8-2019

Multi-authority attribute-based keyword search over encrypted cloud data

Yibin MIAO

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

Ximeng LIU

Kim-Kwang Raymond. CHOO

Hongjun WU

See next page for additional authors

Citation

MIAO, Yibin; DENG, Robert H.; LIU, Ximeng; CHOO, Kim-Kwang Raymond.; WU, Hongjun; and LI, Hongwei. Multi-authority attribute-based keyword search over encrypted cloud data. (2019). *IEEE Transactions on Dependable and Secure Computing*. 1-14. Research Collection School Of Information Systems.
Available at: https://ink.library.smu.edu.sg/sis_research/5063

Advance online, 2019 Aug, DOI: 10.1109/TDSC.2019.2935044

Multi-authority Attribute-Based Keyword Search over Encrypted Cloud Data

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

Abstract—Searchable Encryption (SE) is an important technique to guarantee data security and usability in the cloud at the same time. Leveraging Ciphertext-Policy Attribute-Based Encryption (CP-ABE), the Ciphertext-Policy Attribute-Based Keyword Search (CP-ABKS) scheme can achieve keyword-based retrieval and fine-grained access control simultaneously. However, the single attribute authority in existing CP-ABKS schemes is tasked with costly user certificate verification and secret key distribution. In addition, this results in a single-point performance bottleneck in distributed cloud systems. Thus, in this paper, we present a secure Multi-authority CP-ABKS (MABKS) system to address such limitations and minimize the computation and storage burden on resource-limited devices in cloud systems. In addition, the MABKS system is extended to support malicious attribute authority tracing and attribute update. Our rigorous security analysis shows that the MABKS system is selectively secure in both selective-matrix and selective-attribute models. Our experimental results using real-world datasets demonstrate the efficiency and utility of the MABKS system in practical applications.

Index Terms—Searchable encryption, attribute-based encryption, multi-authority, selective-matrix model, selective-attribute model.

1 Introduction

With the convergence of cloud computing and Internet of Things (IoT) [1], cloud-assisted outsourcing services [2], [3], [4], [5] are becoming more commonplace. For example, outsourcing significant volume of data to a third-party cloud server, resource-limited devices (e.g., mobile terminals, sensor nodes) can minimize local data storage and computation requirements and facilitate the sharing of data (e.g., health records in a healthcare context) with other data users. However, privacy leakage is an inherent risk in data outsourcing. Hence, one typically deploys the encryption-before-outsourcing mechanism to achieve both data security and privacy in the semi-trusted or compromised cloud environment. This, however, restricts retrieval/searching over encrypted cloud data. Hence, the searchable encryption (SE) schemes [6], [7], [8], [9], [10], [11] have gained in popularity, since SE schemes allow one to securely search and selectively retrieve encrypted cloud data of interest based on user-specified keywords.

- *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; School of Physical and Mathematical Sciences, Nanyang Technological University, 21 Nanyang Link 637371, Singapore. E-mail: ybmiao@xidian.edu.cn*
- *R. H. Deng is with the Department of Information Systems and Department of Electrical and Computer Engineering, Singapore Management University, 80 Stamford Road 178902, Singapore. Email: robertdeng@smu.edu.sg*
- *X. Liu is with the Key Laboratory of Information Security of Network Systems, College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350108, China. Email: snbnix@gmail.com*
- *K.-K. R. Choo is with the Department of Information Systems and Cyber Security, 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*

Apart from the privacy-preserving information retrieval functionality, the fine-grained access control is also an essential functionality in cloud systems. Ciphertext-Policy Attribute-Based Keyword Search (CP-ABKS) scheme, for example, is a viable tool to achieve fine-grained access control and keyword-based ciphertexts retrieval simultaneously. Most existing CP-ABKS schemes [4], [5], [12], [13], [14] are designed for single attribute authority scenarios, where the single attribute authority needs to perform time-consuming user certificate verification [15] and secret key distribution. This also results in the single attribute authority being the single-point performance bottleneck (e.g., poor robustness and inefficiency) in large-scale distributed cloud systems. Should this single attribute authority be compromised or offline, then the cloud service will also be affected (e.g., being unavailable during that period). For example, data users may be stuck in the waiting queue for a long time before obtaining their corresponding secret keys. Such a single-point performance bottleneck can potentially degrade secret key generation performance, and affect CP-ABKS scheme availability. Traditional multi-authority ABE schemes [16], [17] in which each authority separately manages disjoint attribute sets also incur the same issue. For example, in multi-authority CP-ABE schemes, the DU's attributes (i.e., job, skill, health, etc.) are managed by various attribute authorities (i.e., talent market, authentication center, hospital, etc.). However, the DU still suffers from the above issue if one of the attribute authorities breaks down. Furthermore, simply combining previous multi-authority schemes also poses security concerns. For example, tracing a malicious authority that has issued, intentionally or unintentionally, incorrect secret keys for data users can be challenging.

The RAAC (Robust and Auditable Access Control) scheme [18] with heterogeneous architecture allows multiple Attribute Authorities (AAs) to independently conduct user certificate verification and generate the intermediate secret keys for data users on behalf of the Central Authority (CA). However, this scheme cannot support keyword-based ciphertexts retrieval. The latter is an extremely useful feature in information retrieval systems, to mitigate the issue of systems returning many irrelevant search results and resulting in bandwidth and computation resource wastage. Besides, most of existing CP-ABKS schemes focus on specifying expressive access structure, but the storage and computation costs in these schemes almost linearly increase with the number of system attributes rather than user attributes. Hence, such schemes are not suitable for resource-limited device deployments. Furthermore, the malicious AAs provided by third-parties may conduct incorrect operations (e.g., AAs may maliciously or incorrectly generate the intermediate secret key for the suspected data user, as shown in Section 5.2), and malicious DUs may access sensitive information by using outdated secret keys when their attributes have been updated in dynamic applications.

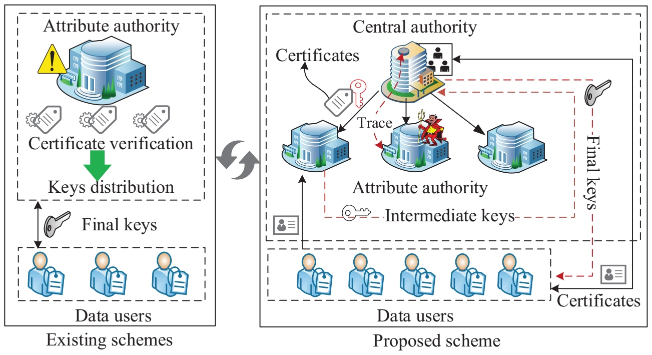


Fig. 1: Differences between MABKS system and prior schemes.

In light of these issues, we propose a **M**ulti-authority **A**ttribute-**B**ased **K**eyword **S**earch (MABKS) scheme for cloud systems to mitigate challenges due to single-point performance bottleneck and high storage and computation requirements (which are unrealistic for resource-limited devices). Key differences between multi-authority architecture in the MABKS system and single-authority architecture in existing schemes are presented in Fig. 1. Specifically, each AA in the MABKS system maintains the entire attribute set and is responsible for verifying the validity of data users' certificates and generating intermediate secret keys for data users, and the CA outputs the final secret keys for DUs. For example, the only fully-trusted department (that acts as CA) in a large company can generate the whole secret keys for staffs who are authorized to access important company documents, but will be burdened with much computation overhead when there are massive staffs, and even suffer from single-point performance bottleneck if this department is compromised or breaks down. The company can rent multiple public servers (that act as AAs) provided by other enterprises (i.e., Tencent, Amazon, Alibaba, etc.) to eliminate the fully-trusted department's computation burden (see Fig. 2) . However, these public servers may execute malicious operations and then return incorrect intermediate secret keys in order to save computation and bandwidth resources, as these servers and the fully-trusted department of this company are not in the same trusted domain. Furthermore, we cannot deploy multiple fully-trusted AAs in scheme constructions due to high communication overhead caused by building security channels. Fortunately, the CA in our MABKS system can trace the malicious AA. However, the traditional multi-authority CP-ABE schemes cannot achieve this goal, and even cannot avoid the possibility of single-point performance bottleneck that also exists in single-authority CP-ABE schemes. In summary, the main contributions of the MABKS system are shown as follows:

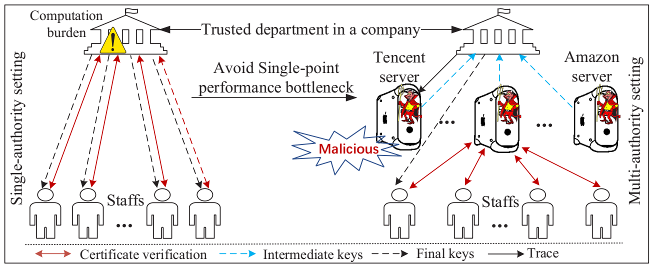


Fig. 2: An example for the multi-authority scenario.

- *Multi-authority architecture*. Different from the previous single-authority CP-ABKS schemes [13], [14] (or traditional multi-authority CP-ABE schemes [16], [17], [19]) that still cannot avoid the limitation of single-point performance bottleneck, the hierarchical structure in the MABKS system enables multiple AAs to separately execute time-consuming user certificate verification and intermediate secret key generation on behalf of CA, which significantly reduces CA's computation requirements.
- *File-level fine-grained keyword search*. Most of the traditional CP-ABKS schemes [4], [5], [12] have independent file key encryption and indexes building processes, while the MABKS system will embed the secret key chosen in file key encryption process into the indexes building process. Thus, the MABKS system not only allows data owners to specify the file-level fine-grained access control over encrypted cloud data but also enables cloud clients (e.g., data owners, data users) to perform keyword-based ciphertexts retrieval.
- *Malicious AAs tracing*. The traditional traceable CP-ABE schemes [20], [21], [22] mainly focus on the malicious data users who may leak their secret keys to unauthorized entities, while the extended MABKS system focuses on tracing the malicious AAs that incorrectly generate intermediate secret keys for data users in two phases (i.e., secret key ownership confirming, malicious AAs tracing).
- *Attribute update*. The extended MABKS system implements the attribute update so that malicious data users cannot access the sensitive cloud data by exploiting old or outdated secret keys. Compared with the attribute update mechanisms [23], [24] in prior CP-ABE schemes that need to update the whole ciphertexts, the extended MABKS just allows data users and cloud server to update a fraction of secret key components and indexes associated with the

updated attributes by using two transformation keys, respectively.

- *Security and efficiency*. The comprehensive security analysis shows that the MABKS system is selectively secure in both selective-matrix and selective-attribute models. Experimental results using real-world datasets demonstrate that the storage and computation overhead increases with the number of user attributes rather than system attributes [14]. In addition, the MABKS system has constant trapdoor size and ciphertexts retrieval overhead, which reduces the storage and computation burden on resource-limited data users and improves the user search experience. Considering that the encryption and decryption overhead still grows with the complexities of access policies in traditional CP-ABKS schemes [13], [14], the MABKS system can utilize the online/offline encryption mechanism and outsourced decryption mechanism [25] to further decrease the data owner and data users' computation overhead, respectively.

The remainder of our paper is structured as follows. Section 2 shows some prior work relating to the MABKS system. Section 3 demonstrates the preliminaries required in the understanding of the MABKS system. In Section 4, the problem formulation (including system model, scheme definition and security model) is presented. Then, we present the detailed construction of the MABKS system and its extensions in Section 5 and analyze the security and performance of the MABKS system in Section 6. Finally, we conclude this paper in Section 7.

2 Related Work

In cloud storage systems, data owners may outsource a large volume of security-critical and privacy-sensitive data for economical and/or operational reasons (e.g., to further reduce data storage and computation requirements). Although encryption mechanism can protect cloud data security and privacy to some extent, the encrypted cloud data retrieval becomes one of several key challenges faced by data users. To provide keyword-based information retrieval and fine-grained access control over encrypted cloud data, this paper particularly relates to CP-ABE (Ciphertext-Policy Attribute-Based Encryption) and SE schemes.

Since Boneh *et al.* [7] put forth the first Public-key Encryption with Keyword Search (PEKS) scheme, which enables cloud server to identify records containing user-specified keyword, a large number of versatile SE schemes have been presented (e.g., single keyword search [26], multi-keyword search [12], [27], ranked keyword search [28], [29], [30], verifiable keyword search [31], [32]). For example, Yang *et al.* [27] formed a novel conjunctive keyword search scheme with designated tester and timing enabled proxy re-encryption function, which allows a data owner to delegate his/her partial access rights to data users who are able to execute search operation in a limited period. To retrieve the most related files flexibly, Li *et al.* [29] gave a ranked multi-keyword search scheme by using the relevance scores and preference factors upon keywords, which supports the complicated logic search. Considering that the semi-trusted cloud server may execute a fraction of search tasks and output some false results, Sun *et al.* [32] first presented a verifiable conjunctive keyword search scheme, which can efficiently check the authenticity of search results and conduct file update operations. Despite attractive advantages (e.g., elastic accessibility, strong reliability, high availability) in cloud data outsourcing services, encryption mechanism on its own is not practical since data owners lose the direct physical control of remote cloud data.

Compared with traditional access control solutions, CP-ABE can achieve one-to-many encryption rather than one-to-one and has been regarded as a promising way to achieve fine-grained access control. Since Bethencourt *et al.* [33] presented the first CP-ABE scheme, which avoids storing the entire user-list and verifies user access permissions, there has been a number of extensions focusing on other problems, such as expressive access policies [24], [34], attribute update [35], [36], hierarchical access policies [37], hidden access policy [38] and verifiable outsourced decryption [39]. For instance, Balu *et al.* [34] proposed an expressive and provable CP-ABE scheme by leveraging the linear inter secret sharing technique, which significantly reduces the secret sharing costs. Zhang *et al.* [36] proposed a practical CP-ABE scheme, which offers user revocation and attribute update. As the ciphertext size and decryption cost grow with the complexities of access policies, Mao *et al.* [39] gave the generic construction of CPA (Chosen-Plaintext Attack)-secure CP-ABE scheme with verifiable outsourced decryption. Despite the number of research efforts on this topic, existing CP-ABE schemes have not entirely solved the problem of keyword-based data retrieval.

To tackle this problem, Attribute-Based Encryption (ABE) [33], [40] scheme has been extended to SE scheme. Such an extension is also referred to as ABKS [14], [38], [41] in the literature. Existing ABKS schemes are broadly divided into two categories [13], namely Key-Policy ABKS (KP-ABKS) [38], [41] and Ciphertext-Policy ABKS (CP-ABKS) [14]. CP-ABKS scheme allows one to achieve keyword-based ciphertexts retrieval and fine-grained access control simultaneously. For example, Zheng *et al.* [13] gave the first CP-ABKS scheme, which enables data owner to delegate search capabilities to data users by enforcing access control over encrypted cloud data. However, this scheme just supports single keyword search in single-owner scenarios and hence affects user's search experience. After that, Sun *et al.* [14] presented an authorized multi-keyword search scheme in a challenging multi-owner scenario [42] to achieve fine-grained owner-enforced search privileges. Qiu *et al.* [43] gave a more secure CP-ABKS scheme to guarantee access policy privacy and resist off-line keyword guessing attack. However, these CP-ABKS schemes only support single attribute-authority, which may incur single-point performance bottleneck. This is because the single AA (also referred as CA) in these schemes must issue both user certificate verification and secret key generation. Furthermore, existing CP-ABE schemes [16], [17] supporting multi-authority cannot be directly extended to SE scheme to mitigate the discussed concerns since each authority separately keeps disjoint attribute subsets.

In practice, a preferred CP-ABKS scheme should be constructed without sacrificing efficiency while supporting both

TABLE 1: Functionalities in various schemes: A comparative summary

Schemes	F1	F2	F3	F4	F5	F6
[4]		✓	✓	✓		YES
[5]		✓	✓	✓		NO
[16]	✓		✓			NO
[17]	✓		✓	✓		NO
[18]	✓		✓		✓	YES
[37]	✓		✓	✓		NO
[41]		✓	✓			YES
[14]		✓	✓			NO
[13]		✓	✓			NO
[43]		✓	✓			NO
MABKS	✓	✓	✓	✓	✓	YES

Notes. F1: Multi-authority; F2: Keyword-based retrieval; F3: Fine-grained access control; F4: Attribute update; F5: Malicious AA tracing; F6: High efficiency.

fine-grained access control and keyword search. The storage and computation costs of existing CP-ABKS schemes linearly increase with the number of system attributes instead of user attributes; hence, such schemes are not practical for deployment on resource-constrained devices. This is one of the gaps we aim to solve in this paper. Specifically, our proposed MABKS system has constant trapdoor size and ciphertexts retrieval overhead based on RAAC scheme [18]. The extended MABKS system is also designed to support malicious AAs tracing and attribute update. Compared with existing schemes, the extended MABKS system has several advantages (e.g., multi-authority, keyword-based retrieval, fine-grained access control, attribute update, malicious AA tracing, high efficiency[1]), as summarized in TABLE 1.

3 Preliminaries

In this section, we review some cryptographic background associated with the MABKS system. Assume that G, G_T are two multiplicative cyclic groups of prime order p and g is the generator of G, the bilinear map $e : G \times G \to G_T$ has several properties: (1) Bilinearlity. $e(g^a, g^b) = e(g, g)^{ab}$, $\forall a, b \in \mathcal{Z}_p$; (2) Non-degeneracy: $e(g, g) \neq 1$; (3) Computability. There exists an efficient algorithm to compute $e(g, g)$. The symbol $x \in X$ is defined as choosing an element x uniformly at random from the set X, and $[1, y]$ represents an integer set $\{1, 2, ..., y\}$, where y is an integer. The symbols used are shown in TABLE 2.

3.1 Access Structure

Let $\mathcal{P} = \{P_1, P_2, \cdots, P_n\}$ be a set of parties (or attributes). The collection $\mathbb{A} \subseteq 2^{\{P_1, P_2, \cdots, P_n\}}$ is monotonic when the following condition holds: for two arbitrary party (or attribute) sets B, C, $C \in \mathbb{A}$ holds on condition that $B \in \mathbb{A}$, $B \subseteq C$. An monotonic access structure [33] is a collection $\mathbb{A}$ with non-empty subsets of $\mathcal{P}$, (e.g., $\mathbb{A} \subseteq 2^{\{P_1, P_2, \cdots, P_n\}} \backslash \emptyset$). The sets in $\mathbb{A}$ are called authorized entities; otherwise, the ones are called unauthorized entities. It should be noted that each party is described by an attribute in this paper, and all authorized attribute sets belong to $\mathbb{A}$. Besides, $\mathbb{A}$ represents the monotone access structure.

1. The data users' storage and computation costs of trapdoor generation or ciphertexts search (or decryption) are approximately constant, or outsourced to other entities.

TABLE 2: Symbol definitions in preliminaries

Notations	Descriptions	Notations	Descriptions
G, G_T	Cyclic groups	g	G's generator
$e : G \times G \to G_T$	Bilinear map	$[1, y]$	Integer set
$\mathcal{P} = \{P_1, P_2, \cdots, P_n\}$	Parties (attributes)	$\mathbb{A}$	Access structure
B, C	Arbitrary party set	$\mathcal{M}$	LSSS matrix
$\rho(\cdot)$	Mapping function	s	Shared secret
$\mathbf{v} = \{s, r_2, \cdots, r_n\}^\top$	Column vector	$\mathcal{S}$	User attributes
$I = \{i : \rho(i) \in \mathcal{S}\}$	Row subset in $\mathcal{M}$	$\{\omega_i\}$	Constant set
$\lambda_i = \mathcal{M}_i \cdot \mathbf{v}$	i-th share of s	$\mathcal{M}_i$	i-th row in $\mathcal{M}$

Notes: $\rho(i)$ maps the i-th row of $\mathcal{M}$ to an attribute; I denotes the row subset with corresponding attributes in $\mathcal{S}$.

3.2 Linear Secret Sharing Scheme

To reconstruct the secret key, we will exploit the linear secret sharing scheme defined in Waters's scheme [44], which is demonstrated as follows.

Definition 1 (*Linear Secret-Sharing Scheme (LSSS)*)**.** If the following conditions both hold, the LSSS over $\mathcal{P}$ is linear.

- The share for each party (or attribute) forms a vector over $\mathcal{Z}_p$;
- Let $\mathcal{M}$ be the LSSS matrix (l rows and n columns) which can be used to describe the access structure $\mathbb{A}$, and its i-th row is defined as $\mathcal{M}_i (i \in [1, l])$. The mapping function $\rho(\cdot)$ maps each row $\mathcal{M}_i$ to a certain attribute $\rho(i)$. Given a column vector $\mathbf{v} = \{s, r_2, \cdots, r_n\}^\top$, the symbol $\mathcal{M} \cdot \mathbf{v}$ represents the l shares of s in LSSS, and $\mathcal{M}_i \cdot \mathbf{v}$ is the share λ_i belonging to attribute $\rho(i)$, where $s \in \mathcal{Z}_p$ denotes the secret to be shared, and $r_2, \cdots, r_n \in \mathcal{Z}_p$ are random elements.

The linear secret-sharing scheme enjoys the property of *linear reconstruction*. Let $\mathbb{A}$ represent the access structure, $\mathcal{S}$ denote an attribute set that satisfies $\mathbb{A}$, and the symbol $I \subseteq \{1, 2, \cdots, l\}$ represent the row set that has corresponding attributes in $\mathcal{S}$, namely $I = \{i : \rho(i) \in \mathcal{S}\}$, then there exists a tuple of constants $\{\omega_i \in \mathcal{Z}_p\}_{i \in I}$ such that $\sum_{i \in I} \omega_i \lambda_i = s$. Note that $\lambda_i = \mathcal{M}_i \cdot \mathbf{v}$ denotes each row's share of secret s according to LSSS, and $\{\omega_i\}$ can be found in polynomial time in the size of $\mathcal{M}$.

3.3 Security Assumptions

To proof the security of the MABKS system, we present some necessary security assumptions (e.g., decisional q-parallel Bilinear Diffie-Hellman Exponent (BDHE) assumption [44], Decisional Bilinear Diffie-Hellman (DBDH) assumption [45]).

Definition 2 (*Decisional q-parallel Bilinear Diffie-Hellman Exponent (BDHE)* assumption)**.** Let (G, G_T, e, g) be the bilinear map parameters, $a, z, b_1, \cdots, b_q$ be random elements. Even though an adversary has the tuple ϕ shown with Eq. 1, it is still difficult for the adversary to distinguish

$e(g,g)^{a^{q+1}z}$ from a random element R in group G_T, where $1 \le j, i \le q, i \ne j$.

$$\phi = \left\{ \begin{array}{l} g, g^z, g^a, \cdots, g^{a^q}, g^{a^{q+2}}, \cdots, g^{a^{2q}}, \\ g^{z\cdot b_j}, g^{a/b_j}, \cdots, g^{a^q/b_j}, g^{a^{q+2}/b_j}, \cdots, \\ g^{a^{2q}/b_j}, g^{a\cdot z\cdot b_i/b_j}, \cdots, g^{a^q\cdot z\cdot b_i/b_j} \end{array} \right\}. \quad (1)$$

If the following inequation Eq. 2 holds, the algorithm has an advantage ϵ in solving the decisional q-parallel BDHE problem in group G. Hence, the decisional q-parallel BDHE assumption holds on condition that there exists on polynomial time $\mathcal{B}$ that can break the decisional q-parallel BDHE problem with a non-negligible advantage ϵ.

$$|Pr[\mathcal{B}(\phi, e(g,g)^{a^{q+1}z}) = 0] - Pr[\mathcal{B}(\phi, R) = 0]| \ge \epsilon. \quad (2)$$

Definition 3 (*Decisional Bilinear Diffie-Hellman (DBDH) assumption*)**.** Let (G, G_T, e, g) be the bilinear map parameters, the challenger $\mathcal{C}$ first randomly selects three elements $a, b, c \in \mathcal{Z}_p$, and then flips a fair binary coin $\kappa \in \{0,1\}$. If $\kappa = 1$, $\mathcal{C}$ outputs a tuple $\{g, g^a, g^b, g^c, e(g,g)^{abc}\}$; otherwise, it returns the tuple $\{g, g^a, g^b, g^c, e(g,g)^z\}$. The goal of adversary $\mathcal{A}$ is to output a guess bit κ' of κ. If the following inequation Eq. 3 holds, then $\mathcal{A}$ can break the DBDH problem with an advantage at least ϵ, note that the probability is over random elements a, b, c, z and random bits consumed by $\mathcal{A}$.

$$\left| \begin{array}{l} Pr[\mathcal{A}(g, g^a, g^b, g^c, e(g,g)^{abc}) = 1] \\ - Pr[\mathcal{A}(g, g^a, g^b, g^c, e(g,g)^z) = 1] \end{array} \right| \ge \epsilon. \quad (3)$$

If there exists no adversary $\mathcal{A}$ that can break the DBDH problem with an advantage at least ϵ, then we can say that the DBDH assumption holds.

4 Problem Formulation

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

4.1 System & Threat Models

We consider a cloud storage system in the cloud computing environment, which involves with five entities in Fig. 3, namely Central Authority (CA), multiple Attribute Authorities (AAs), Data Owner (DO), Cloud Service Provider (CSP) and Data User (DU). It is worth noticing that DUs are usually resource-limited entities (i.e., mobile devices, wearable devices, sensor nodes, etc.). However, the CA and multiple AAs have sufficient computation and storage capabilities to accomplish the assigned tasks.

In the cloud storage system, the DO collects files and generates the ciphertexts including indexes and file encryption key ciphertexts (Step ①). To relieve the computation and storage burden, the DO outsources ciphertexts to CSP which has capacities to issue huge amounts of data storage and search operations. Before conducting search queries, the DU must issue the secret key generation, which is cooperatively conducted by CA and his chosen AA (Step ②). Specifically, the DU first obtains his certificate by submitting his identity to CA, then sends his certificate to the selected AA. The AA needs to perform the user certificate verification and send the intermediate secret key to CA. The CA returns the final secret key to the DU. After that, the DU first generates

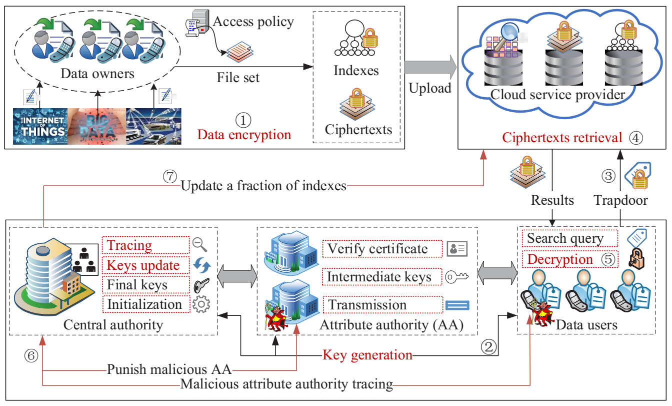


Fig. 3: The system model of MABKS scheme.

the trapdoor according to his specified keyword, then sends the trapdoor (or search token) along with his attributes to CSP (Step ③). In the step ④, the CSP first checks whether both the attributes and trapdoor satisfy the access policy and indexes, respectively, then returns the relevant search results to DU if above two conditions hold. When gaining the search results, the DU needs to obtain the corresponding file decryption keys before he can decrypt these encrypted search results (Step ⑤). When the suspect AA mistakenly or intentionally outputs the incorrect intermediate secret key for a suspect DU, the CA can trace the malicious AA by interacting with the suspect DU (Step ⑥). If the attributes of a certain DU have been updated, the CA will generate the transformation keys to update the DU's final secret key and indexes so that the malicious DU cannot access the sensitive information by using his old or outdated secret key (Step ⑦). The role of each entity is presented as follows:

- *Central authority*. The CA can not only generate final secret keys for DUs but also trace the malicious AAs which generate incorrect intermediate secret keys for DUs in the extended MABKS system.
- *Attribute authorities*. Each AA, which has adequate storage and computation capabilities, can separately perform user certificate validation according to DU's claimed attributes and generate the corresponding intermediate secret key on behalf of the CA. Note that the goal of introducing multiple AAs is to relieve CA from burdensome tasks of certificate verification and key generation, and further reduce the possibility of single-point performance bottleneck.
- *Cloud service provider*. The CSP which has numerous store space and powerful computation capability can provide data storage and information retrieval services for DOs and DUs, respectively.
- *Data owner*. The DO collects and outsources his encrypted cloud data to CSP in order to share his data with multiple DUs, and significantly reduce local storage and computation burden.
- *Data user*. The DU can issue ciphertexts retrieval requirements based on interested keywords before being verified his legitimacy by a certain AA and gaining the final secret key from CA.

Additionally, the CSP is an honest-but-curious entity

MABKS system definition

The overview of the MABKS system is presented as follows:

Setup(1^k). Given a security parameter k, the CA conducts the algorithm to output the public key PK and master key MSK.

KeyGen(PK, S, MSK). Take the attribute set S of a certain DU as input, the selected AA (i.e., $AA_j(j \in [1, m])$) and CA separately generate the intermediate secret key $SK_{u,0}$ and final secret key $SK_{u,1}$. Notice that the CA also outputs public/secret key pairs (PK_j, SK_j) for selected AA_j, and the AA_j needs to issue user certificate validation on behalf of CA.

Enc$(PK, \mathcal{F}, \mathcal{W}, \mathbb{A})$. Given the file set $\mathcal{F}$, keyword set $\mathcal{W}$ and access structure $\mathbb{A}$, the DO first outputs the file ciphertexts $\mathcal{C}^*$, file encryption key ciphertexts CT and encrypted indexes Ind, and then returns the tuple $\{\mathcal{C}^*, CT, Ind, \mathbb{A}\}$ to CSP.

Trap$(PK, w', S, SK_{u,1})$. Given the queried keyword w', the DU generates the trapdoor $T_{w'}$ according to his attributes S and secret key $SK_{u,1}$ and submits $(T_{w'}, S)$ to CSP.

Search$(PK, \mathcal{C}^*, Ind, CT, T_{w'}, S, \mathbb{A})$: After gaining submitted trapdoor $T_{w'}$ along with DU's attributes S, the CSP checks whether S (resp., $T_{w'}$) satisfies $\mathbb{A}$ (resp., Ind). If above conditions both hold, the CSP returns the relevant search results $\mathcal{C}_{w'}$ and corresponding encryption key ciphertexts CT^*.

Dec$(PK, \mathcal{C}_{w'}, CT^*, SK_{u,1})$; Given the tuple $(\mathcal{C}_{w'}, CT^*)$, the DU needs to obtain the relevant file encryption keys which are used to decrypt $\mathcal{C}_{w'}$ by using his secret key $SK_{u,1}$ and CT^*.

Fig. 4: Overview of the MABKS system

which honestly abides by established protocols but may try to spy out some sensitive information. The CA, which should be real-time online to generate the final secret keys for DUs, is assumed to be fully trusted. The AAs provided by third-parties may perform maloperations incurred by carelessness or malicious behaviors. The malicious DUs may collude with each other or even compromise any AA to obtain unauthorized accesses beyond their access privileges. The DOs are fully credible.

4.2 Definition of MABKS System

The MABKS system is a tuple of algorithms involving **Setup**, **KeyGen**, **Enc**, **Trap**, **Search** and **Dec**. The overview of the MABKS system is presented in Fig. 4

4.3 Security Model

To protect files confidentiality, sensitive information cannot be accessed by unauthorized DUs or CSP. Thus, the MABKS system should be secure in the selective-matrix and selective-attribute models. Besides, the MABKS system should resist user collusion attacks.

In the selective-matrix model [44], [46], the security model of the MABKS system allows a certain adversary $\mathcal{A}$ to query for the secret key which cannot be utilized to decrypt the challenging ciphertexts. In the MABKS system, ciphertexts are encrypted with an access structure $\mathbb{A}$ and secret keys are created with attribute sets, where $\mathbb{A}$ can be described by LSSS. Next, we give the following selective-matrix model between challenger $\mathcal{C}$ and $\mathcal{A}$, where $\mathcal{A}$ chooses challenging ciphertexts encrypted by $\mathbb{A}^*$ and issues the secret key generation for attribute set S such that S does not satisfy $\mathbb{A}^*$.

- *Setup*: $\mathcal{C}$ runs the **Setup** to create the system public parameters PK and sends PK to $\mathcal{A}$.
- *Phase 1*: A repeatedly issues secret key queries for attribute sets $S_1, S_2, \cdots, S_{q_1}$.
- *Challenge*: First, A submits two records R_0, R_1 with equal length and a challenging access structure $\mathbb{A}^*$, but one restriction is that all attribute sets $S_1, S_2, \cdots, S_{q_1}$ do not match with $\mathbb{A}^*$. Then, $\mathcal{C}$ selects a random bit $\kappa \in \{0, 1\}$ and encrypts the record R_κ with $\mathbb{A}^*$. Finally, $\mathcal{C}$ sends the challenging ciphertext C_κ to $\mathcal{A}$.
- *Phase 2*: $\mathcal{A}$ repeats *Phase 1* for attribute sets $S_{q_1+1}, \cdots, S_q$, but notice that none of these attribute sets satisfy the aforementioned $\mathbb{A}^*$.
- *Guess*: $\mathcal{A}$ outputs a guess bit $\kappa' \in \{0, 1\}$. If $\kappa' = \kappa$, $\mathcal{A}$ wins this game; otherwise, it fails.

It is worth noticing that this above security model can be extended to deal with chosen-ciphertext attacks by allowing $\mathcal{A}$ to conduct decryption queries in *Phase 1* and *Phase 2*.

Definition 4. The MABKS system is secure if there exist probabilistic polynomial time (PPT) adversaries that can have at most a negligible advantage $Adv_{\mathcal{A}}(1^k) = |Pr[\kappa' = \kappa] - \frac{1}{2}| < \epsilon$ in breaking the above security game in selective-matrix model, where k represents the security parameter.

As for the selective-attribute model, in which $\mathcal{A}$ must submit a challenging attribute set before gaining the public system parameters and adaptively issuing secret key and trapdoor generation queries, the MABKS system can achieve stronger security when compared with previous ABE schemes in selective-set model [40]. Next, we show this kind of security game between $\mathcal{A}$ and $\mathcal{C}$ as follows. This security game is demonstrated in **Supplemental Material A**.

Definition 5. The MABKS system is secure if there does not exist an adversary that can break the above security game with a non-negligible advantage $Adv_{\mathcal{A}}(1^k) = |Pr[\kappa' = \kappa] - \frac{1}{2}| \geq \epsilon$ in selective-attribute model.

5 The Proposed MABKS System

For ease of understanding, we first show some notation descriptions used in the MABKS system in TABLE 3 before introducing its concrete construction. Different from the traditional CP-ABKS schemes that can achieve fine-grained access control and keyword-based ciphertexts retrieval at the same time by simply combining CP-ABE and SE techniques, the MABKS system can gain the file-level fine-grained key by embedding the secret $s \in \mathcal{Z}_p$ chosen in file key encryption process into the corresponding index building process. However, the traditional single-authority CP-ABKS schemes or multi-authority CP-ABE schemes still suffer from the single-point performance bottleneck. Hence, the MABKS system first employs the heterogeneous architecture including a trusted CA and multiple AAs, note that each AA selected by a certain DU can conduct the time-consuming certificate verification and generate intermediate secret keys for data users on behalf of CA, thereby eliminating the CA's computation burden significantly and then avoiding the single-point performance bottleneck. Besides, the encryption and decryption costs in traditional CP-ABKS schemes increase with the complexities of access policies linearly, which impose high computation burden on DO and DUs, respectively. Then, the modified MABKS will separately use the online/offline ABE mechanism and outsourced decryption mechanism to solve these two flaws. However, in actual deployments, the AAs provided by third-parties may execute malicious operations (i.e., returning incorrect intermediate secret keys for data users). To overcome this shortcoming, the extended MABKS system redefines the random elements $\alpha, \beta \in \mathcal{Z}_p$ chosen in the MABKS system as $\alpha = \mathcal{H}(ID_u \| Ts \| 0)$, $\beta = \mathcal{H}(ID_u \| Ts \| 1)$, where Ts denotes the timestamp, then computes $k^*_{j,\tau,u,0}$ to confirm the secret key ownership, and recovers the public key to trace malicious AA. In dynamic scenarios, the role of each DU may change dynamically. The DUs will use the original or outdated secret keys to access original documents/files. To prevent this kind of unauthorized accesses, the extended MABKS system requires the CA to update the master key MSK, public parameters PK and output two transformation keys, note that these two transformation keys allow the DUs and CSP to update the secret key component $SK_{u,1}$ and a fraction of indexes associated with updated attributes, respectively.

TABLE 3: Notation descriptions in MABKS system

Notations	Descriptions
$\mathcal{U} = \{Att_1, \cdots, Att_U\}$	System attribute set
$AA = \{AA_1, \cdots, AA_m\}$	Attribute authority set
(PK, MSK)	Public parameters/master key
(PK_j, SK_j)	AA_j's public/secret key pair
$(Cert_j, Cert_u)$	AA_j's/DU's ceetificates
$(SK_{u,0}, SK_{u,1})$	DU's intermediate/final key
$(\mathcal{F} = \{f\}, \mathcal{W} = \{w\})$	File/keyword set
$\{sk_f, CT_f\}$	File key plaintext/ciphertext
$\mathcal{C}^* = \{Enc_{sk_f}(f)\}$	File ciphertexts for $\mathcal{F}$
$Ind = \{I_w\}$	Encrypted indexes for W
$T_{w'} = (T_0, T_1)$	Trapdoor for queried keyword w'
$(\mathcal{C}_{w'}, CT^*)$	Returned file/file key ciphertexts

5.1 Construction of MABKS System

In this part, we demonstrate the concrete construction of the basic MABKS system. It is worth noticing that the construction of the MABKS system includes six phases, namely system initialization (**Setup**), secret key generation (**KeyGen**), ciphertexts generation (**Enc**), trapdoor generation (**Trap**), ciphertexts retrieval (**Search**) and ciphertexts decryption (**Dec**). The high-level description of MABKS algorithms is shown in Fig. 5.

In **Setup**, the fully-trusted CA creates the global public parameters PK and the master key MSK, where MSK is kept by itself. In **KeyGen**, the DU's secret key is cooperatively generated by CA and his selected AA. Each AA which owns the whole attributes needs to conduct user certificate

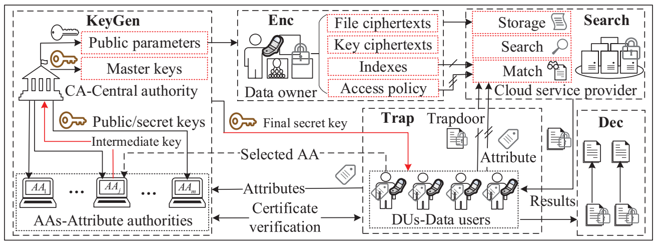


Fig. 5: The high-level description of MABKS algorithms.

verification independently and generates the intermediate secret key, thereby reducing the user legitimacy verification burden on CA. When gaining the intermediate secret key returned by the chosen AA, the CA outputs the final secret key for each DU. In **Enc**, the DO first encrypts each file with a symmetric encryption key, then protects the symmetric keys and builds the indexes for the whole file set with the specified access structure. In **Trap**, the DU first needs to encrypt his plaintext search query as trapdoor by using his final secret key so that the sensitive information is not leaked to the honest-but-curious CSP, then sends his attributes and trapdoors to CSP. In **Search**, the CSP can return the DU's interested search results iff both attributes and trapdoor match with the access policy and indexes, respectively. In **Dec**, the DU needs to obtain the corresponding symmetric keys for search results before he/she can decrypt them. The concrete algorithm descriptions of MABKS system is shown in Fig. 6. It is worth noticing that the detailed algorithms for key generation process, ciphertexts generation process and ciphertexts retrieval process are shown in **Supplemental Material B**.

Remarks. In the MABKS system, each AA can separately issue user certificate authentication and generate intermediate secret keys for DUs on behalf of CA, then the problems of single-point performance bottleneck and high computation burden on CA can be solved. Furthermore, the MABKS system has constant-size trapdoor and ciphertexts retrieval overhead, which applies to resource-limited device deployment. Last but not the least, the MABKS system can achieve both keyword-based ciphertexts retrieval and fine-grained access control, and authorized DUs can gain and decrypt interested search results iff their attribute sets (resp., trapdoors) satisfy established access structures (resp., indexes). However, the MABKS scheme still incurs high computation burden on DO and DUs in **Enc** and **Dec** algorithms, respectively. To solve these challenging issues, we demonstrate the idea that how to achieve online/offline encryption and outsourced decryption mechanisms by using the online/offline ABE [48] and outsourcing the decryption of ABE ciphertexts [49], [50], respectively. Note that we just show the modified algorithms in our MABKS system, which are shown in Fig. 7. However, some other challenging issues shown in the following sections still need to be addressed.

5.2 Extension to Support AA Tracing

To prevent the malicious AA from generating the illegal intermediate secret key, an efficient tracing mechanism must be furnished in MABKS to guarantee that the secret key components really belong to a suspected DU. It is worth

The algorithms in MABKS system

The concrete construction of the MABKS system is presented as follows:

Setup(1^k): Let k be the security parameter, the CA first creates the global bilinear parameters $\mathcal{GP} = (G, G_T, e, g)$ and selects other two generators g_0, g_1 of group G, then randomly chooses elements $a_0, a_1, b_0, b_1 \in \mathcal{Z}_p$ and anti-collision hash function $\mathcal{H} : \{0,1\}^* \to \mathcal{Z}_p$ before computing $\theta = e(g,g)^{a_0}$, $A = g^{a_1}$, $A_0 = g^{a_0}$, $B_0 = g^{b_0}$. Given the system attributes $\mathcal{U} = \{Att_1, \cdots, Att_U\}$, the CA picks a random element $\nu_i \in \mathcal{Z}_p$ for each attribute $Att_i \in \mathcal{U}(i \in [1, U])$ and computes $H_i = B_0^{\nu_i}$. Finally, the CA sets the public parameters PK and master key MSK with Eq. 4.

$$\begin{aligned} &PK = \{\mathcal{GP}, \mathcal{H}, g_0, g_1, \theta, A, A_0, B_0, H_1, \cdots, H_U\};\\ &MSK = \{a_0, a_1, b_0, b_1, \nu_1, \cdots, \nu_U\}. \end{aligned} \tag{4}$$

KeyGen(PK, S, MSK): The key generation process of a certain DU is involved with both AA and CA, where CA deals with registration requests of multiple attribute authorities $AA = \{AA_1, AA_2, \cdots, AA_m\}$ and different DUs by using its public/secret key pair [15], which is beyond the scope of our discussion. For each attribute authority $AA_j(j \in [1, m])$ with identity ID_j, the CA picks a random element $k_j \in \mathcal{Z}_p$ and outputs AA_j's public/secret key pair (PK_j, SK_j) along with a certificate $Cert_j$, where $PK_j = g^{k_j}$, $SK_j = k_j$. In addition, a certain DU with identity ID_u first gets the certificate $Cert_u$ from CA. Then, the DU conducts the key generation query with randomly selecting $AA_j(j \in [1, m])$. Finally, the selected AA_j verifies the validity of DU's certificate[a] $Cert_u$. Assume that the AA_j honestly executes established protocols, the AA_j will abort this process if the DU does not have the legal attribute set that he claims to; otherwise, it will return the intermediate secret key $SK_{u,0}$ and send $SK_{u,0}$ to CA with the following steps.

- For each attribute $\tau \in S$, the AA_j selects two random elements $\alpha, \beta \in \mathcal{Z}_p$ and computes $k'_{j,u,\tau,0} = H_\tau^{k_j \alpha}$, $k''_{j,u,\tau,0} = H_\tau^{\beta}$.
- The AA_j returns $SK_{u,0} = \{k'_{j,u,\tau,0}, k''_{j,u,\tau,0}\}_{\tau \in S}$ and the related tuple (ID_j, ID_u, S) to CA which is responsible for generating the final secret key $SK_{u,1}$.

After gaining $(SK_{u,0}, ID_j)$, the CA first looks up AA_j's corresponding public key PK_j, then selects a random element $\gamma_u \in \mathcal{Z}_p$ before generating DU's final secret key $SK_{u,1}$ with Eq. 5, finally securely sends $SK_{u,1} = (K_0, K_1, K_2, K_3, \{k'_{j,u,\tau,1}, k''_{j,u,\tau,1}, k'''_{j,u,\tau,1}\}_{\tau \in S})$ to a certain DU via AA_j.

$$\begin{aligned} &K_0 = g^{\gamma_u}, K_1 = g_0^{a_0} g_1^{\gamma_u}, K_2 = g^{a_0} PK_j^{a_1 b_0 \alpha} g^{a_1 a_0 \beta};\\ &K_3 = PK_j^{b_0 \alpha} g^{a_0 \beta}, k'_{j,u,\tau,1} = (k'_{j,u,\tau,0})^{b_0} g^{b_1(\alpha+\beta)};\\ &k''_{j,u,\tau,1} = (k''_{j,u,\tau,0})^{a_0} g^{-b_1(\alpha+\beta)}, k'''_{j,u,\tau,1} = H_\tau^{\gamma_u}. \end{aligned} \tag{5}$$

Enc$(PK, \mathcal{F}, \mathcal{W}, \mathbb{A})$: Take as input the file set $\mathcal{F} = \{f\}$ and the keyword dictionary $\mathcal{W} = \{w\}$, the DO first encrypts each file $f \in \mathcal{F}$ with corresponding symmetric key sk_f, then protects sk_f with specified access structure $\mathbb{A}$ described by matrix $\mathcal{M}_{l \times n}$. Notice that the encrypted files are defined as $\mathcal{C}^* = \{Enc_{sk_f}(f)\}$, and the access structure embedded in keywords is similar to that of files.

- The DO chooses a column vector $\mathbf{v} = \{s, r_2, \cdots, r_n\}$ and computes $\lambda_i = \mathcal{M}_i \mathbf{v}$, where $\mathcal{M}_i(i \in [1, l])$ represents the i-th row of $\mathcal{M}$. After that, the DO picks random elements $\pi_1, \cdots, \pi_l \in \mathcal{Z}_p$ and outputs the file ciphertext $CT_f = (C'_f, C'', \{C_{i,1}, C_{i,2}\})$, where $C'_f = sk_f \cdot e(g,g)^{a_0 s}$, $C'' = g^s$, $C_{i,1} = (g^{a_1})^{\lambda_i} H_{\rho(i)}^{-\pi_i}$, $C_{i,2} = g^{\pi_i}$.
- The DO extracts keywords from each file $f \in \mathcal{F}$ according to keywords $\mathcal{W}$ before building index I_w for each keyword $w \in \mathcal{W}$. For each attribute $\rho(i)(i \in [1, l])$, the DO computes $I_0 = g_1^s$, $I_{1,i} = \varpi_i^s$, $I_2 = e(A_0, g_0)^s$, $I_3 = B_0^{s/\mathcal{H}(w)}$, where $\varpi_i = H_{\rho(i)}^{1/\mathcal{H}(w)}$. Finally, the DO sets $I_w = (I_0, I_2, I_3, \{I_{1,i}\})$. Note that the component I_3 is mainly used to update indexes in Section 5.3.
- The DO sends the final ciphertexts $(\mathcal{C}^*, CT, Ind)$ and access structure $\mathbb{A}$ to CSP, where $CT = \{CT_f\}$, $Ind = \{I_w\}$.

Trap$(PK, w', S, SK_{u,1})$: When a certain DU intends to search encrypted files including queried keyword w', the DU needs to generate a trapdoor $T_{w'}$ and sends it to CSP. The DU first chooses a random element $\gamma'_u \in \mathcal{Z}_p$ and computes $T_0 = K_0 g^{\gamma'_u}$, $\varphi = K_1 g_1^{\gamma'_u}$, $\varphi_\tau = k'''_{j,u,\tau,1} H_\tau^{\gamma'_u}$, then computes $T_1 = \varphi \prod_{\tau \in S} \varphi'_\tau$, where $\varphi'_\tau = \varphi_\tau^{1/\mathcal{H}(w')}$. Finally, the DU sets the search token as $T_{w'} = (T_0, T_1)$ and returns $T_{w'}$ to CSP.

Search$(PK, \mathcal{C}^*, Ind, CT, T_{w'}, S, \mathbb{A})$: After receiving the DU's submitted trapdoor (or search token) $T_{w'}$ and attribute set S, the CSP first checks whether S satisfies the access structure $\mathbb{A}$. If not, the CSP aborts this search operation; otherwise, it continues to check whether $T_{w'}$ matches with indexes Ind with Eq. 6, where $i \in [1, l]$.

$$I_2 \cdot e(T_0, I_0 \cdot \prod_{\rho(i) \in S} I_{1,i}) = e(C'', T_1). \tag{6}$$

If Eq. 6 holds, the CSP returns the encrypted files $\mathcal{C}_{w'}$ containing the keyword w' and corresponding encryption key ciphertexts CT^*; otherwise, it returns $\perp$.

Dec$(PK, \mathcal{C}_{w'}, CT^*, SK_{u,1})$: After gaining the relevant search results $\mathcal{C}_{w'}$, the DU can decrypt them with his final secret key. Suppose that the submitted attributes match with the access structure embedded in file ciphertexts and let $I \subseteq \{1, 2, \cdots, l\}$ be defined as $I = \{i : \rho(i) \in S\}$, there must exist a tuple of constants $\{\omega_i\}$ such that $\sum_{i \in I} \omega_i \lambda_i = s$, where $\lambda_i = \mathcal{M}_i \mathbf{v}$. Before gaining file plaintexts, the DU needs to obtain the secret value s and further have symmetric keys for $\mathcal{C}_{w'}$ with the following steps:

- For each attribute $\tau \in S$, the DU first computes $\psi_\tau = k'_{j,u,\tau,1} \cdot k''_{j,u,\tau,1} = H_\tau^{(k_j b_0 \alpha + a_0 \beta)}$.
- Given the constant set $\{\omega_i\}$, the DU then further gains $\xi = e(g,g)^{a_0 s}$ with Eq. 7.

$$\xi = \frac{e(C'', K_2)}{\prod_{i \in I}(e(C_{i,1}, K_3) \cdot e(C_{i,2}, \psi_{\rho(i)}))^{\omega_i}}. \tag{7}$$

- The DU computes C'_f/ξ to get the symmetric key sk_f for file f. Finally, the DU can decrypt $\mathcal{C}_{w'}$ with corresponding symmetric keys.

a. The CA distributes a certificate, which is used to blind each DU's attributes to public key and auxiliary information, based on traditional Public Key Infrastructure (PKI) architectures [47], and the selected AA_j can check the validity of each DU's certificate as both AA_j and DUs trust CA.

Fig. 6: Concrete construction of MABKS system

Modified algorithms to support online/offline encryption and outsourced decryption

The modified algorithms[a] are shown as follows:

KeyGen(PK, S, MSK): Different from the original **KeyGen** algorithm, the CA also selects a random element $Tr \in \mathcal{Z}_p$ in the final secret key generation process. The CA computes the transformation key $SK'_{Tr} = (K'_2, K'_3, \{TK'_{j,s,\tau,1}, TK''_{j,s,\tau,1}\})$, where $K'_2 = K_2^{1/Tr}$, $K'_3 = K_3^{1/Tr}$, $TK'_{j,s,\tau,1} = (k'_{j,u,\tau,1})^{1/Tr}$, $TK''_{j,s,\tau,1} = (k''_{j,u,\tau,1})^{1/Tr}$, then returns the DU's final secret key $SK'_{u,1} = (Tr, K_0, K_1, \{k'''_{j,u,\tau,1}\})$ and CSP's transform key SK'_{Tr}, where $\tau \in S$. Note that the MABKS system does not need to modify **Trap** algorithm as the secret key components $(K_0, K_1, \{k'''_{j,u,\tau,1}\})$ used to generate trapdoor are still unchanged.

To avoid the heavy computation burden of **Enc** on resource-limited DO, we extend the online-offline ABE technique [48] into the MABKS system. The online/offline **Enc** consists of two phases, namely offline phase in which DO can conduct the vast majority of ciphertext generation tasks prior to knowing the specific files and access policies, and online phase in which DO can rapidly output the final ciphertexts. Note that the computationally intensive tasks in an offline phase can also be conducted by the client-server, which further the computation overhead of resource-constrained DO. The original **Enc** includes **Offline-Enc** and **Online-Enc**.

- **Offline-Enc**(PK, Ψ): Let the maximum bound of rows in LSSS access structure embedded in each file key ciphertext be Ψ, then DO (i.e., mobile devices, sensor nodes, etc.) selects random element $s, \lambda'_i, \pi_i \in \mathcal{Z}_p$ and computes $C'_f = e(g,g)^{a_0 s}$, C'', $C'_{i,1} = g^{a_1 \lambda'_i}$, $C''_{i,1} = H_{\rho(i)}^{-\pi_i}$, $C_{i,2}$, I_0, I_2 in offline phase when he/she is plugged into a power source. Then, the DO outputs the intermediate file key ciphertext $CT^* = (C'_f, C'', \lambda'_i, \{C'_{i,1}, C''_{i,1}, C_{i,2}\}_{i \in [1,\Psi]})$ and intermediate index $I^* = (I_0, I_2)$.
- **Online-Enc**$(PK, CT^*, I^*, \mathcal{F}, \mathcal{W}, \mathbb{A})$: The generation process of file ciphertexts $\mathcal{C}^* = \{Enc_{sk_f}(f)\}$ is the same as that of original **Enc**. Then, the DO chooses a column vector $\mathbf{v} = \{s, r_2, \cdots, r_n\}$ and computes $C^*_i = \lambda_i - \lambda'_i$, $C''_f = sk_f \cdot C'_f$. Finally, to build index I_w each keyword in f, the DO further computes $I_{i,1} = \varpi_i^s$, $I_3 = B_0^{s/\mathcal{H}(w)}$, where $\varpi_i = H_{\rho(i)}^{1/\mathcal{H}(w)}$, $i \in [1, l]$. The final ciphertexts are defined as $(\mathcal{C}^*, CT = \{CT_f\}, Ind = \{I_w\})$, where $CT_f = (C''_f, C'', \{C^*_i, C'_{i,1}, C''_{i,1}, C_{i,2}\})$, $I_w = (I_0, I_2, I_3, I_{1,i})$.

Search$(PK, \mathcal{C}^*, Ind, CT, T_{w'}, S, \mathbb{A})$: Apart from verifying the correctness of Eq. 6, the CSP also executes ciphertexts transformation so that the burdensome ciphertexts decryption burden exposed on resource-limited DU can be relieved. For each attribute $\tau \in S$, the CSP first computes $\psi'_\tau = TK'_{j,s,\tau,1} \cdot TK''_{j,s,\tau,1} = H_\tau^{(k_j b_0 \alpha + a_0 \beta)/Tr}$, then he computes $\xi' = e(g,g)^{a_0 s/Tr}$ with Eq. 8. Finally, the CSP returns the search results $\mathcal{C}_{w'}$, corresponding file key ciphertexts $CT^* = \{C''_f\}$ as well as transformed ciphertexts ξ' to CSP.

$$\xi' = \frac{e(C'', K'_2)}{\prod_{i \in I}(e(C'_{i,1} A^{C^*_i} C''_{i,1}, K'_3) \cdot e(C_{i,2}, \psi_{\rho(i)}))^{\omega_i}}. \tag{8}$$

Dec$(PK, \mathcal{C}_{w'}, CT^*, \xi', SK'_{u,1})$: To obtain the file encryption key sk_f, the DU computes $sk_f = C''_f / \xi'^{Tr}$, which just takes one exponentiation operation E_T to recover each file encryption key.

a. The modified MABKS system facilitates online/offline encryption and outsourced decryption, which greatly reduces the DO and DUs' computation burden. It is worth noticing that the outsourced decryption does not incur burdensome computation burden on CSP as its computation overhead still grows with the number of DUs' submitted attributes rather than system attributes. Besides, we just assess the performance of our original MABKS schemes in Section 6.2 as these modified algorithms do not bring much additional computation costs.

Fig. 7: Modified MABKS system

noticing that the tracing process periodically conducted by CA is divided into two phases, namely secret key ownership confirming and malicious AA tracing.

- To confirm that the suspected DU really owns submitted secret key components, the CA first checks the key components $(k'_{j,u,\tau,1}, K_3)$ by asking DU with identity ID_u. Notice that two random elements $\alpha, \beta \in \mathcal{Z}_p$ in **KeyGen** are redefined as $\alpha = \mathcal{H}(ID_u \| Ts \| 0)$, $\beta = \mathcal{H}(ID_u \| Ts \| 1)$, and the AA_j sends $(SK_{u,0}, S, ID_j, ID_u, Ts)$ to CA, where Ts denotes the timestamp and $SK_{u,0} = \{k'_{j,\tau,u,0}, k''_{j,\tau,u,0}\}_{\tau \in S}$. Then, the CA computes $\alpha^* = \mathcal{H}(ID_u \| Ts \| 0) = \alpha$, $\beta^* = \mathcal{H}(ID_u \| Ts \| 1) = \beta$, $k^*_{j,\tau,u,0} = H_\tau^{a_0 \beta} g^{-b_1(\alpha^* + \beta^*)}$. Finally, the CA checks whether the following Eq. 9 holds. If Eq. 9 holds, the suspected DU indeed owns aforementioned secret key components; otherwise, the suspected DU should be punished and then CA proceeds to next phase.
$$e(H_\tau, K_3) = e(g, k'_{j,u,\tau,1} k^*_{j,\tau,u,0}). \tag{9}$$
- To further trace which AA has maliciously or incorrectly generated the secret key for above suspected DU, the CA first computes $PK^* = (K_3 g^{-a_0 \beta})^{1/b_0 \alpha}$. Then, the CA traverses the public keys of AAs, if the public key PK_j of AA_j is equal to PK^*, we can say that the AA_j does not accurately verify the legitimacy of suspected DU and the AA_j should be punished or kicked out of the trusted domain.

5.3 Extension to Support Attribute Update

In practice, the changes of DUs' access permissions or roles require that the CA should generate new secret keys so that the DUs cannot access unauthorized information by reusing their old or outdated secret keys. To enhance the practicability and feasibility of the MABKS system in actual scenarios, we take the attribute update into account in the MABKS system.

When there are some attributes which need to be updated, the CA first updates the master key MSK and public parameters PK, then generates two transformation keys to update the secret key $SK_{u,1}$ of DU and indexes Ind stored in CSP. Finally, it increases the version number by one so that the DUs whose attributes match with the access structure in other version number cannot generate valid search tokens. Notice that the MABKS system just updates a fraction of secret keys and indexes rather than the whole ciphertexts. Thus, the MABKS system is suitable for storage and computation resource-limited devices. The modified algorithms are shown as follows:

- **Setup**: Let $\mathcal{U}^* \subseteq \mathcal{U}$ be the updated attributes. For each attribute $Att^* \in \mathcal{U}^*$, the CA first selects a new random element $\nu^* \in \mathcal{Z}_p$ to update MSK and computes $H_{Att^*} = B_0^{\nu^*}$ to update PK, respectively. Then, the CA outputs two transformation keys $\nu' = \nu^* - \nu_{Att}$, $\nu'' = \nu^*/\nu_{Att}$, where ν_{Att} is the original random element chosen for attribute Att^*. Finally, the CA sends ν', ν'' to CSP and DU to update Ind and $SK_{u,1}$, respectively.
- **KeyGen**: Once receiving ν'', the DU updates secret key component $k'''_{j,u,Att^*,1} = B_0^{\nu_{Att}\gamma_u}$ by computing $k^*_{j,u,Att^*,1} = (k'''_{j,u,Att^*,1})^{\nu''} = H_{Att^*}^{\gamma_u} = B_0^{\nu^*\gamma_u}$, where Att^* denotes the updated attribute belonging to S.
- **Enc**: After gaining ν', the CSP just needs to update index component $I_{1,i} = \varpi_i^s$ as $I^*_{1,i}$ by computing $I^*_{1,i} = I_{1,i} \cdot I_3^{\nu'} = H_{\rho(i)}^{s/\mathcal{H}(w)} \cdot B_0^{(s\nu^* - s\nu_{Att})/\mathcal{H}(w)} = \varpi^s_{Att^*}$, where ϖ_{Att^*} is set as $H_{Att^*}^{1/\mathcal{H}(w)}$.

6 Analysis of MABKS System

In this section, the security and performance of MABKS system are presented, respectively. Due to space limitation, the correctness of the MABKS system is presented in **Supplemental Material C**.

6.1 Security

In this section, we first prove that the MABKS system is secure in selective-matrix and selective-attribute models with following two theorems, then we show that the MABKS system can resist user collusion attack and AA collusion attack.

Theorem 1. There does not exist a polynomial-time adversary that can selectively break the MABKS system with a challenge matrix $\mathcal{M}^*$ of size $l^* \times n^*$ on condition that the decisional q-parallel BDHE assumption holds, where $l^*, n^* \leq q$.

Proof: For convenience, the secret key components $K_2 = g^{a_0} PK_j^{a_1 b_0 \alpha} g^{a_1 a_0 \beta}$, $K_3 = PK_j^{b_0\alpha} g^{a_0\beta}$ can be reduced as $K_2 = g^{a_0} g^{a_1(k_j b_0 \alpha + a_0\beta)} = g^{a_0} g^{a_1 \eta}$, $K_3 = g^{k_j b_0 \alpha + a_0 \beta} = g^{\eta}$, where $k_j b_0 \alpha + a_0 \beta = \eta$. Besides, for each attribute $\psi_\tau = k'_{j,u,\tau,1} \cdot k''_{j,u,\tau,1} = H_\tau^{\eta}$. Thus, the tuple $(K_2, K_3, \{\psi_\tau\})$ is consistent with secret key form in CP-ABE scheme [33].

Assume that an adversary $\mathcal{A}$ can break the MABKS system with a non-negligible advantage δ and choose a challenging matrix $\mathcal{M}^*_{l^* \times n^*}$, we need to build a simulator $\mathcal{B}$ that plays the decisional q-parallel BDHE problem as follows. The detail proof of **Theorem** 1 is shown in **Supplemental Material D**. □

Theorem 2. The MABKS system is secure in the selective-attribute model on the condition that the DBDH assumption holds.

Proof: Assume that an adversary $\mathcal{A}$ can break the MABKS system, then there must exist a simulator $\mathcal{B}$ that can use $\mathcal{A}$ to break the DBDH assumption as follows:

Given the bilinear map parameters (G, G_T, e, p, g), the challenger $\mathcal{C}$ first selects a random bit $\kappa \in \{0,1\}$. If $\kappa = 0$, $\mathcal{C}$ returns a tuple $(g^a, g^b, g^c, e(g,g)^{abc})$ to $\mathcal{B}$; otherwise, $\mathcal{C}$ sends the tuple $(g^a, g^b, g^c, e(g,g)^z)$ to $\mathcal{B}$, where $a, b, c, z \in \mathcal{Z}_p$. Then, $\mathcal{B}$ outputs a guess bit $\kappa' \in \{0,1\}$ and sets $A_0 = g^{a_0} = g^a$, $g_0 = g^b$, $g^* = g^c$. The detail proof of **Theorem** 2 is shown in **Supplemental Material E**. □

Apart from guaranteeing the MABKS system security, the user collusion attack can be avoided in the MABKS system and AA collusion attack can be prevented in the extended MABKS system. Similar to the CP-ABE scheme [33], the MABKS system prevents user collusion attack by assigning a global identity ID_u for each DU. In **KeyGen** , secret key components $(K_2, K_3, \{k'_{j,u,\tau,1}, k''_{j,u,\tau,1}\})$ are associated with a common random value $k_j b_0 \alpha + a_0 \beta \in \mathcal{Z}_p$, and other secret key components $(K_0, K_1, \{k'''_{j,u,\tau,1}\})$ are confused by a random element $\gamma_u \in \mathcal{Z}_p$. Thus, it is difficult for malicious DUs to collude and gain valid secret keys without random values $(k_j b_0 \alpha + a_0 \beta, \gamma_u)$.

Furthermore, the malicious AAs should not collude with each other to spy out some sensitive information. If two collusive AAs want to generate the valid secret key for the same DU, they must have the same values α, β. Assume that malicious AA_1, AA_2 can generate the secret key components $(K_{2,1}, K_{3,1}, \{k'_{j,u,\tau,1,1}, k''_{j,u,\tau,1,1}\})$, $(K_{2,2}, K_{3,2}, \{k'_{j,u,\tau,1,2}, k''_{j,u,\tau,1,2}\})$ for DU with identity ID_u, and random elements α, β are different, AA_1 and AA_2 can first have $\frac{K_{2,1}}{K_{2,2}} = g^{a_1 b_0 \alpha (k_1 - k_2)}$, $\frac{K_{3,1}}{K_{3,2}} = g^{b_0 \alpha (k_1 - k_2)}$, $\frac{k'_{j,u,\tau,1,1} k''_{j,u,\tau,1,1}}{k'_{j,u,\tau,1,2} k''_{j,u,\tau,1,2}} = H_\tau^{b_0 \alpha (k_1 - k_2)}$, then they can gain $g^{a_0} = (\frac{K_{3,1}}{(K_{3,1}/K_{3,2})^{k_1/(k_1-k_2)}})^{1/\beta}$. Finally, they can compute $g^{a_1 b_0}$, $g^{a_1 a_0}$, $H_\tau^{a_0}$, $H_\tau^{b_0}$. However, if the CA has kept the used α, β, these two AAs will have different random elements (α, β) and thus cannot generate the valid secret keys for the DU. Therefore, the MABKS system can resist user collusion attack [33] and AAs collusion attack [18].

6.2 Performance

Compared with prior schemes (i.e., HP-CPABKS [43], ABKS-UR [14]), we show the performance analysis (e.g., theoretical performance, actual performance, etc.) of the MABKS system. Note that we assume that each attribute just has one value in HP-CPABKS scheme.

TABLE 4: Theoretical computation costs in various schemes

Schemes	MABKS	HP-CPABKS [43]	ABKS-UR [14]
KeyGen	$(3\|S\|+8)E$	$(2\|U\|+1)E+E_T$	$(2\|U\|+1)E+E_T$
Enc	$(4\|U\|+3)E+2E_T+P$	$(2\|U\|+1)E+E_T$	$(\|U\|+1)E+E_T$
Trap	$(2\|S\|+2)E$	$(2\|U\|+1)E$	$(2\|U\|+1)E$
Search	$2P$	$(2\|U\|+1)P+E_T$	$(\|U\|+1)P+E_T$
Dec	$(2\|S\|+1)P+\|S\|E_T$	—	—

Notes. "$|U|$": Number of system attributes; "$|S|$": Number of submitted attributes; "—": Without consideration.

Firstly, we present the theoretical performance regarding computation and storage costs in TABLE 4 and TABLE 5, respectively. As for the computation costs of aforementioned schemes in TABLE 4, we mainly take several time-consuming operations into consideration, namely bilinear pairing operation P, exponentiation operation E (or E_T) in group G (or G_T). In **KeyGen**, the secret key generation time

TABLE 5: Theoretical storage costs in various schemes

Schemes	MABKS	HP-CPABKS [43]	ABKS-UR [14]
KeyGen	$(3\vert S\vert+4)\vert G\vert+3\vert\mathcal{Z}_p\vert$	$\Theta+(\vert U\vert+2)\vert\mathcal{Z}_p\vert$	$\vert\mathcal{Z}_p\vert+\Theta+\vert G_T\vert$
Enc	$\Theta+\Delta+2\vert G\vert+2\vert G_T\vert$	$\Theta+(\vert U\vert+1)\vert\mathcal{Z}_p\vert$	$\Theta+\vert G_T\vert$
Trap	$\vert\mathcal{Z}_p\vert+2\vert G\vert$	$\Theta+2\vert\mathcal{Z}_p\vert$	$\vert\mathcal{Z}_p\vert+\Theta$
Search	$2\vert G_T\vert$	$4\vert G_T\vert$	$(\vert U\vert+3)\vert G_T\vert$
Dec	$(2\vert S\vert+1)\vert G_T\vert+\vert S\vert\vert G\vert$	—	—

Notes: $\Theta=(2|U|+1)|G|$; $\Delta=|U||G|+|U||\mathcal{Z}_p|$.

in the MABKS system is fewer than that of the other two schemes when setting $|S| \ll |U|$ in practice. Although the computation cost of **Enc** in the MABKS system is more than those of CP-ABKS, HP-CPABKS and ABKS-UR schemes, the **Enc** does not affect user search experience as the ciphertexts generation is just a one-time cost operation. As for trapdoor generation in **Trap**, the MABKS system is obviously superior to remaining schemes due to $|S| \ll |U|$. As the computation overhead of **Search** in the MABKS system is constant and that of the other two schemes is affected by the variable $|U|$, the MABKS system is much more efficient than aforementioned three schemes in terms of ciphertexts retrieval. In **Dec**, we just analyze the ciphertexts decryption time in the MABKS system without considering other schemes since the encrypted data in HP-CPABKS and ABKS-UR schemes are decrypted by traditional symmetric or public-key algorithms. In general, the MABKS system is efficient and feasible in practice, which can be applied in a broad range of applications. In TABLE 5, element lengths in $G, G_T, \mathcal{Z}_p$ are defined as $|G|$, $|G_T|$ and $|\mathcal{Z}_p|$, respectively. With the same reasons about computation overhead analysis, the MABKS system has fewer storage costs in **KeyGen**, **Trap** and **Search** than the other two schemes. It is worth noticing that the MABKS system has constant trapdoor size and storage cost of ciphertexts retrieval, which is well suited for resource-constrained devices (e.g., sensor nodes, smartphones, etc.).

Secondly, we conduct a series of experiments to assess the actual performance of aforementioned schemes by utilizing the real-world Enron Email Dataset[2]. This public dataset contains about 517431 emails from 151 users distributed in 3500 folders, and its size is about 422 MB. Each message presented in the folders contains the senders, receiver email address, data, time, subject, body, text and some other specific technical details. The experimental simulations are conducted on an Ubuntu Server 15.04 with Intel Core i5 Processor 2.3 GHz by utilizing C and Paring Based Cryptography (PBC) Library. It is worth noticing that we select the Type A as $E(F_q): y^2 = x^3 + x$, and the groups G, G_T of order p as the subgroups of $E(F_q)$. When the lengths of parameters p and q are set as 160 bits and 512 bits, respectively, then we can have $|\mathcal{Z}_p| = 160$ bits, $|G| = |G_T| = 1024$ bits. Finally, 10000 files are chosen from this public dataset, and the experiments are executed 100 times. For ease of comparison, we set $|U| \in [1, 100]$, $|S| \in [1, 50]$.

In Figs. 8a, 8b, the computation and storage costs of the MABKS system are both affected by the number of user attributes ($|S|$), while those of other schemes almost linearly increase with the number of system attributes ($|U|$). In **KeyGen**[3], when setting $|U| = 100$, we notice that the computation cost of MABKS linearly increases with increasing the value of $|S|$, but the computation costs of HP-CPABKS and ABKS-UR schemes still keep unchanged. With the same reason shown in the analysis of computation overhead in the key generation process, we can draw a similar conclusion when comparing the storage costs in various schemes. Thus, we can say that the MABKS system outperforms the other two schemes in terms of key generation due to $|S| \ll 50$ in practice.

Apart from analyzing the CA's key generation cost with equipping with multiple AAs, we also compare the CA's computation costs in two cases (i.e., without AAs, with AAs, etc.). Furthermore, we demonstrate how many key generation queries can be processed by CA in above two cases in one minute. Finally, we show the CA's computation cost for tracing each malicious AA. The CA's additional performance analysis about reduced rate and each AA tracing is shown in TABLE 6.

TABLE 6: CA's performance in key generation and each AA tracing processes

Value of $\vert S\vert$	Without AAs	With AAs	Reduced rate	Tracing costs
10	(424 ms, 141)	(138 ms, 435)	67%	247 ms
20	(666 ms, 90)	(248 ms, 242)	63%	423 ms
30	(862 ms, 70)	(374 ms, 161)	57%	661 ms
40	(1151 ms, 52)	(503 ms, 120)	57%	968 ms
50	(1488 ms, 40)	(614 ms, 98)	59%	1145 ms

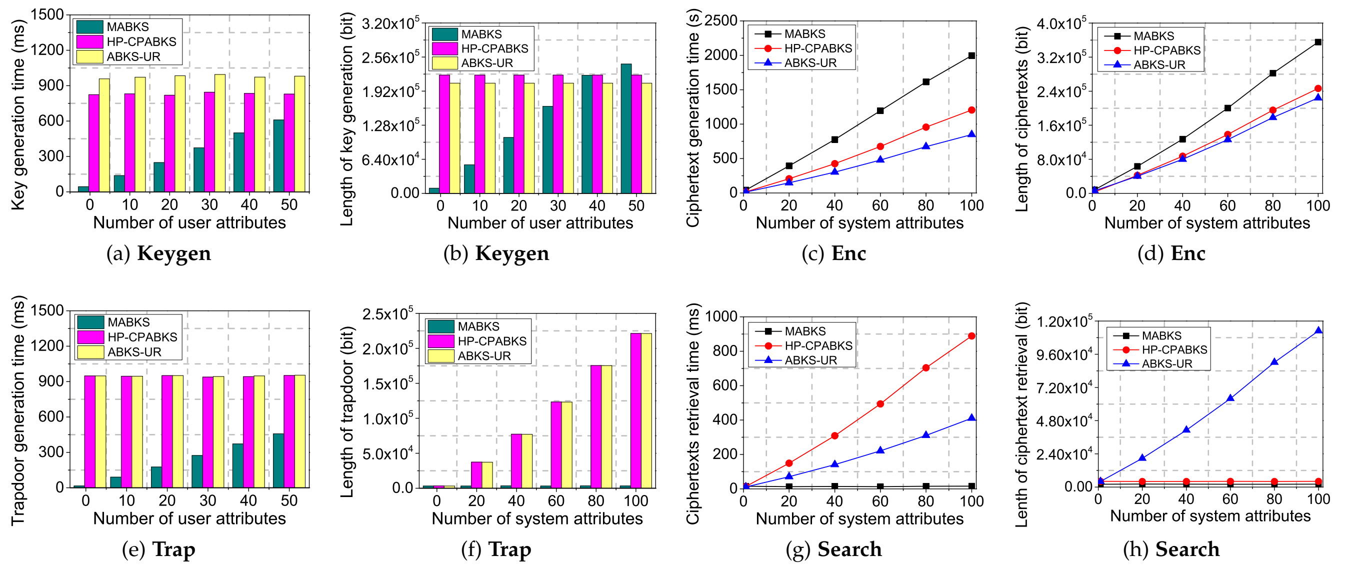
Notes. "$|S|$": Number of submitted attributes; "The symbol $(*, *)$" in above two cases (i.e., without AAs, with AAs, etc.) represents the CA's key generation cost for each DU and the number of key generation queries processed by CA in one minute; Reduced rate means that how much CA's computation burden on each DU's key generation can be reduced by comparing above two cases; AA tracing cost means that the CA's computation cost for each AA tracing.

As for the **Enc** in Figs. 8c, 8d, the MABKS system needs to build indexes as well as encrypt file keys, but the other schemes just generate the searchable indexes without considering record key encryption. Therefore, the MABKS system has higher computation and storage overhead when compared with the other two schemes, but it will not affect user search experience as ciphertexts generation process is executed only once in cloud systems. Besides, we have shown the idea that how to reduce the DO's computation burden by offering online/offline encryption mechanism in Section 5.1.

When comparing the performance of trapdoor generation in Figs. 8e, 8f, we also fix the value of $|U|$ as 100 and vary that of $|S|$ from 1 to 50. We notice that the computation cost of the MABKS system increases with the value of $|S|$, while those of other schemes remain nearly constant. Besides, the length of trapdoor generation of the MABKS system in **Trap** is constant $|\mathcal{Z}_p| + 3|G|$, and the storage costs of other two schemes are affected by the value of $|U|$. Hence, the performance of trapdoor generation in the MABKS system is superior to that of the other two schemes. When setting $|S| = 50$, $|U| = 100$, the MABKS system needs (458 ms, 0.41 KB) to generate the search token, but the CP-

2. http://www.cs.cmu.edu/~enron/

3. We just analyze the CA's computation and storage costs in MABKS without considering the selected AA.



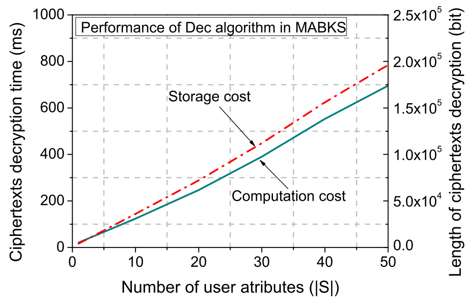
(a) **Keygen** (b) **Keygen** (c) **Enc** (d) **Enc**

(e) **Trap** (f) **Trap** (g) **Search** (h) **Search**

Fig. 8: Practical performance analysis in various algorithms.

ABKS, HP-CPABKS and ABKS-UR schemes take (952 ms, 27.03 KB), (954 ms, 27.01 KB) to issue trapdoor generations, respectively. Thus, the MABKS system is appropriate for most applications with resource-constrained devices.

In Figs. 8g, 8h, the performance of ciphertexts retrieval in the MABKS system in **Search** is superior to that of the other two schemes. The computation cost ($2P$) and storage cost ($2|G_T|$) of the MABKS system are constant, but the computation overhead of other aforementioned two schemes almost linearly increases as the value of $|U| \in [1, 100]$ increases. Furthermore, the storage cost of ABKS-UR scheme is still affected by the variable $|U|$. For instance, when setting $|S| = 40, |U| = 40$, the computation and storage costs of ciphertexts retrieval in the MABKS system are 14 ms and 0.26 KB, and those of other schemes (e.g., HP-CPABKS, ABKS-UR) are (309 ms, 0.50 KB), (141 ms, 5.02 KB), respectively.



(a) **Dec**

Fig. 9: Performance analysis of **Dec** in MABKS.

In Fig. 9a, we just analyze the performance of the MABKS system in **Dec** as other schemes do not need to decrypt record encryption keys. By encrypting symmetric keys with DO's specified strategy, the MABKS system can achieve file-level fine-grained access control and further guarantee keys security even though malicious DUs collude with each other. Besides, both the computation and storage costs increase with increasing the number of user attributes $|S| \in [1, 50]$. For instance, when setting $|S| = 20$, the computation and storage costs of MABKS system are 247 ms and 8.80 KB during decrypting each search result. In the modified MABKS system, we also show how to outsource ciphertexts decryption tasks to CSP. Then, the DU just needs one exponentiation operation to recover each file key. Note that outsourced decryption mechanism does not incur much computation burden on CSP as the outsourced decryption overhead grows with the value of S instead of U.

To further evaluate the performance of the MABKS system as well as other schemes (e.g., HP-CPABKS scheme, ABKS-UR scheme) in terms of **KeyGen**, **Enc**, **Trap** and **Search**, we also conduct a large number of tests over other datasets (e.g., National Science Foundation Research Awards Abstract 1990-2003 dataset (or NSF dataset)[4], the Request For Comments database (or RFC dataset)[5]). For comparison, we set $|U| = 100$, $|S| = 50$, randomly select 10000 files from these three datasets, and conduct the experiments 100 times. The test results are shown in TABLE 7. The performance of aforementioned four schemes is approximately similar in different datasets. Besides, the performance of the MABKS systems outperforms that of other schemes except for the **Enc** algorithm.

In summary, the performance assessment of the above schemes echoes those of the theoretical analysis shown in TABLE 4 and TABLE 5. In supporting a heterogeneous architecture and fine-grained keyword search functionalities, the MABKS system does not incur additional computation and storage costs. Furthermore, our extended MABKS system only executes $(2P + E)|S''|$, $(2|\mathcal{U}^*| + 1)E$ operations to trace AAs and update attributes, where $|S''|$ denotes the number of suspected attributes in S and $|\mathcal{U}^*|$ represents the number of updated attributes. Clearly, it does not incur high computation overhead as the values of variables $|S''|$, $|\mathcal{U}^*|$

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

TABLE 7: The actual performance analysis in different datasets

Schemes		MABKS		HP-CPABKS [43]		ABKS-UR [14]	
Performance		CC	SC	CC	SC	CC	SC
KeyGen	Enron	610 ms	29.65 KB	829 ms	27.12 KB	981 ms	25.27 KB
	NSF	589 ms	28.87 KB	779 ms	26.49 KB	974 ms	25.18 KB
	RFC	641 ms	29.94 KB	883 ms	27.31 KB	991 ms	26.03 KB
Enc	Enron	1997 s	433.8 MB	1206 s	300.8 MB	848 s	273.6 MB
	NSF	1864 s	419.3 MB	1178 s	289.6 MB	837 s	269.4 MB
	RFC	2049 s	453.5 MB	1295 s	314.2 MB	853 s	277.4 MB
Trap	Enron	458 ms	0.41 KB	952 ms	27.03 KB	954 ms	27.01 KB
	NSF	432 ms	0.38 KB	938 ms	26.69 KB	941 ms	26.86 KB
	RFC	479 ms	0.48 KB	978 ms	27.65 KB	971 ms	27.43 KB
Search	Enron	16 ms	0.26 KB	889 ms	0.50 KB	41 ms	13.78 KB
	NSF	15 ms	0.24 KB	864 ms	0.46 KB	37 ms	13.65 KB
	RFC	19 ms	0.27 KB	912 ms	0.51 KB	52 ms	13.86 KB

Notes. "CC": Computation Costs; "SC": Storage Costs.

are very small. Hence, the MABKS system is practical in a broad range of applications.

7 Conclusion

In this paper, we proposed an efficient and feasible MABKS system to support multiple authorities, in order to avoid having performance bottleneck at a single point in cloud systems. Furthermore, the presented MABKS system allows us to trace malicious AAs (e.g., to prevent collusion attacks) and support attribute update (e.g., to avoid unauthorized access using outdated secret keys). We then demonstrated the selective security level of the system in selective-matrix and selective-attribute models under decisional q-parallel BDHE and DBDH assumptions, respectively. We also evaluated the system's performance and demonstrated that significant computation and storage cost reductions were achieved, in comparison to prior ABKS schemes. However, the main flaw is that the MABKS system cannot support expressive search queries such as conjunctive keyword search, fuzzy search, subset search and so on. The future work will focus on building an efficient and flexible index construction so that the MABKS system is capable of supporting various search requests.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. 61702404, No. 61702105, No. U1804263), the Fundamental Research Funds for the Central Universities (No. JB191506), the National Natural Science Foundation of Shaanxi Province (No. 2019JQ-005), the China Postdoctoral Science Foundation Funded Project (No. 2017M613080), the 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.

References

[1] Y. T. Demey and M. Wolff, "Simiss: A model-based searching strategy for inventory management systems," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 172–182, 2017.
[2] C. Huang, R. Lu, H. Zhu, J. Shao, and X. Lin, "Fssr: Fine-grained ehrs sharing via similarity-based recommendation in cloud-assisted ehealthcare system," in *Proc. ACM on Asia Conference on Computer and Communications Security (AsiaCCS'16)*, 2016, pp. 95–106.
[3] 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.
[4] Y. Miao, J. Ma, X. Liu, J. Weng, H. Li, and H. Li, "Lightweight fine-grained search over encrypted data in fog computing," *IEEE Transactions on Services Computing*, vol. PP, no. 1, pp. 1–14, 2018.
[5] 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, no. 1, pp. 1–14, 2017.
[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)*, 2000, pp. 44–55.
[7] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'04)*, vol. 3027, 2004, pp. 506–522.
[8] H. Li, D. Liu, Y. Dai, T. H. Luan, and S. Yu, "Personalized search over encrypted data with efficient and secure updates in mobile clouds," *IEEE Transactions on Emerging Topics in Computing*, vol. 6, no. 1, pp. 97–109, 2018.
[9] 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.
[10] X. Zhang, Y. Tang, H. Wang, C. Xu, Y. Miao, and H. Cheng, "Lattice-based proxy-oriented identity-based encryption with keyword search for cloud storage," *Information Sciences*, vol. PP, pp. 1–15, 2019.
[11] J. Li, Y. Huang, Y. Wei, S. Lv, Z. Liu, C. Dong, and W. Lou, "Searchable symmetric encryption with forward search privacy," *IEEE Transactions on Dependable and Secure Computing*, vol. PP, pp. 1–15, 2019.
[12] 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.
[13] 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)*, 2014, pp. 522–530.
[14] 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.
[15] L. Harn and J. Ren, "Generalized digital certificate for user authentication and key establishment for secure communications," *IEEE Transactions on Wireless Communications*, vol. 10, no. 7, pp. 2372–2379, 2011.
[16] M. Chase, "Multi-authority attribute based encryption," in *Proc. IACR Theory of Cryptography Conference (TCC'07)*, 2007, pp. 515–534.
[17] K. Yang and X. Jia, "Expressive, efficient, and revocable data access control for multi-authority cloud storage," *IEEE transactions on parallel and distributed systems*, vol. 25, no. 7, pp. 1735–1744, 2014.
[18] K. Xue, Y. Xue, J. Hong, W. Li, H. Yue, D. S. Wei, and P. Hong, "Raac: Robust and auditable access control with multiple attribute authorities for public cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 4, pp. 953–967, 2017.
[19] V. K. A. Sandor, Y. Lin, X. Li, F. Lin, and S. Zhang, "Efficient decentralized multi-authority attribute based encryption for mobile cloud data storage," *Journal of Network and Computer Applications*, vol. 129, pp. 25–36, 2019.
[20] J. Ning, X. Dong, Z. Cao, L. Wei, and X. Lin, "White-box traceable ciphertext-policy attribute-based encryption supporting flexible attributes," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 6, pp. 1274–1288, 2015.
[21] Y. Yang, X. Liu, X. Zheng, C. Rong, and W. Guo, "Efficient traceable authorization search system for secure cloud storage," *IEEE Transactions on Cloud Computing*, vol. PP, pp. 1–14, 2018.
[22] J. Ning, Z. Cao, X. Dong, and L. Wei, "White-box traceable cp-abe for cloud storage service: how to catch people leaking their access credentials effectively," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 5, pp. 883–897, 2018.
[23] J. Hur and D. K. Noh, "Attribute-based access control with efficient revocation in data outsourcing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 7, pp. 1214–1221, 2011.

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

[25] Q. Xu, C. Tan, W. Zhu, Y. Xiao, Z. Fan, and F. Cheng, "Decentralized attribute-based conjunctive keyword search scheme with online/offline encryption and outsource decryption for cloud computing," *Future Generation Computer Systems*, vol. PP, pp. 1–33, 2019.

[26] 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.

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

[28] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Transactions on parallel and distributed systems*, vol. 25, no. 1, pp. 222–233, 2014.

[29] 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.

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

[31] 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)*, 2012, pp. 917–922.

[32] 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)*, 2015, pp. 2110–2118.

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

[34] A. Balu and K. Kuppusamy, "An expressive and provably secure ciphertext-policy attribute-based encryption," *Information Sciences*, vol. 276, pp. 354–362, 2014.

[35] J. Li, W. Yao, J. Han, Y. Zhang, and J. Shen, "User collusion avoidance cp-abe with efficient attribute revocation for cloud storage," *IEEE Systems Journal*, 2017.

[36] P. Zhang, Z. Chen, K. Liang, S. Wang, and T. Wang, "A cloud-based access control scheme with user revocation and attribute update," in *Proc. Australasian Conference on Information Security and Privacy (ACISP'16)*, 2016, pp. 525–540.

[37] Z. Wan, J. L. Liu, and R. H. Deng, "Hasbe: a hierarchical attribute-based solution for flexible and scalable access control in cloud computing," *IEEE transactions on information forensics and security*, vol. 7, no. 2, pp. 743–754, 2012.

[38] H. Cui, R. H. Deng, G. Wu, and J. Lai, "An efficient and expressive ciphertext-policy attribute-based encryption scheme with partially hidden access structures," in *Proc. International Conference on Provable Security (ProvSec'16)*, 2016, pp. 19–38.

[39] X. Mao, J. Lai, Q. Mei, K. Chen, and J. Weng, "Generic and efficient constructions of attribute-based encryption with verifiable outsourced decryption," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 5, pp. 533–546, 2016.

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

[41] 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.

[42] 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.

[43] S. Qiu, J. Liu, Y. Shi, and R. Zhang, "Hidden policy ciphertext-policy attribute-based encryption with keyword search against keyword guessing attack," *Science China Information Sciences*, vol. 60, no. 5, p. 052105, 2017.

[44] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization." in *Proc. International Conference on Practice and Theory in Public Key Cryptography (PKC'11)*, vol. 6571, 2011, pp. 53–70.

[45] A. Sahai and B. Waters, "Fuzzy identity-based encryption." in *Proc. Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'05)*, vol. 3494, 2005, pp. 457–473.

[46] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Proc. Annual International Cryptology Conference on Advances in Cryptology (CRYPTO'01)*, 2001, pp. 213–229.

[47] W. T. Polk and N. E. Hastings, "Bridge certification authorities: Connecting b2b public key infrastructures," in *PKI Forum Meeting Proceedings*, 2000, pp. 27–29.

[48] S. Hohenberger and B. Waters, "Online/offline attribute-based encryption," in *Proc. International workshop on public key cryptography (PKC'14)*. Springer, 2014, pp. 293–310.

[49] M. Green, S. Hohenberger, B. Waters *et al.*, "Outsourcing the decryption of abe ciphertexts." in *Proc. USENIX Security Symposium (USENIX'11)*, vol. 2011, no. 3, 2011.

[50] J. Ning, Z. Cao, X. Dong, K. Liang, H. Ma, and L. Wei, "Auditable σ-time outsourced attribute-based encryption for access control in cloud computing," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 1, pp. 94–105, 2018.

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.

Robert H. Deng (F'16) is AXA Chair Professor of Cybersecurity and Professor of Information Systems in the School of Information Systems, Singapore Management University since 2004. His research interests include data security and privacy, multimedia security, network and system security. He served/is serving on the editorial boards of many international journals, including 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.

Ximeng Liu (M'16) received the B.E. degree with the Department of Electronic Engineering from Xidian University, Xi'an, China, in 2010 and Ph.D. degree with the Department of Telecommunication Engineering from Xidian University, Xi'an, China in 2015. He is currently a post-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 various awards including the UTSA College of Business Col. Jean Piccione and Lt. Col. Philip Piccione Endowed Research Award for Tenured Faculty in 2018, ESORICS 2015 Best Paper Award. He is an Australian Computer Society Fellow, and an IEEE Senior Member.

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.