
Enc	$\mathbb{E}_T+\mathbb{E}(n+\|\mathcal{N}\|)$	$\|G_T\|+(n+\|\mathcal{N}\|)\|\mathbb{G}\|$	$\|\mathcal{L}\|(\mathbb{H}+2\mathbb{E}+\mathbb{P}+\mathbb{E}_T)$	$\|\mathcal{L}\|(\|\mathbb{Z}_p\|+\|\mathbb{G}\|)$	$(n+d+2)\mathbb{E}+\Theta_3$	$(n+3)\|\mathbb{G}\|+2\|\mathbb{G}_T\|+d\Theta_1$
Update	—	—	—	—	$10\mathbb{P}+12\mathbb{H}+12\mathbb{E}$	$10\|\mathbb{G}_T\|+8\|\mathbb{G}\|$
Trap	$\mathbb{E}_T+l\mathbb{E}$	$\|\mathbb{G}_T\|+l\|\mathbb{G}\|$	—	—	$2\mathbb{E}$	$\|\mathbb{G}\|+\|\mathbb{Z}_p\|$
Search	$l\mathbb{P}$	$l\|\mathbb{G}_T\|$	—	—	$\mathbb{E}_T+\mathbb{P}$	$2\|\mathbb{G}_T\|$
Verify	$4q\mathbb{P}$	$4q\|\mathbb{G}_T\|$	$2q\mathbb{E}+q\mathbb{H}+3\mathbb{P}+\mathbb{E}+\mathbb{E}_T$	$(q+1)\|\mathbb{Z}_p\|+\|\mathbb{G}\|+2\|\mathbb{G}_T\|$	$2q\mathbb{E}+q\mathbb{H}+2\mathbb{P}+\mathbb{E}$	$(2q+1)\|\mathbb{Z}_p\|+\|\mathbb{G}\|+2\|\mathbb{G}_T\|$
Dec	$q\|\mathcal{S}\|(\mathbb{E}_T+\mathbb{P})$	$q(\|\mathcal{S}\|+1)\|\mathbb{G}_T\|$	$q\|\mathcal{T}\|^2\mathbb{E}$	$3q\|\mathcal{T}\|\|\mathbb{Z}_p\|+2q\|\mathcal{T}\|\|\mathbb{G}\|$	$t^2q\mathbb{E}$	$3qt\|\mathbb{Z}_p\|+qt\|\mathbb{G}\|$

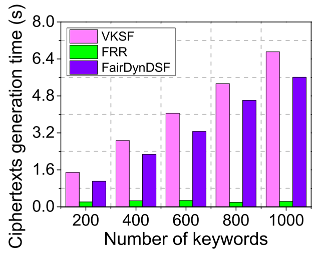
Notes. $\Theta_1 = |\mathbb{Z}_p| + |\mathbb{G}|$; $\Theta_2 = |\mathbb{Z}_p| + |\mathbb{G}| + |\mathbb{G}_T|$; $\Theta_3 = 2\mathbb{E} + \mathbb{P} + 2\mathbb{H} + 2\mathbb{E}_T$;
"$|\mathcal{U}|$": Number of DUs; "$|\mathcal{S}|$": Number of DU's attributes in the VKSF; "$|\mathcal{T}|$": Number of committees in the FRR; "$|\mathcal{N}|$": Number of leaf nodes in the access structure of VKSF; "$|\mathcal{L}|$": Number of blocks in each record in the FRR; "d": Number of DOs in FairDynDSF; "n": Number of keywords in keyword set $\mathcal{W}$; "l": Number of queried keywords; "q": Number of search results or challenging results; "t": Thread number of DOs in FairDynDSF.



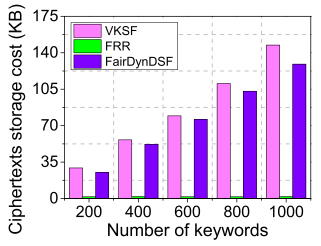
(a) Computation costs in **Keygen**



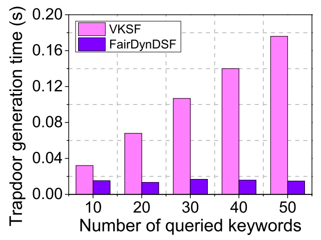
(b) Storage costs in **Keygen**



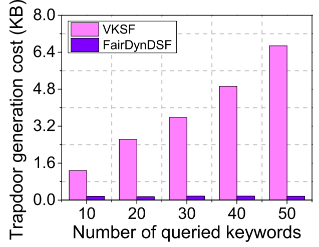
(c) Computation costs in **Enc**



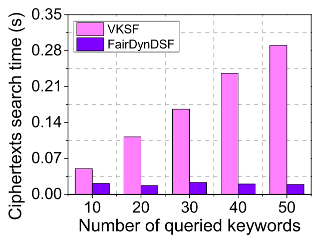
(d) Storage costs in **Enc**



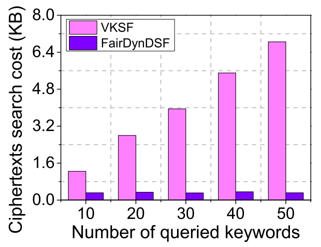
(e) Computation costs in **Trap**



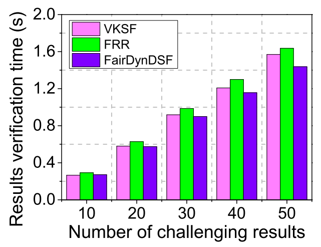
(f) Storage costs in **Trap**



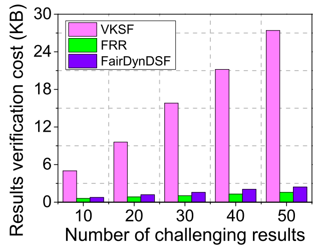
(g) Computation costs in **Search**



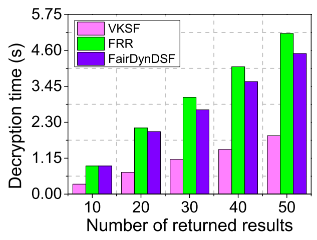
(h) Storage costs in **Search**



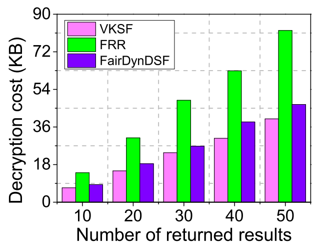
(i) Computation costs in **Verify**



(j) Storage costs in **Verify**



(k) Computation costs in **Dec**



(l) Storage costs in **Dec**

Fig. 6: Practical performance analysis in various schemes.

the FRR does not need to build indexes according to keyword set $\mathcal{W}$ in **Enc**, the ciphertexts generation time and cost in the FRR is much less than those of VKSF and FairDynDSF. As for result verification mechanism in **Verify**, which depends on the number of returned results, FairDynDSF still does not incur extra computation and storage costs. A number of DOs have to conduct mutual authentications before delegating the ciphertexts decryption permission to DU in **Dec**, which can realize the flexible access control, but FairDynDSF has less computation and storage costs when compared with the FRR scheme. It is worth noticing that the VKSF just checks the legitimacy of DU's attributes without executing interactions between multiple DOs as this scheme cannot be applied in the multi-owner setting. In conclusion, FairDynDSF is feasible according to its theoretical analysis.

2) Empirical Tests: Next, we demonstrate the actual performance analysis by performing the empirical tests over real-world Enron Email dataset[4]. Note that this public dataset has a size of 422 MB and consists of about 517431 emails from 151 users distributed in 3500 folders. Besides, we conduct the simulation on an Ubuntu Server 15.04 with Intel Core i5-7200 CPU 2.5 GHz by utilizing the C language as well as Paring Based Cryptography (PBC) Library. In the process of experiments, we select the Type A in PBC as $E(F_q) : y^2 = x^3 + x$, and let $\mathbb{G}, \mathbb{G}_T$ of order p be the subgroups of $E(F_q)$, where

[4]http://www.cs.cmu.edu/~enron/

the lengths of p and q are defined as 160 bits and 512 bits, respectively. For comparison, we set $|\mathcal{S}| = |\mathcal{T}| = |\mathcal{L}| = t = 5$, $d = 9(2t-1 \geq d)$, $|\mathcal{N}| = 50$, $|\mathcal{U}| = q = l \in [1, 50]$, $n \in [1, 1000]$ in the throughout paper.

In **KeyGen**, the VKSF outputs the secret key for each DU with attributes $\mathcal{S}$, while FRR and FairDynDSF also generate secret keys for other entities (e.g., $\mathcal{T}$ committees in the FRR, d DOs in FairDynDSF) except for DUs. Hence, we set $|\mathcal{S}| = |\mathcal{T}| = 5$, $d = 9$ and vary the value of $|\mathcal{U}|$ from 1 to 50. From Fig. 6a we notice that the key generation time linearly increases with $|\mathcal{U}|$. FairDynDSF scheme has less computation cost than other two schemes as it just executes $(\mathbb{P} + \mathbb{E} + \mathbb{E}_T)$ for each DU, while the other two schemes need $6\mathbb{E}$ for each DU. With the same reason shown in Fig. 6a, FairDynDSF also greatly reduces the storage cost of key generation in Fig. 6b. For example, FairDynDSF mainly stores $(|\mathbb{Z}_p| + |\mathbb{G}| + |\mathbb{G}_T|)$ for each DU, and the VKSF and FRR should store $6|\mathbb{G}|$, $6(|\mathbb{Z}_p| + |\mathbb{G}|)$, respectively.

To enable keyword-based ciphertexts search, VKSF and FairDynDSF need to build indexes according to keyword dictionary $\mathcal{W}$. Thus, the computation and storage costs of entire ciphertexts in VKSF and FairDynDSF are much more than those of FRR due to $n \gg |\mathcal{L}|$. In here, we set $|\mathcal{L}| = 5$ so that FRR can accelerate the generation of proof information or result verification. As shown in Fig. 6c, Fig. 6d, the performance of **Enc** in VKSF and FairDynDSF for each record linearly depends on the variable $n \in [1, 1000]$, while that of FRR remains nearly constant. Since the VKSF also creates the ciphertexts for the leaf nodes in the specified access policy, and DOM in FairDynDSF distributes the secret share for each DO, FairDynDSF still outperforms the VKSF in terms of ciphertexts generation time and storage overhead by setting $|\mathcal{N}| = 50$, $d = 9$. However, **Enc** can be implemented once and for all, which cannot negatively impact the DUs' search experience.

As the FRR scheme does not offer keyword search functionality, we analyze the performance of **Trap** and **Search** in both VKSF and FairDynDSF. It is apparent from Fig. 6e, Fig. 6f, Fig. 6g, Fig. 6h that FairDynDSF has approximately constant trapdoor generation and ciphertexts retrieval overhead, which is a particularly important feature for the resource-limited IoT devices. As the hash operation $h : \{0,1\}^* \to \mathbb{Z}_p$ in **Trap** and modular multiplication operation in **Search** are much more efficient than the concerned operations (i.e., $\mathbb{P}, \mathbb{E}, \mathbb{E}_T$) in the theoretical analysis, the computation costs of these two algorithms in FairDynDSF remain nearly constant with the number of queried keywords. For instance, FairDynDSF mainly needs to execute $2\mathbb{E}$ and $\mathbb{P} + \mathbb{E}_T$ to output the trapdoor and retrieve encrypted EMRs of interest, no matter how many keywords he queries. Besides, the storage costs of **Trap** and **Search** in FairDynDSF are $(|\mathbb{G}| + |\mathbb{Z}_p|)$, $2|\mathbb{G}_T|$, respectively, which are also not affected by the number of queried keywords. However, the performance of these two algorithms in VKSF is affected by the number of queried keywords ($l \in [1, 50]$) as the VKSF cannot implement the multi-keyword search.

In Fig. 6i, Fig. 6j, we demonstrate the performance of result correctness (or data integrity) verification in **Verify**, which varies with the number of returned results ($q \in [1, 50]$). The VKSF scheme takes about $4|\mathbb{P}|$ to check the authenticity of each search result by using the invertible Bloom lookup table and Merkle hash tree, and both FRR and FairDynDSF need about $2\mathbb{E} + \mathbb{H}$ to finish the same task. As $\mathbb{H}$ costs much more time than $\mathbb{E}, \mathbb{P}$, these three schemes have approximately equal computation overhead in Fig. 6i. However, both FairDynDSF and FRR still have less storage overhead than VKSF. With regard to the storage overhead in **Verify**, FairDynDSF and FRR need about $(2q+1)|\mathbb{Z}_p|$, $(q+1)|\mathbb{Z}_p|$, respectively, but the VKSF needs about $4q|\mathbb{G}_T|$.

To decrypt the encrypted results, the DU in the VKSF should gain the authorization from the DO by checking whether his attributes satisfy the specified access policy, which takes $|\mathcal{S}|(\mathbb{E}_T + P)$ operations for each result. However, FairDynDSF and FRR should undergo the interactive process before gaining the encryption key. To decrypt search result in **Dec**, the FRR must conduct $|\mathcal{T}|^2\mathbb{E}$ before each committee obtains the encryption key, FairDynDSF should take $t^2\mathbb{E}$ so that the DU can restore the encryption key by gaining at least t authorizations from DOs. Thus, the computation and storage overhead of results decryption in VKSF is much less than that of FRR and FairDynDSF, which is shown in Fig. 6k and Fig. 6l. With supporting the multi-owner setting, FairDynDSN needs to conduct extra $t(t-1)\mathbb{E}$ caused by interactions among multiple DOs. Thus, the computation and storage overhead of **Dec** in FairDynDSN is less efficient than that of VKSF. However, when applied in the single owner setting, FairDynDSN is efficient than VKSF, and can achieve flexible access control and threshold decryption privilege. When compared with the FRR scheme, FairDynDSF has similar computation cost in Fig. 6k, but has less storage cost in Fig. 6l, as the FRR needs to store the additional $q|\mathcal{T}||\mathbb{G}|$ or $qt|\mathbb{G}|$.

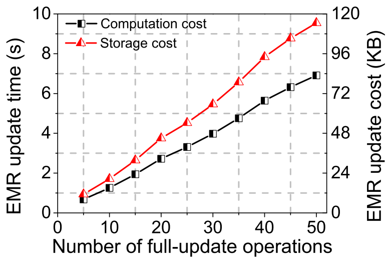


Fig. 7: The performance of **Update** algorithm in FairDynDSF.

As both VKSF and FRR schemes cannot support dynamic data update in **Update**, we just show FairDynDSF's computation and storage costs of one full-update operation including modification, insertion and deletion in Fig. 7, which almost linearly grows with the number of full-update operations. Note that FairDynDSF needs to execute $4\mathbb{P} + 5\mathbb{H} + 5\mathbb{E}$, $2\mathbb{P} + 2\mathbb{H} + \mathbb{E}$ for each modification (or insertion) operation and each deletion operation, respectively. Besides, FairDynDSF stores $4|\mathbb{G}_T| + 3|\mathbb{G}|$, $2|\mathbb{G}_T| + 2|\mathbb{G}|$ when conducting one modification (or insertion) and one deletion operation. When dealing with the possible disputes between the DOM and the PCS, the FAS first needs to verify the validity of exchanged signatures in the current update. As for case 1 in **Arbitrate**, FAS needs to execute $6\mathbb{P} + (q+2)\mathbb{H} + (2q+1)\mathbb{E}$ to terminate

TABLE IV: An example of actual tests in different datasets

Schemes	VKSF [29]						FRR [35]						FairDynDSF					
	Computation costs (s)			Storage costs (KB)			Computation costs (s)			Storage costs (KB)			Computation costs (s)			Storage costs (KB)		
Datasets	Enron	NSF	RFC	Enron	NSF	RFC	Enron	NSF	RFC	Enron	NSF	RFC	Enron	NSF	RFC	Enron	NSF	RFC
KeyGen	1.58	1.51	1.63	147.41	139.23	154.37	1.63	1.61	1.74	41.43	38.23	47.32	0.80	0.76	0.89	13.16	12.94	13.87
Enc	6.71	6.54	7.02	147.42	132.49	151.36	0.23	0.19	0.25	1.85	1.82	1.93	5.61	5.57	5.63	129.12	121.38	135.28
Trap	0.18	0.16	0.19	36.68	34.58	40.12	—	—	—	—	—	—	0.02	0.02	0.03	0.17	0.16	0.21
Search	0.29	0.25	0.32	6.85	6.62	7.13	—	—	—	—	—	—	0.02	0.01	0.04	0.32	0.29	0.38
Verify	1.57	1.42	1.69	27.41	25.78	28.34	1.64	1.61	1.73	1.57	1.54	1.62	1.44	1.42	1.45	2.42	2.36	2.45
Dec	1.87	1.84	1.99	39.91	38.25	41.03	5.14	4.98	5.22	82.21	79.23	84.57	4.51	4.49	4.58	46.81	45.67	48.12

Notes. $|\mathcal{S}| = |\mathcal{T}| = |\mathcal{L}| = t = 5$, $d = 9$, $|\mathcal{N}| = 50$ throughout all algorithms; $\mathcal{U} = 50$ in **KeyGen**; $n = 1000$ in **Enc**; $l = 50$ in **Trap**, **Search**; $q = 50$ in **Verify**, **Dec**.

this kind of disputes completely. About the other two cases in **Arbitrate**, the FAS should conduct $12\mathbb{P} + 6\mathbb{H} + 3\mathbb{E}$ to end these kinds of disputes. However, in practice, these disputes happen occasionally, or at least not very often. Hence, the fair arbitration in FairDynDSF is still acceptable.

To further demonstrate the performance of FairDynDSN and the other two schemes (i.e., VKSF, FRR, etc.) in other datasets such as NSF dataset (National Science Foundation Research Awards Abstract 1990-2003 dataset)[5] and RFC dataset (Request For Comments database)[6], we also perform a series of experiments on these three datasets (i.e., Enron Email dataset, NSF dataset, RFC dataset), which are shown in TABLE IV. Although these results are different in three datasets, the conclusion of performance comparison in Enron dataset is consistent with those in NSF dataset and RFC dataset.

VI. Conclusion

With cloud-assisted IoE being increasingly popular, there is a need to design cryptographic schemes that can be deployed on a broad range of heterogeneous devices. Therefore, in this paper, we proposed an efficient and practical FairDynDSF, which supports result verification, dispute arbitration, dynamic update, decryption authorization and expressive keyword search simultaneously. In addition, FairDynDSF is also designed to be resilient to data corruption attacks and sufficiently lightweight for deployment on resource-constrained IoT devices. The formal security analysis showed that FairDynDSF is secure against inside KGAs, and the empirical examination using various datasets demonstrated that FairDynDSF is practical and scalable in practice.

Future research includes improving the performance of encryption and result verification, as well as implementing a prototype of the improved framework in a real-world setting for further evaluation.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. 61702404, No. 61702105), the China Postdoctoral Science Foundation Funded Project (No. 2017M613080), the Fundamental Research Funds for the Central Universities (No. 20101196508), the project funded by China Scholarship Council (No. 201806965010), the Singapore National Research Foundation under the NCR Award (No. NRF2014NCR-NCR001-012) and the AXA Research Fund.

[5]http://kdd.ics.uci.edu/databases/nsfabs/nsfawards.html

[6]http://www.ietf.org/rfc.html

References

[1] B. Xu, L. Da Xu, H. Cai, C. Xie, J. Hu, F. Bu *et al.*, "Ubiquitous data accessing method in iot-based information system for emergency medical services." *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1578–1586, 2014.

[2] M. B. Mollah, M. A. K. Azad, and A. Vasilakos, "Secure data sharing and searching at the edge of cloud-assisted internet of things," *IEEE Cloud Computing*, no. 1, pp. 34–42, 2017.

[3] A. Castiglione, K.-K. R. Choo, M. Nappi, and S. Ricciardi, "Context aware ubiquitous biometrics in edge of military things," *IEEE Cloud Computing*, vol. 4, no. 6, pp. 16–20, 2017.

[4] A. Kumari, S. Tanwar, S. Tyagi, N. Kumar, M. Maasberg, and K.-K. R. Choo, "Multimedia big data computing and internet of things applications: A taxonomy and process model," *Journal of Network and Computer Applications*, vol. 124, pp. 169–195, 2018.

[5] F. Tao, Y. Cheng, L. Da Xu, L. Zhang, and B. H. Li, "Cciot-cmfg: cloud computing and internet of things-based cloud manufacturing service system," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1435–1442, 2014.

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

[7] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. International conference on the theory and applications of cryptographic techniques (EUROCRYPT'04)*. Springer, 2004, pp. 506–522.

[8] Y. Miao, J. Ma, X. Liu, X. Li, Z. Liu, and H. Li, "Practical attribute-based multi-keyword search scheme in mobile crowdsourcing," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 3008–3018, 2018.

[9] 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.

[10] J. Ning, J. Xu, K. Liang, F. Zhang, and E.-C. Chang, "Passive attacks against searchable encryption," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 3, pp. 789–802, 2019.

[11] Y. Miao, J. Ma, X. Liu, J. Weng, H. Li, and H. Li, "Lightweight fine-grained search over encrypted data in fog computing," *IEEE Transactions on Services Computing*, vol. PP, pp. 1–14, 2018.

[12] H. Li, Y. Yang, T. H. Luan, X. Liang, L. Zhou, and X. S. Shen, "Enabling fine-grained multi-keyword search supporting classified sub-dictionaries over encrypted cloud data," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 3, pp. 312–325, 2016.

[13] R. Chen, Y. Mu, G. Yang, F. Guo, X. Huang, X. Wang, and Y. Wang, "Server-aided public key encryption with keyword search," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 12, pp. 2833–2842, 2016.

[14] Y. Miao, J. Ma, X. Liu, X. Li, Q. Jiang, and J. Zhang, "Attribute-based keyword search over hierarchical data in cloud computing," *IEEE Transactions on Services Computing*, vol. PP, pp. 1–14, 2017.

[15] Y. Miao, X. Liu, R. H. Deng, H. Wu, H. Li, J. Li, and D. Wu, "Hybrid keyword-field search with efficient key management for industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. PP, pp. 1–12, 2018.

[16] G. S. Poh, J.-J. Chin, W.-C. Yau, K.-K. R. Choo, and M. S. Mohamad, "Searchable symmetric encryption: designs and challenges," *ACM Computing Surveys*, vol. 50, no. 3, p. 40, 2017.

[17] Q. Chai and G. Gong, "Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers," in *Proc. IEEE International Conference on Communications (ICC'12)*. IEEE, 2012, pp. 917–922.

[18] 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.

[19] Y. Miao, J. Weng, X. Liu, K.-K. R. Choo, Z. Liu, and H. Li, "Enabling verifiable multiple keywords search over encrypted cloud data," *Information Sciences*, vol. 465, pp. 21–37, 2018.

[20] X. Jiang, J. Yu, J. Yan, and R. Hao, "Enabling efficient and verifiable multi-keyword ranked search over encrypted cloud data," *Information Sciences*, vol. 403, pp. 22–41, 2017.

[21] W. Sun, X. Liu, W. Lou, Y. T. Hou, and H. Li, "Catch you if you lie to me: Efficient verifiable conjunctive keyword search over large dynamic encrypted cloud data," in *Proc. IEEE Conference on Computer Communications (INFOCOM'15)*. IEEE, 2015, pp. 2110–2118.

[22] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data." *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 340–352, 2016.

[23] H. Jin, H. Jiang, and K. Zhou, "Dynamic and public auditing with fair arbitration for cloud data," *IEEE Transactions on cloud computing*, vol. 6, no. 3, pp. 680–693, 2018.

[24] A. Küpçü, "Official arbitration with secure cloud storage application," *The Computer Journal*, vol. 58, no. 4, pp. 831–852, 2013.

[25] Y. Zhang, R. H. Deng, J. Shu, K. Yang, and D. Zheng, "Tkse: Trustworthy keyword search over encrypted data with two-side verifiability via blockchain," *IEEE Access*, 2018.

[26] W. Sun, S. Yu, W. Lou, Y. T. Hou, and H. Li, "Protecting your right: verifiable attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 4, pp. 1187–1198, 2016.

[27] Y. Miao, J. Ma, X. Liu, J. Zhang, and Z. Liu, "Vkse-mo: verifiable keyword search over encrypted data in multi-owner settings," *Science China Information Sciences*, vol. 60, no. 12, p. 122105, 2017.

[28] Z. Liu, T. Li, P. Li, C. Jia, and J. Li, "Verifiable searchable encryption with aggregate keys for data sharing system," *Future Generation Computer Systems*, vol. 78, pp. 778–788, 2018.

[29] Z. Chen, F. Zhang, P. Zhang, J. K. Liu, J. Huang, H. Zhao, and J. Shen, "Verifiable keyword search for secure big data-based mobile healthcare networks with fine-grained authorization control," *Future Generation Computer Systems*, vol. 87, pp. 712–724, 2018.

[30] J. Zhu, Q. Li, C. Wang, X. Yuan, Q. Wang, and K. Ren, "Enabling generic, verifiable, and secure data search in cloud services," *IEEE Transactions on Parallel and Distributed Systems*, vol. PP, pp. 1–14, 2018.

[31] Y. Miao, X. Liu, K.-K. R. Choo, R. H. Deng, J. Li, H. Li, and J. Ma, "Privacy-preserving attribute-based keyword search in shared multi-owner setting," *IEEE Transactions on Dependable and Secure Computing*, vol. PP, pp. 1–15, 2019.

[32] J. Liu, K. Huang, H. Rong, H. Wang, and M. Xian, "Privacy-preserving public auditing for regenerating-code-based cloud storage," *IEEE transactions on information forensics and security*, vol. 10, no. 7, pp. 1513–1528, 2015.

[33] J. Shen, J. Shen, X. Chen, X. Huang, and W. Susilo, "An efficient public auditing protocol with novel dynamic structure for cloud data," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 10, pp. 2402–2415, 2017.

[34] H. Tian, Y. Chen, C.-C. Chang, H. Jiang, Y. Huang, Y. Chen, and J. Liu, "Dynamic-hash-table based public auditing for secure cloud storage," *IEEE Transactions on Services Computing*, vol. 10, no. 5, pp. 701–714, 2017.

[35] H. Wang, Q. Wu, B. Qin, and J. Domingo-Ferrer, "Frr: Fair remote retrieval of outsourced private medical records in electronic health networks," *Journal of biomedical informatics*, vol. 50, pp. 226–233, 2014.

[36] F. Bao, R. H. Deng, and W. Mao, "Efficient and practical fair exchange protocols with off-line ttp," in *Proc. IEEE Symposium on Security and Privacy (SP'98)*. IEEE, 1998, pp. 77–85.

[37] Q. Huang, D. S. Wong, and W. Susilo, "How to protect privacy in optimistic fair exchange of digital signatures," *Information Sciences*, vol. 325, pp. 300–315, 2015.

[38] Q. Huang and H. Li, "An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks," *Information Sciences*, vol. 403, pp. 1–14, 2017.

[39] R. Xie, C. He, D. Xie, C. Gao, and X. Zhang, "A secure ciphertext retrieval scheme against insider kgas for mobile devices in cloud storage," *Security and Communication Networks*, vol. 2018, 2018.

[40] B. Wang, B. Li, and H. Li, "Panda: Public auditing for shared data with efficient user revocation in the cloud," *IEEE Transactions on services computing*, vol. 8, no. 1, pp. 92–106, 2015.

[41] R. Goyal, S. Hohenberger, V. Koppula, and B. Waters, "A generic approach to constructing and proving verifiable random functions," in *Proc. Theory of Cryptography Conference (TCC'17)*. Springer, 2017, pp. 537–566.

[42] P. Feldman, "A practical scheme for non-interactive verifiable secret sharing," in *Proc. Annual Symposium on Foundations of Computer Science (SFCS'87)*. IEEE, 1987, pp. 427–438.

[43] Y. T. Kalai, X. Li, A. Rao, and D. Zuckerman, "Network extractor protocols," in *Proc. Annual IEEE Symposium on Foundations of Computer Science (FOCS'08)*. IEEE, 2008, pp. 654–663.

Yinbin Miao (M'18) received the B.E. degree with the Department of Telecommunication Engineering from Jilin University, Changchun, China, in 2011, and Ph.D. degree with the Department of Telecommunication Engineering from Xidian University, Xi'an, China, in 2016. He is also a postdoctor in Nanyang Technological University from September 2018 to September 2019. 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. He is a member of the IEEE.

Ximeng Liu received the B.E. degree with the Department of Electronic Engineering from Xidian University, Xi'an, China, in 2010 and Ph.D. degree with the Department of Telecommunication Engineering from Xidian University, Xi'an, China in 2015. He is currently a post-doctor 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.

Kim-Kwang Raymond Choo (SM'15) received the Ph.D. in Information Security in 2006 from Queensland University of Technology, Australia. He currently holds the Cloud Technology Endowed Professorship at The University of Texas at San Antonio (UTSA). He is the recipient of the UTSA College of Business Col. Jean Piccione and Lt. Col. Philip Piccione Endowed Research Award for Tenured Faculty in 2018, IEEE TrustCom 2018 and ESORICS 2015 Best Paper Awards, Fulbright Scholarship, etc. He is also an Australian Computer Society Fellow and the Co-Chair of IEEE Multimedia Communications Technical Committee's Digital Rights Management for Multimedia Interest Group.

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

Hongjun Wu received the B.Eng. and M.Eng. degrees from the National University of Singapore in 1998 and 2000, respectively, and the Ph.D. degree from the Katholieke Universiteit Leuven in 2008. He was a Nanyang Assistant Professor from 2010 to 2016. He has been an Associate Professor with the School of Physical and Mathematical Sciences, Nanyang Technological University, since 2016. His research interests include cryptography, cryptanalysis, and computer security.

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. His research interests include network security, applied cryptography, and trusted computing. He is a member of IEEE, a member of China Computer Federation and a member of China Association for Cryptologic Research.