

Time-Controllable Keyword Search Scheme With Efficient Revocation in Mobile E-Health Cloud

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

Abstract—Electronic health (e-health) systems may outsource data such as patient e-health records to mobile cloud servers for efficiency gains (e.g., minimizing local storage and computation costs). However, such a move may result in privacy implications in the presence of semi-honest cloud servers. Searchable Encryption (SE) can potentially facilitate privacy-preserving searches based on keywords for encrypted data stored in the mobile cloud, but most existing SE solutions do not support temporal access control (i.e., a mechanism that grants access permissions to users for specified time ranges). Hence, in this paper we design a time-controllable keyword search scheme by using an attribute-based comparable access control. This allows users to match indexes encrypted at specified time intervals. Then, we improve the basic framework to support efficient user revocation using secret sharing. We then formally prove the security of our proposed frameworks against chosen-keyword attack and key collusion attack, as well as achieving keyword secrecy. We also evaluate the performance of our proposed approach using a real-world dataset to demonstrate their practical utility.

Index Terms—Chosen-keyword attack, comparable access control, e-health system, searchable encryption, user revocation.

I. INTRODUCTION

CLOUD-BASED electronic health (e-health) systems are increasingly common, although a number of challenges remain. For example, to support sensitive services (e.g., record sharing and medical diagnosis through mobile devices) to be rendered in a mobile cloud-based environment, data need to be encrypted prior to being outsourced to the cloud server. Hence, this motivates the use of cryptographic solutions such as Searchable Encryption (SE) [1], [2], which facilitates keyword search over encrypted data. However, existing SE solutions [3], [4] do not generally support flexible access control. Existing access control approaches can be broadly categorized into authentication-based approaches [5] and encryption-based approaches [6], [7]. The former approach (e.g., password authentication) is generally a server-dominated access control and assumes that the cloud server is fully trusted. In other words, privacy cannot be assumed when the associated cloud server is compromised. Encryption-based approaches, on the other hand, are implemented using common cryptographic primitives such as Identity-Based Encryption (IBE) [8] and Attribute-Based Encryption (ABE) [9], [10]. The encryption-based approaches are promising in facilitating access control in SE, but there are still two key limitations, namely: flexible time-controllable keyword search and efficient user revocation.

A large amount of time-series data are generated for the ubiquitous existence of smart devices [11], which forces the need for flexible time-controllable keyword search. Let us consider a mobile e-health system that is designed to support keyword searches. As depicted in Fig. 1, a patient with mobile wearable devices builds indexes in a specified time interval $[t_i, t_j]$ and allows doctors to make queries on the encrypted data within a specified time range, say $[t'_i, t'_j]$. Two common techniques (e.g., temporary search and timing enabled function) can be used to achieve time-controllable keyword search. The temporary keyword search mechanism [12], [13] needs to generate multiple indexes if the same keyword is encrypted at different time cells (e.g., $t_1, \dots, t_{\max}$). However, this incurs significant computation and storage costs for the patient. The timing enabled proxy re-encryption function [14], [15] just generates a single index for each keyword in specified time interval $[t_i, t_j]$,

Manuscript received 5 October 2022; revised 5 May 2023; accepted 15 May 2023. Date of publication 18 May 2023; date of current version 4 April 2024. The work of Kim-Kwang Raymond Choo was supported by the Cloud Technology Endowed Professorship. This work was supported in part by the National Natural Science Foundation of China under Grant 62072361, in part by the Key Research and Development Program of Shaanxi under Grant 2022GY-019, in part by Shaanxi Fundamental Science Research Project for Mathematics and Physics under Grant 22JSY019, in part by the National Natural Science Foundation of China under Grant 62125205, in part by the National Natural Science Foundation of China under Grant 62272195, in part by Fellowship pf China Postdoctoral Science Foundation under Grant 2022T150507, in part by the Opening Project of Intelligent Policing Key Laboratory of Sichuan Province under Grant ZNJW2023KFMS002. Recommended for acceptance by D. Mohaisen. (Corresponding author: Zhiquan Liu.)

Yinbin Miao and Feng Li are with the School of Cyber Engineering, Xidian University, Xi'an 710071, China, and also with the Henan Key Laboratory of Network Cryptography Technology, State Key Laboratory of Mathematical Engineering, Advanced Computing, Zhengzhou 450001, China (e-mail: ybmiao@xidian.edu.cn; feng.li@stu.xidian.edu.cn).

Xinghua Li is with the State Key Laboratory of Integrated Services Networks, School of Cyber Engineering, Xidian University, Xian, Shaanxi 710071, China (e-mail: xhli1@mail.xidian.edu.cn).

Zhiquan Liu is with the College of Cyber Security, Jinan University, Guangzhou 510632, China, and also with the Cyberdataforce (Beijing) Technology Ltd, Beijing 100020, China (e-mail: zqliu@vip.qq.com).

Jianting Ning is with the School of Mathematics, Computer Science, Fujian Normal University, Fuzhou 350117, China (e-mail: jtning88@gmail.com).

Hongwei Li is with the Department of Computer Science, Engineering, University of Electronic Science, Technology of China, Chengdu 610051, China (e-mail: hongweili@uestc.edu.cn).

Kim-Kwang Raymond Choo is with the Department of Information Systems, Cyber Security, University of Texas at San Antonio, San Antonio, TX 78249 USA (e-mail: raymond.choo@fulbrightmail.org).

Robert H. Deng is with the School of Information Systems, Singapore Management University, Singapore 178902 (e-mail: robertdeng@smu.edu.sg).

Digital Object Identifier 10.1109/TMC.2023.3277702

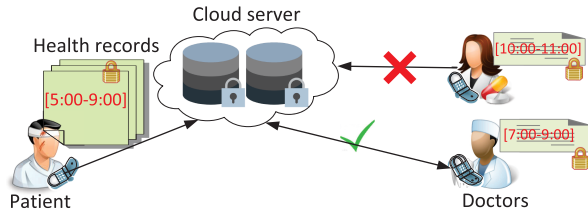


Fig. 1. Simplified mobile e-health system use case.

but still requires authorized Data Users (DUs) to interact with the time server to obtain the time token associated with time interval $[t_i, t_j]$. This also incurs additional communication and computation burdens on mobile users. Furthermore, the second solution cannot grant flexible access permissions associated with time interval $[t'_i, t'_j] \subset [t_i, t_j]$.

Let us then assume that the patient allows a certain doctor to access his/her encrypted e-health records, and the practical keyword search system should enable the patient to deny the access rights of revoked doctors without incurring significant computation costs, using either indirect revocation or direct revocation mechanism. Using the indirect revocation mechanisms [16], [17], [18], [19] allows the authority to maintain the revocation list while issuing a key update algorithm so that only non-revoked users can update their secret keys. The authority not only has to be online in dynamic applications but also has to generate the re-encryption keys. One can observe that such an approach incurs significant communication and computation overheads. The direct revocation mechanism [20], [21] requires the owners to specify the revocation list in the ciphertext generation phase, but does not require non-revoked users to execute the key update operations. However, this mechanism needs to update most of the existing indexes.

From the above discussions, we observe that existing SE solutions cannot efficiently generate indexes and secret keys associated with time intervals, and also are not capable of supporting efficient user revocation without updating the associated secret keys. This observation motivates our research in this paper. Specifically, we first propose a basic Time-Controllable Keyword Search (TCKS) framework for a mobile e-health setting. The framework uses both attribute-based keyword search [7] and comparison-based encryption [22], [23] as the building blocks, in order to achieve fine-grained time-controllable keyword search. Then, we improve the basic framework to construct an enhanced framework (hereafter, referred to as TCKS+) for achieving efficient user revocation. Specifically, a summary of our work is as follows:

- We design a flexible time-controllable keyword search mechanism using the attribute-based comparable access control. It not only allows owners to build indexes according to specified time intervals without incurring significant computation and storage overheads, but also enables any authorized user to generate a valid token using his/her secret key granted within a time range.
- We design an efficient user revocation mechanism based on secret sharing, which allows owners to manage user revocation lists and embed the secret shares of revoked

users into indexes. Such an approach also prevents revoked users from making valid search queries.

- We formally prove that our proposed frameworks are secure against Chosen-Keyword Attack (CKA) and key collusion attack, and conduct extensive experiments to demonstrate the efficiency and feasibility of our proposed frameworks.

The remainder of this paper is organized as follows. We review the related literature in Section II, and present the system model, problem definition, threat model and security model of our proposed frameworks in Section III. We then present our proposed frameworks in Section IV, followed by its security analysis and performance analysis in Sections V and VI, respectively. We finally conclude the paper in the final section.

II. RELATED WORK

As discussed in the preceding section, SE solutions can be broadly divided into two categories [24], namely: Symmetric SE (SSE) [25] and Asymmetric SE (ASE) [2]. SSE generally deployed in a one-to-one model only allows the secret key holders to create searchable indexes and tokens. While ASE is suitable for a one-to-many model, which allows anyone to build indexes with public keys and enables secret key holders to generate valid tokens. Due to high security and practicability, ASE has been extensively studied by researchers and practitioners. We will mainly focus on the typical Public-key Encryption with Keyword Search (PEKS) schemes (belonging to ASE), based on IBE and ABE.

IBE-based keyword search: The traditional PEKS schemes solve the inherent defect of SSE, but also bring efficiency and security concerns, such as complex certificate management and keyword guessing attacks [26], [27]. To solve these issues, Li et al. put forward a designated-server authenticated keyword search solution using IBE [6]. This scheme allows the sender to use his secret key to build indexes for resisting offline keyword guessing attacks, but just supports the single keyword search. The solution proposed by Lu et al. can support conjunctive keyword search while achieving the same security goal [28]. To allow the authority to efficiently make search queries without using the secret keys of authorized users, Liu et al. introduced the dual-token keyword search scheme based on IBE [29]. However, the above solutions are still vulnerable to attackers with quantum computers. Thus, Zhang et al. proposed a proxy-oriented identity-based keyword search solution using lattice-based cryptographic primitive [30]. As for user revocation, Shu et al. designed efficient local server revocation and global server revocation mechanisms in crowdsourcing [31], which incur minimal overheads on the system.

ABE-based keyword search: Due to heavy public key management burden and redundant ciphertexts caused by IBE, ABE that provides fine-grained access control has been applied to SE to form a new notion: Attribute-Based Keyword Search (ABKS) [7], [16]. To search encrypted records that have hierarchical structures, Miao et al. first proposed a basic ABKS scheme supporting single keyword search, and improved it to provide multi-keyword search and user revocation. Then, Miao

TABLE I
A COMPARATIVE SUMMARY BETWEEN OUR SCHEMES AND PREVIOUS SCHEMES

Schemes	Search type	User revocation	Fine-grained access control	Time-Control	Security level	Encryption Mechanism
[7]	Single	✗	✓	✗	CKA	ABE
[12]	Single	✗	✗	✓	CKA	PKE
[13]	Single	✗	✓	✓	CKA	ABE
[14]	Multiple	✗	✗	✓	CKCTA	PKE
[15]	Multiple	✗	✗	✓	CKCTA	PKE
[16]	Single	✓	✓	✗	CKA	ABE
[17]	Single	✓	✓	✗	CPA/CKA	ABE
[21]	Single	✓	✓	✗	CKA	ABE
[31]	Single	✓	✗	✗	CKA	IBE
[37]	Single	✗	✗	✓	CKA	PKE
[38]	Single	✗	✗	✓	CKA/KGA	PKE
[39]	Multiple	✗	✓	✓	KGA	PKE
[40]	Single	✓	✓	✓	SCKA	ABE
TCKS	Single	✗	✓	✓	CKA	ABE
TCKS+	Single	✓	✓	✓	CKA	ABE

Notes. "CKA": Chosen-Keyword Attack; "SCKA": Selective Chosen-Keyword Attack; "KGA": Keyword Guessing Attack; "CKCTA": Chosen Keyword Chosen-Time Attack; "PKE": Public Key Encryption.

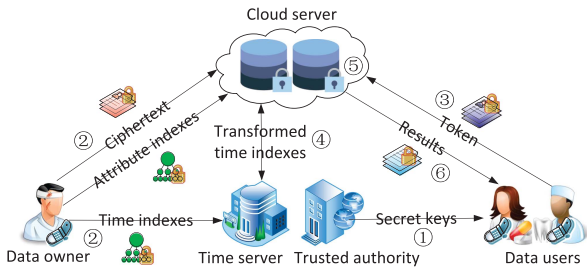


Fig. 2. System model of our proposed frameworks.

et al. deployed ABKS in the shared multi-owner setting where each file is co-owned by multiple owners [32], and extended it to support the multi-authority scenarios [33]. While above solutions cannot support efficient user revocation. Thus, Cui et al. used an administration server to achieve efficient user revocation mechanism without re-encrypting or updating ciphertexts [21], Liu et al. utilized the blockchain to achieve the same goal [17], but these two schemes cannot prevent the privacy leakage of access policy. Motivated by this, Qiu et al. presented a hidden policy ABKS scheme resisting the off-line keyword guessing attack [34], and Wang et al. used the multilinear maps to construct ABKS for supporting hidden policy and fast keyword search [35].

Existing IBE-based or ABE-based SE schemes can achieve both keyword search and user revocation, but cannot meet the various practical requirements such as time-controllable keyword search. Although the 0/1-encoding technique used in the scheme [36] supports attribute comparison, it requires a significant amount of computation and communication resources to represent user attributes as ranges, particularly for large range. Additionally, it cannot support flexible time access control. For example, 0/1-encoding can only determine whether there is an intersection between the time interval $[t_i, t_j]$ and the time range $[t'_i, t'_j]$, but cannot determine whether the current time t_c falls within both $[t_i, t_j]$ and $[t'_i, t'_j]$ simultaneously. This limitation poses a risk of unauthorized access. Yu et al. embedded the time

t_c into indexes and allowed users to generate valid tokens at the time t_c , which achieves revocable keyword search [37] but requires the time synchronization. To allow efficient update indexes at different time cells, Zhang et al. presented an improved revocable keyword search scheme [38] by using hash chain and anonymous receiver encryption. Two schemes [14], [15] both re-encrypt the indexes at a time, but require that owner and user should have the same time. Abdalla et al. proposed a public-key encryption with temporary keyword search scheme [12], in which an attacker cannot distinguish the ciphertexts of two keywords within its chosen time period. Although Ameri et al. designed a time-controllable ABKS scheme [13] allowing the owners to encrypt indexes using multiple time intervals, this scheme cannot prevent the cloud server from searching indexes without time information. Liu et al. proposed a time-controlled hierarchical multi-keyword search scheme [39], in which a new concept of transition time is introduced to achieve the time-based access control. However, this scheme relies on a non-colluding double-server architecture and an online data owner. Xu et al. proposed time-controllable ABKS over encrypted personal health records [40], in which a time attribute is added to a user's key to support simple time control. The access is permitted only if the user's key expiration time X is greater than or equal to the data file's expiration time Y and the rest of the policy attributes match. Then, the time control is used to support user revocation for a piece of data and not for the system. In fact, the data users in [40] do not leave the system.

To solve the above issues, we propose a time-controllable keyword search framework with efficient user revocation. Compared with previous schemes, our proposed frameworks have versatile features demonstrated in Table I.

III. PROBLEM FORMULATION

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

A. System Model

In this paper, we consider a cloud-assisted e-health record sharing application. As depicted in Fig. 2, the system model of our frameworks consists of five entities, namely Data Owner (DO), Data Users (DUs), Cloud Server (CS), Time Server (TS) and Trusted Authority (TA). Both DO (patient) and DUs (doctors) are the users with mobile terminals. DO outsources the storage and management of his health data, which he cannot afford, to CS. Meanwhile, DUs cannot conveniently monitor the vital signs of patients anytime and anywhere using their mobile devices. The role of each entity is shown as follows:

- *Trusted authority*: TA is responsible for distributing public keys for all entities, and generates secret keys for authorized DUs.
- *Data owner*: DO encrypts e-health records, and builds indexes for keywords based on descriptive attributes and time attribute. Then, he outsources encrypted records and descriptive attribute indexes to CS, and sends time indexes to TS. DO can be a mobile user, who is responsible for encrypting a small amount of data collected each time

before uploading it to CS, rather than encrypting a large amount of data at once.

- *Data users*: The authorized DUs can make queries according to their time and descriptive attributes, and submit the tokens of interested keywords to CS. As demonstrated in Section VI-B, the computation and storage costs of a query token are friendly to mobile devices. Thus, DUs can be mobile users.
- *Time server*: TS stores the time indexes and transforms them when DUs make queries, and sends the transformed time indexes to CS.
- *Cloud server*: CS stores record ciphertexts and indexes, and conducts search operations based on DUs' tokens.

In the above application, TA initializes the system and generates the secret key for each DU having both descriptive attributes and time attribute (Step ①). DO first encrypts e-health records and builds indexes for keywords by using the access policy generated from descriptive and time attributes. Then, DO uploads the record ciphertexts and indexes associated with descriptive attributes to CS, and sends the indexes associated with time attribute to TS (Step ②). When making a search query, each DU generates a token of interested keyword using his secret key at one point t_c , and sends the token to CS (Step ③). Upon receiving the token, CS first interacts with TS and obtains the transformed time indexes at time t_c (Step ④), then checks whether the token matches with the indexes (Step ⑤), finally sends query results to DU (Step ⑥).

B. Problem Definition

Let p, q, s_1, s_2, p', q' be secure large primes, where $p + 1 = 2p's_1$, $q + 1 = 2q's_2$. Given two cyclic groups $\mathbb{G}, \mathbb{G}_T$ of composite order $n = s'n'$, where $s' = s_1s_2$, $n' = p'q'$, the bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ has three features: 1) Bilinearity. For any group elements $U, V \in \mathbb{G}$ and random elements a, b in the field $\mathbb{Z}_n^*$, we have $e(U^a, V^b) = e(U, V)^{ab}$; 2) Non-degeneracy. For any generators $U, V \in \mathbb{G}$, we have $e(U, V) \neq 1$; 3) Computability. $e(U, V)$ can be efficiently computed. We define the RSA-type Bilinear map system as $(N, \mathbb{G}, \mathbb{G}_T, e)$ with composite order n [22], where $N = pq$ is the RSA-modulus. Let $\mathbb{G}_{s'}, \mathbb{G}_{n'}$ be two subgroups of order s', n' respectively, we have $e(g, h) = 1$ for any group elements $g \in \mathbb{G}_{s'}, h \in \mathbb{G}_{n'}$. It is worth noticing that $e(g, w) \neq 1, e(h, w) \neq 1$, where $w \in \mathbb{G}$.

In this work, we model the fine-grained keyword search by using the CP-ABE [10] and ABKS [7]. Specifically, DO first embeds the access policy in the indexes to control DUs' access permissions. Then, each DU generates a token tk by using his secret key associated with an attribute set, and can make valid search queries on the condition that his descriptive attributes $Atts$ satisfy the specified access policy T . The access policy including descriptive attributes is described by the following *access tree*:

Let T be a tree that represents the specified access structure. Each non-leaf node of T stands for a threshold gate that is described by its child nodes and threshold value. Given the non-leaf node v , num_v is the number of its child nodes and k_v is its threshold value such that $0 < k_v \leq num_v$. The threshold gate is an "OR" game if $k_v = 1$, and is an "AND" gate if $k_v = num_v$.

Each leaf node v represents an attribute and its threshold value is $k_v = 1$. we define the function $parent(v)$ as the parent node of v , and use the function $att(v)$ to represent the attribute associated with the leaf node. As the child nodes have an ordering that ranges from 1 to num , we also use a function $index(v)$ to represent the number associated with the node v . Assume that the access tree T has a root $root$, then we denote the T_v as the subtree rooted at the node v . If there exists an attribute set γ satisfying T_v , then we have $T_v(\gamma) = 1$. If v is a non-leaf node, we need to compute $T_{v'}(\gamma)$ for all child nodes $\{v'\}$ of v . $T_v(\gamma)$ returns 1 if and only if at least k_v child nodes return 1. If v is a leaf node, then $T_v(\gamma) = 1$ if and only if $att(v) \in \gamma$.

To model the access permission of time attribute time, we use the multi-dimensional range derivation function [23]. For achieving time-controllable keyword search, each DU is assigned with secret components associated with authorized time range $[t'_i, t'_j]$, and can make valid search queries if his authorized time range is a subset of time intervals (e.g., $[t_i, t_j]$) embedded in the indexes. The *multi-dimensional range derivation function* is shown as follows:

Let $U = \{t_1, \dots, t_{max}\}$ be a countable integer set with total ordering relation, namely $0 \leq t_1 \leq \dots \leq t_{max} \leq Z$, where Z is maximum integer. The order-preserving map $\psi(\cdot)$ outputs the cryptographic value $v_{[t_i, t_j]} = \psi([t_i, t_j]) = \varphi^{\lambda t_i \mu^{Z-t_j}} \in \mathbb{G}_{n'}^*$ when taking the range $[t_i, t_j]$ as input, where $\varphi \in \mathbb{G}_{n'}$, $\lambda, \mu \in \mathbb{Z}_{n'}$. Then we define a multi-dimensional range derivation function $F(\cdot)$ with following two properties:

- *Easy to compute*. $F(\cdot)$ can be efficiently computed. If $[t'_i, t'_j] \subseteq [t_i, t_j]$, we can have $v_{[t'_i, t'_j]} = F(v_{[t_i, t_j]})$ by (1), where $t_i \leq t'_i$ and $t_j \geq t'_j$

$$\begin{aligned} F(v_{[t_i, t_j]}) &= v_{[t_i, t_j]}^{\lambda^{t'_i-t_i} \mu^{t_j-t'_j}} \\ &= (\varphi^{\lambda t_i \mu^{Z-t_j}})^{\lambda^{t'_i-t_i} \mu^{t_j-t'_j}} = v_{[t'_i, t'_j]}, \end{aligned} \quad (1)$$

- *Hard to invert*: It is infeasible for any probabilistic polynomial time algorithm to compute $v_{[t'_i, t'_j]}$ from $v_{[t_i, t_j]}$ if $[t'_i, t'_j] \not\subseteq [t_i, t_j]$ (i.e., $t_i > t'_i$ or $t_j < t'_j$).

C. Threat Model

In our proposed frameworks, TA and DO are all assumed to be fully trusted. Similarly to the existing schemes [14], [41], [42], TS is assumed to be fully trusted to improve system's security with sacrificing a small computation cost. Otherwise, TS may infer some privacy information from accessible time intervals. While CS is honest-but-curious¹, which honestly executes operations according to established agreements but is curious to infer some sensitive information based on statistics or background information. It is worth noting that the use of the two-server framework, consisting of CS and TS, contributes

¹If CS is assumed to be malicious, it may return false search results, and even not correctly execute the search protocol. To ensure the correctness of both query results and the execution of the search protocol, the verification mechanism has been proposed in many existing works such as [43], [44], [45]. Therefore, we remain consistent with most works by assuming that CS is an honest-but-curious entity.

to preventing privacy leakage. If our schemes are implemented under the single server framework, the time index transformation is performed by CS. In this case, CS has the knowledge of accessible time intervals of e-health records, thereby posing a risk of privacy leakage. Malicious² DUs may collude with each other, but CS cannot collude with malicious DUs. The potential threats caused by malicious DUs or CS are demonstrated as follows:

- *Key collusion attacks*: Malicious DUs with different attribute sets may try to collude with others to gain the valid secret key, which incurs unauthorized accesses.
- *Token linkability*: CS may tell whether two submitted tokens are generated from the same query if the token generation algorithm is deterministic.
- *Chosen-keyword attack*: CS may distinguish indexes from ciphertexts of its chosen keywords by using the public information.
- *Keyword secrecy leakage*: CS may encrypt a keyword of its choice, and check whether the resulting ciphertext and received tokens are generated from the same keyword.

D. Security Model

Assume that an adversary $\mathcal{A}$ (modelling CS) does not have any matching token, then it cannot infer any information about underlying keywords. We formalize this security property by using the following Chosen-Keyword Attack (CKA) game which is simulated between $\mathcal{A}$ and a challenger $\mathcal{C}$ as follows:

- *Setup*: Let I_{Enc} be the input of Enc. $\mathcal{A}$ first chooses a non-trivial challenge I_{Enc}^* that can be satisfied by any unauthorized DU, and sends it to $\mathcal{C}$. Then, $\mathcal{C}$ calls Setup to generate the public key pk and master key msk .
- *Phase 1*: $\mathcal{A}$ issues queries to the following oracles for polynomially many times. $\mathcal{C}$ stores a keyword list L_{kw} whose initial value is null.
 - $\mathcal{O}_{KeyGen}$: Let I_{KeyGen} be the input of KeyGen. If I_{KeyGen} matches with I_{Enc}^* , $\mathcal{C}$ aborts this process. Otherwise, $\mathcal{C}$ calls KeyGen to generate the secret key sk for $\mathcal{A}$.
 - $\mathcal{O}_{Token}$: $\mathcal{C}$ first generates the secret key sk with I_{KeyGen} , then calls Token to generate a token tk based on sk and queried keyword kw before returning it to $\mathcal{A}$. If I_{KeyGen} satisfies I_{Enc}^* , $\mathcal{C}$ adds kw into L_{kw} .
- *Challenge*: $\mathcal{A}$ chooses challenging keywords $kw_0, kw_1 \notin L_{kw}$ and sends them to $\mathcal{C}$. $\mathcal{C}$ chooses a random bit $b \in \{0, 1\}$ and calls Enc to compute the index cph_b with inputs I_{Enc}^*, kw_b , then returns cph_b to $\mathcal{A}$.
- *Phase 2*: $\mathcal{A}$ continues to make queries to oracles in Phase 1. While the restriction is that two challenging keywords kw_0, kw_1 cannot be queried in the oracle $\mathcal{O}_{Token}$ if I_{KeyGen} satisfies I_{Enc}^* .
- *Guess*: $\mathcal{A}$ outputs a guess bit $b' \in \{0, 1\}$ and wins the above CKA game if $b' = b$.

²Our schemes are designed based on the Ciphertext Policy ABE (CP-ABE) mechanism. The key collusion attack is a significant concern in CP-ABE, as malicious users try to collude to obtain the valid secret key corresponding to a large attribute set, thereby more gaining unauthorized access. Therefore, in our schemes, we also assume that DUs are malicious entities who can launch key collusion attacks.

Definition 1: Our proposed schemes are secure against CKA if $\mathcal{A}$'s advantage $Adv_{\mathcal{A}} = |Pr[b' = b] - \frac{1}{2}| < \epsilon$ in breaking above security game is negligible.

CS may encrypt a keyword of its choice and match the resulting ciphertext with the tokens submitted by DUs, which attempts to launch keyword guessing attack. We formalize the weak security notion as *keyword secrecy*, which means that CS's probability in inferring the keyword information from its keyword ciphertext and submitted tokens is negligible. The keyword secrecy game conducted between adversary $\mathcal{A}$ and challenger $\mathcal{C}$ is demonstrated as follows:

- *Setup*: $\mathcal{C}$ calls Setup to generate the public key pk and master key msk , then sends pk to $\mathcal{A}$.
- *Phase 1*: $\mathcal{A}$ makes the polynomial time queries to following oracles:
 - $\mathcal{O}_{KeyGen}$: $\mathcal{C}$ calls KeyGen to generate the secret key sk corresponding to I_{KeyGen} submitted by $\mathcal{A}$, then adds I_{KeyGen} in the list L_{KeyGen} .
 - $\mathcal{O}_{Token}$: With the secret key sk , $\mathcal{C}$ calls Token to generate the token tk according to the queried keyword kw submitted by $\mathcal{A}$, then sends tk to $\mathcal{A}$.
- *Challenge*: $\mathcal{A}$ submits a non-trivial I_{Enc}^* to $\mathcal{C}$. $\mathcal{C}$ chooses a keyword kw^* from the keyword space $\mathcal{W}$ and the input I_{KeyGen}^* that satisfies I_{Enc}^* , then calls Enc and Token to generate the index $cph^* \leftarrow Enc(kw^*, I_{Enc}^*)$ and token $tk^* \leftarrow Token(sk, kw^*)$ respectively. Finally, $\mathcal{C}$ sends the tuple (cph^*, tk^*) to $\mathcal{A}$. It is worth noticing that any input $I_{KeyGen} \in L_{KeyGen}$ does not satisfy I_{Enc}^* .
- *Guess*: After guessing q^* different keywords, $\mathcal{A}$ outputs a guess kw' , and wins this security game if $kw' = kw^*$.

Definition 2: If the adversary $\mathcal{A}$ wins the above keyword secrecy game with at most $\frac{1}{|\mathcal{W}| - q^*} + \epsilon$, then our proposed frameworks guarantee the keyword secrecy, where ϵ is a negligible advantage.

In our system model, DUs may combine their attribute sets together to obtain unauthorized access permissions beyond their privileges. At a technical level, our proposed frameworks should resist the key collusion attack. Each DU has been assigned the valid secret key associated with his attributes S (including descriptive attributes $Attrs$ and time attribute $time$), but may attempt to derive new secret key through collusion attacks. We formalize the security via the key collusion attack game between adversary $\mathcal{A}$ and challenger $\mathcal{C}$ as follows:

- *Setup*: $\mathcal{C}$ calls Setup to generate the public key pk for $\mathcal{A}$, and keeps the master key msk .
- *Learning*: $\mathcal{A}$ makes queries to the key generation on behalf of a number of DUs with different attribute sets $\mathbb{S} = \{S_1, \dots, S_U\}$. Then, $\mathcal{C}$ calls KeyGen for generating $(sk_1, \dots, sk_U)$ for $\mathcal{A}$.
- *Challenge*: $\mathcal{C}$ sends a challenge to $\mathcal{A}$ on behalf of a certain DU with an attribute set $S^* \notin \mathbb{S}$.
- *Response*: $\mathcal{A}$ responses the secret key sk^* to $\mathcal{C}$. If sk^* is valid and can bring more access privileges for DU with an attribute set S^* , then $\mathcal{A}$ wins this security game.

Definition 3: If $\mathcal{A}$ just has a negligible advantage in deriving the valid secret key that has more access privileges than that of any collusive DU, then our proposed frameworks can resist the key collusion attack caused by malicious DUs.

IV. PROPOSED TIME-CONTROLLABLE KEYWORD SEARCH FRAMEWORK

In this section, we first introduce the technical overview of our proposed basic and enhanced time-controllable keyword search frameworks, then present their concrete constructions.

A. Technical Overview

To achieve fine-grained keyword search, we utilize the previous Ciphertext-Policy Attribute-Based Keyword Search (ABKS) [7] solution. The algorithms of ABKS are demonstrated as follows:

- **Setup**(1^κ). Given the security parameters λ , TA outputs the Bilinear map parameters $(\mathbb{G}, \mathbb{G}_T, e, g, p)$. Then, TA selects two hash functions $H_1 : \{0, 1\}^* \rightarrow \mathbb{G}$, $H_2 : \{0, 1\}^* \rightarrow \mathbb{Z}_p$ and random elements $a, b, c \in \mathbb{Z}_p$. Finally, TA outputs the public key $pm = (H_1, H_2, g^a, g^b, g^c)$ and master key $mk = (a, b, c)$.
- **KeyGen**($mk, pk, Atts$). TA selects $r \in \mathbb{Z}_p$ and computes $A = g^{(ac-R)/b}$. For each attribute $at_j \in Atts$, TA selects $r_j \in \mathbb{Z}_p$ and computes $A_j = g^r H_1(at_j)^{r_j}$, $B_j = g^{r_j}$. The secret key that corresponds to attribute set $Atts$ is defined as $sk = (Atts, A, \{A_j, B_j\})$.
- **Enc**(kw, T). DO chooses $r_1, r_2 \in \mathbb{Z}_p$ and computes $W = g^{cr_1}$, $W_0 = g^{a(r_1+r_2)} g^{bH_2(kw)r_1}$, $W' = g^{br_2}$. Let r_2 be the secret key shared in the access tree T , DO computes $W_v = g^{q_v(0)}$, $D_v = H_1(att(v))$ for each leaf node v in T . The index for keyword kw is set as $cph = (T, W, W_0, W', \{W_v, D_v\})$.
- **Token**(sk, kw'). DU chooses $s \in \mathbb{Z}_p$ and computes $tok_1 = (g^a g^{bH_2(kw')})^s$, $tok_2 = g^{cs}$, $tok_3 = A^s$. For each $at_j \in Atts$, DU computes $A'_j = A_j^s$, $B'_j = B_j^s$. The token is defined as $tk = (Atts, tok_1, tok_2, tok_3, \{A'_j, B'_j\})$.
- **Search**(cph, tk). If the attribute set $Atts$ does not satisfy the access tree T , CS aborts the process. Otherwise, CS computes $E_v = e(A'_j, W_v) / e(B'_j, D_v) = e(g, g)^{rsq_v(0)}$ for each $at_j \in S$, where S is a subset of $Atts$ and satisfies T . Using the recursive algorithm, CS finally obtains $E_{root} = e(g, g)^{rsr_2}$. Then, CS returns the search results if (2) holds

$$e(W_0, tok_2) \stackrel{?}{=} e(W, tok_1) E_{root} e(tok_3, W'). \quad (2)$$

Achieving time-controllable search. ABKS is deployed in the Bilinear map of prime order, and its secret r_2 is embedded in term $E_{root} = e(g, g)^{rsr_2}$. While our proposed frameworks are deployed in the composite order Bilinear map. Simply using ABKS cannot achieve the time-controllable keyword search. To solve this issue, we split the secret encrypted in indexes into two parts s, t_{time} in our proposed frameworks. In a nutshell, the secret s is embedded in the term $E_{root} = e(g, w)^{rzs}$ that can be obtained when each DU's descriptive attributes $Atts$ satisfy the access policy T . The secret t_{time} is embedded in the term $E_{time} = e(g, w)^{rzt_{time}}$ that can be derived if there exist time cells that fall into time ranges $[t_i, t_j]$, $[t'_i, t'_j]$ specified in index generation and token generation, respectively. Then, we replace the term E_{root} in (2) with $E_{root} \cdot E_{time}$ for achieving time-controllable

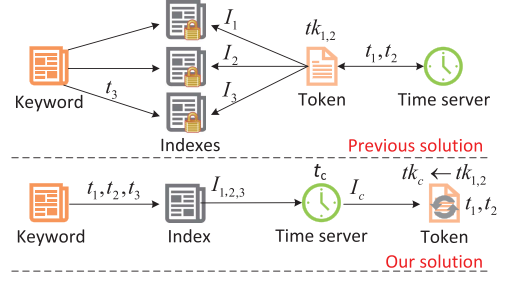


Fig. 3. Example: deriving share of time attribute between previous and our solution.

keyword search. DU can make valid queries iff the current time t_c falls into both $[t_i, t_j]$ and $[t'_i, t'_j]$. It is worth noting that there is usually only one time attribute.

As for the descriptive attributes, it is usually easy to derive $E_{root} = e(g, w)^{rzs}$ by using the access tree shown in Section III-B. For the time attribute, we can use the multi-dimensional range derivation function (shown in Section III-B) to respectively transform the index encrypted by time interval $[t_i, t_j]$ and token encrypted by time range $[t'_i, t'_j]$ into ciphertexts associated with the same time cell $[t_c, t_c]$, where $t_c \in [t_i, t_j]$ and $t_c \in [t'_i, t'_j]$ and the unit of t_c is not restricted to a particular value and can be specified according to the user's requirements. As long as t_c is an integer value, our schemes can be utilized for time access control. Thus, we can easily derive $E_{time} = e(g, w)^{rzt_{time}}$. The difference of deriving E_{time} between previous solution and our frameworks is shown in Fig. 3. The previous solution needs to match $tk_{1,2}$ with I_1, I_2, I_3 respectively, while our solution only matches tk_c with I_c .

Achieving efficient user revocation. To further facilitate user revocation in our enhanced framework, we allow DO to specify a revocation list $List = (id_1, \dots, id_d)$ including at most d DUs. Motivated by the Shamir's secret sharing scheme [46], we generate a d -degree polynomial $P(\cdot)$ and set $P(0)$ as the secret in (3). It is worth noticing that $P(0)$ is shared among n parties, and can be recovered by using the following Lagrange interpolation if there exist at least $d + 1$ secret shares $(P(id_1), \dots, P(id_d), P(id_k))$, where $k \notin [1, d]$. In Enc of our enhanced framework, we embed $P(0)$ into DUs' secret keys. If the identity id_k of queried DU does not belong to $List$, then CS can cancel the term including $P(0)$. Otherwise, CS cannot cancel it, and thus cannot match the submitted token with indexes

$$P(0) = \sum_{i=0}^d \lambda_i P(x_i), \text{ where } \lambda_i = \prod_{j=0, j \neq i}^d \frac{x_j}{x_j - x_i}. \quad (3)$$

B. Construction of Basic Framework (TCKS)

The construction of basic framework (namely TCKS) consists of five algorithms, namely Setup, KeyGen, Enc, Token and Search, which are shown as follows:

Setup (1^κ): Given the security parameter κ , TA first generates the composite order Bilinear parameters $BP = (N = pq, \mathbb{G}, \mathbb{G}_T, e, w, g, \varphi)$, where w, g, φ are generators of groups $\mathbb{G}, \mathbb{G}_s, \mathbb{G}_{n'}$, respectively. Then, TA chooses random elements $\alpha, \beta, c \in \mathbb{Z}_n^*$, $\lambda, \mu \in \mathbb{Z}_{n'}^*$ and hash functions $H_1 : \{0, 1\}^* \rightarrow$

$\mathbb{G}_{s'}, H_2 : \{0, 1\}^* \rightarrow \mathbb{Z}_n^*$. Finally, TA computes w^α, w^β, g^c and outputs the public key pk , master key msk by (4). It is worth noticing that TA also outputs a multi-dimensional range derivation function $F(\cdot)$ shared among TS and authorized DUs

$$\begin{aligned} pk &= (BP, \lambda, \mu, w^\alpha, w^\beta, g^c); \\ msk &= (p, q, n', \alpha, \beta, c). \end{aligned} \quad (4)$$

KeyGen (msk, pk, S): Given the attribute set S including descriptive attributes $Atts = \{at_j\}$ and time attribute time, TA calls **ABKS.KeyGen** to generate the secret key sk for DU, note that time is defined as a time range $[t'_i, t'_j]$. TA first selects an element $r \in \mathbb{Z}_n^*$ and computes D . Then, TA chooses a random element $(r_j, r_{time} \in \mathbb{Z}_n^*)$ for each attribute in S and computes $D_j, D_j^1, D_{time}, D_{time}^1, D_{time}^2$. Finally, TA sends the secret key $sk = (D, \{D_j, D_j^1\}, D_{time}, D_{time}^1, D_{time}^2)$ to DU, which is defined by (5)

$$\begin{aligned} D &= g^{(\alpha c - r)/\beta}; \\ \forall at_j \in Atts : D_j &= g^r H_1(at_j)^{r_j}, D_j^1 = w^{r_j}; \\ D_{time} &= g^r H_1(time)^{r_{time}}, D_{time}^1 = w^{r_{time}}, D_{time}^2 = v_{[t'_i, t'_j]}^{r_{time}}. \end{aligned} \quad (5)$$

Enc (pk, W, R): Given the e-health records $R = \{R_i\}$ and corresponding keyword set $W = \{kw\}$, DO first encrypts each record R_i using the traditional symmetric encryption algorithm such as AES (Advanced Encryption Standard), which is outside the scope of our discussion.

To build indexes for W , DO specifies an access tree T constructed from descriptive attributes $Atts$. DO chooses a polynomial q_v of degree $d_v = k_v - 1$ for each node v in T in top-down manner. For the root node $root$, DO selects an element $s \in \mathbb{Z}_n^*$ and sets $q_{root}(0) = s$, then chooses other d_{root} points to define the polynomial q_{root} . For other non-root node v , DO sets $q_v(0) = q_{parent(v)}(\text{index}(v))$ and also selects other d_v points to completely construct q_v . DO also selects a random element $t_{time} \in \mathbb{Z}_n^*$ for the attribute time with a range $[t_i, t_j]$.

Then, DO selects an element $u \in \mathbb{Z}_n^*$ and calls **ABKS.Enc** to generate the index for each keyword $kw \in W$ as follows:

- DO computes I, I', I_0 for encrypting each keyword kw .
- Let L be the leaf node set in T . For each leaf node $l \in L$, DO computes I_l, I'_l .
- For the attribute time, DO computes $I_{time}, I_{time}^1, I_{time}^2$.

Finally, DO sets the indexes as $cph = (T, \{I_{kw}, C_{time}\})$, and sends index components $(T, \{I_{kw}\})$, $\{C_{time}\}$ to CS and TS, respectively, where $I_{kw} = (I, I', I_0, \{I_l, I'_l\})$ and $C_{time} = (I_{time}, I_{time}^1, I_{time}^2)$ are defined by (6)

$$\begin{aligned} I &= (g^c)^u, I' = (w^\beta)^{(s+t_{time})}, \\ I_0 &= w^{\alpha(u+s+t_{time})} w^{\beta H_2(kw)u}; \\ \forall l \in L : I_l &= w^{q_l(0)}, I'_l = H_1(\text{att}(l))^{q_l(0)}; \\ I_{time} &= w^{t_{time}}, I_{time}^1 = H_1(time)^{t_{time}}, I_{time}^2 = v_{[t_i, t_j]}^{t_{time}}. \end{aligned} \quad (6)$$

Token (kw', sk, pk): When making a query for the keyword kw' at time t_c , DU selects a random element $z \in \mathbb{Z}_n^*$ and then calls **ABKS.Token** to generate the token tk . Specifically, DU first computes the token

components tok_1, tok_2, tok_3 . Then, for each descriptive attribute $at_j \in Atts$ and time attribute time, DU respectively computes $D_j^z, (D_j^1)^z, (D_{time})^z, (D_{time}^1)^z, (D_{time}^2)^z$. Finally, DU submits the query token $tk = (tok_1, tok_2, tok_3, \{D_j^z, (D_j^1)^z\}, (D_{time})^z, (D_{time}^1)^z, (D_{time}^2)^z)$ and his attribute set S to CS, note that tk is computed by (7). If DU embeds fake time t'_c in the query token tk , the time attribute match will not succeed and access will fail, this is because DU cannot compromise the fully trusted TS to transform the time interval $[t_i, t_j]$ into the time cell $[t'_c, t'_c]$. Thus, there is no motivation for DU to embed fake time into his query token

$$\begin{aligned} tok_1 &= (w^\alpha w^{\beta H_2(kw')})^z, tok_2 = g^{cz}, \\ tok_3 &= D^z = g^{z(\alpha c - r)/\beta}; \\ \forall at_j \in Atts : D_j^z &= g^{zr} H_1(at_j)^{zr_j}, (D_j^1)^z = w^{zr_j}; \\ (D_{time})^z &= g^{zr} H_1(time)^{zr_{time}}, (D_{time}^1)^z = w^{zr_{time}}, \\ (D_{time}^2)^z &= F(v_{[t'_i, t'_j]}^{zr_{time}}) = v_{[t_c, t_c]}^{zr_{time}}. \end{aligned} \quad (7)$$

Search ($tk, \{I_{kw}\}, C_{time}^*, pk, S, T$): Upon receiving the query token tk , CS first interacts with TS to obtain the time indexes at time t_c . TS computes $I_{time}^* = I_{time} \cdot F(I_{time}^2) = (w \cdot v_{[t_c, t_c]})^{t_{time}}$ and returns transformed time indexes $C_{time}^* = (I_{time}^1, I_{time}^*)$ to CS. Then, CS matches tk with index components I_{kw}, C_{time}^* as follows:

- CS checks whether $Atts$ satisfies T . If $Atts$ does not satisfy T , CS aborts this process. Otherwise, CS chooses a subset $Atts^*$ of $Atts$, note that $Atts^*$ satisfies T . If the node v is a leaf node and $\text{att}(v) = at_j \in Atts^*$, CS calls **ABKS.Search** to compute E_v by (8). If v is a non-leaf node, CS calls the following recursive algorithm to compute $e(g, w)^{r_z q_v(0)}$. Specifically, Let S_v be k_v -sized child nodes (e.g., node v') of v , CS computes $e(g, w)^{r_z q_v(0)} = \prod_{v' \in S_v} e(g, w)^{r_z q_{v'}(0) \Delta_{i, S'_v}(0)}$, where $i = \text{index}(v')$, $S'_v = \{\text{index}(v') | v' \in S_v\}$, $\Delta_{i, S'_v}(0) = \prod_{j \in S'_v, j \neq i} \frac{i}{i-j}$. Thus, CS can derive $E_{root} = e(g, w)^{r_z s}$

$$E_v = \frac{e(D_{att(v)}^z, I_v)}{e((D_{att(v)}^1)^z, I'_v)} = e(g, w)^{r_z q_v(0)}. \quad (8)$$

- For the time attribute, if $t_c \in [t_i, t_j] \wedge t_c \in [t'_i, t'_j]$, CS can compute E_{time} by (9)

$$E_{time} = \frac{e((D_{time})^z, I_{time}^*)}{e((D_{time}^1)^z, I_{time}^1)} = e(g, w)^{r_z t_{time}}. \quad (9)$$

- CS checks whether tk matches with the index for keyword kw by (10). If (10) holds, CS returns the query results to DU. Otherwise, CS returns $\perp$

$$e(I_0, tok_2) \stackrel{?}{=} e(I, tok_1) \cdot E_{root} \cdot E_{time} \cdot e(I', tok_3). \quad (10)$$

Remark: Our basic framework achieves the time-controllable keyword search solution, in which DO builds indexes in a specified time interval $[t_i, t_j]$. DU can make valid queries iff the current time t_c falls into the time interval $[t_i, t_j]$ and time range $[t'_i, t'_j]$ embedded in indexes and token, respectively. If DO wants to change the time interval $[t_i, t_j]$ to $[\tilde{t}_i, \tilde{t}_j]$, DO first

finds the element t_{time} corresponding to $[t_i, t_j]$, then computes $\tilde{I}_{\text{time}}^2 = v_{[t_i, t_j]}^{t_{\text{time}}}$ and sends it to TS. After that, TS replaces I_{time}^2 with $\tilde{I}_{\text{time}}^2$. It is worth noticing that each update of a time information only requires one modular exponentiation operation and does not incur high update overheads in dynamic setting. Our basic framework can support many functionalities such as spatial keyword search and range query. However, in practical applications, DUs may leave the system frequently. For security concerns, the revoked DUs should not make valid search queries using their old or updated secret keys. The traditional user revocation and attribute revocation mechanisms need to respectively regenerate secret keys and indexes, which incur high computation burdens on DUs and DOs, respectively. Thus, a challenging issue is how to achieve user revocation without updating ciphertexts frequently.

C. Construction of Enhanced Framework (TCKS+)

The construction of enhanced framework (namely TCKS+) also consists of five algorithms, namely Setup, KeyGen, Enc, Token and Search. The Setup of TCKS+ is almost same as that of TCKS, but the difference is that TA in TCKS+ generates a d -degree polynomial $P(\cdot)$. For achieving user revocation, $P(\cdot)$ is also shared with DO. The other algorithms of TCKS+ are shown as follows:

KeyGen (msk, pk, S): Given the attribute set S including descriptive attributes $\text{Atts} = \{\text{at}_j\}$ and time attribute time, TA calls ABKS.KeyGen to generate the secret key sk for DU with identity id_k , note that time is defined as a time range $[t'_i, t'_j]$. TA first selects an element $r \in \mathbb{Z}_n^*$ and computes D . Then, TA chooses a random element $(r_j, r_{\text{time}} \in \mathbb{Z}_n^*)$ for each attribute in S and computes $D_j, D_j^1, D_j^2, D_{\text{time}}, D_{\text{time}}^1, D_{\text{time}}^2, D_{\text{time}}^3$. Finally, TA sends the secret key $\text{sk} = (D, \{D_j, D_j^1, D_j^2\}, D_{\text{time}}, D_{\text{time}}^1, D_{\text{time}}^2, D_{\text{time}}^3, D_{\text{time}}^4)$ to DU, which is defined by (11), where $P(0)$ is a secret shared among n parties

$$\begin{aligned} D &= g^{(\alpha c - r)/\beta}; \\ \forall \text{at}_j \in \text{Atts} : D_j &= g^r H_1(\text{at}_j)^{r_j P(0)}, \\ D_j^1 &= w^{r_j}, D_j^2 = w^{r_j P(id_k)}; \\ D_{\text{time}} &= g^r H_1(\text{time})^{r_{\text{time}} P(0)}, D_{\text{time}}^1 = w^{r_{\text{time}}}, \\ D_{\text{time}}^2 &= w^{r_{\text{time}} P(id_k)}, D_{\text{time}}^3 = v_{[t'_i, t'_j]}^{r_{\text{time}}}, \\ D_{\text{time}}^4 &= v_{[t'_i, t'_j]}^{r_{\text{time}} P(id_k)}. \end{aligned} \quad (11)$$

Enc (pk, W, R): Given the e-health records $R = \{R_i\}$ and corresponding keyword set $W = \{kw\}$, DO first encrypts each record R_i using the traditional symmetric encryption algorithm such as AES (Advanced Encryption Standard), which is outside the scope of our discussion.

To build indexes for W, DO specifies an access tree T constructed from descriptive attributes. DO chooses a polynomial q_v of degree $d_v = k_v - 1$ for each node v in T in top-down manner. For the root node $root$, DO selects an element $s \in \mathbb{Z}_n^*$ and sets $q_{root}(0) = s$, then chooses other d_{root} points to define

the polynomial q_{root} . For other non-root node v , DO sets $q_v(0) = q_{\text{parent}(v)}(\text{index}(v))$ and also selects other d_v points to completely construct q_v . DO selects a random element $t_{\text{time}} \in \mathbb{Z}_n^*$ for the attribute time with a range $[t_i, t_j]$. Besides, DO specifies a user revocation list $\text{List} = (id_1, \dots, id_d)$ including at most d revoked DUs.

Then, DO selects an element $u \in \mathbb{Z}_n^*$ and calls ABKS.Enc to generate the index for each keyword $kw \in W$ as follows:

- DO computes I, I', I_0 for encrypting each keyword kw .
- Let L be the leaf node set in T. For each leaf node $l \in L$, DO computes I_l, I'_l, I''_l .
- For the attribute time, DO computes $I_{\text{time}}, I_{\text{time}}^1, I_{\text{time}}^2$.

Finally, DO sets the indexes as $\text{cph} = (T, \{I_{kw}, C_{\text{time}}\})$, and sends index components $(T, \{I_{kw}\})$, $\{C_{\text{time}}\}$ to CS and TS, respectively, where $I_{kw} = (I, I', I_0, \{I_l, I'_l\})$ and $C_{\text{time}} = (I_{\text{time}}, I_{\text{time}}^1, I_{\text{time}}^2)$ are defined by (12)

$$\begin{aligned} I &= (g^c)^u, I' = (w^\beta)^{(s+t_{\text{time}})}, \\ I_0 &= w^{\alpha(u+s+t_{\text{time}})} w^{\beta H_2(kw)u}, \\ \forall l \in L : I_l &= w^{q_l(0)}, I'_l = H_1(\text{att}(l))^{q_l(0)}, \\ I''_l &= (I'_l)^{\sum_{i=1}^d \lambda_i P(id_i)}, \\ I_{\text{time}} &= w^{t_{\text{time}}}, I_{\text{time}}^1 = H_1(\text{time})^{t_{\text{time}}}, \\ I_{\text{time}}^2 &= (I_{\text{time}}^1)^{\sum_{i=1}^d \lambda_i P(id_i)}, I_{\text{time}}^3 = v_{[t_i, t_j]}^{t_{\text{time}}}. \end{aligned} \quad (12)$$

Token ($kw', \text{sk}, \text{pk}$): When making a query for the keyword kw' at time t_c , DU selects a random element $z \in \mathbb{Z}_n^*$ and then calls ABKS.Token to generate the token tk. Specifically, DU first computes the token components $\text{tok}_1, \text{tok}_2, \text{tok}_3$. For each descriptive attribute $\text{at}_j \in \text{Atts}$ and time attribute time, DU respectively computes $D_j^z, (D_j^1)^z, (D_{\text{time}})^z, (D_{\text{time}}^1)^z, (D_{\text{time}}^2)^z$. DU submits the query token $\text{tk} = (\text{tok}_1, \text{tok}_2, \text{tok}_3, \{D_j^z, (D_j^1)^z\}, (D_{\text{time}})^z, (D_{\text{time}}^1)^z, (D_{\text{time}}^2)^z)$ and his attribute set S to CS, which is computed by (13)

$$\begin{aligned} \text{tok}_1 &= (w^\alpha w^{\beta H_2(kw')})^z, \text{tok}_2 = g^{cz}, \\ \text{tok}_3 &= D^z = g^{z(\alpha c - r)/\beta}, \\ \forall \text{at}_j \in \text{Atts} : D_j^z &= g^{zr} H_1(\text{at}_j)^{zr_j P(0)}, \\ (D_j^1)^z &= w^{zr_j}, (D_j^2)^z = w^{zr_j P(id_k)}; \\ (D_{\text{time}})^z &= g^{zr} H_1(\text{time})^{zr_{\text{time}}}, (D_{\text{time}}^1)^z = w^{zr_{\text{time}}}, \\ (D_{\text{time}}^2)^z &= w^{zr_{\text{time}} P(id_k)}, \\ (D_{\text{time}}^3)^z &= F(v_{[t'_i, t'_j]}^{zr_{\text{time}}}) = v_{[t_c, t_c]}^{zr_{\text{time}}}; \\ (D_{\text{time}}^4)^z &= F(v_{[t'_i, t'_j]}^{zr_{\text{time}} P(id_k)}) = v_{[t_c, t_c]}^{zr_{\text{time}} P(id_k)}. \end{aligned} \quad (13)$$

Search ($\text{tk}, \{I_{kw}\}, C_{\text{time}}^*, \text{pk}, S, T$): Upon receiving the query token tk, CS first interacts with TS to obtain the time indexes at time t_c . TS computes I_{time}^* by (14), and returns transformed time indexes $C_{\text{time}}^* = (I_{\text{time}}^1, I_{\text{time}}^2)$ to CS

$$I_{\text{time}}^* = I_{\text{time}} \cdot F(I_{\text{time}}^3) = (w \cdot v_{[t_c, t_c]})^{t_{\text{time}}}, \quad (14)$$

Then, CS matches tk with index components $I_{kw}, C_{\text{time}}^*$ as follows:

- CS checks whether Atts satisfies T: If Atts does not satisfy T, CS aborts this process. Otherwise, CS chooses a subset Atts^* of Atts, note that Atts^* satisfies T. If the node v is a leaf node and $\text{att}(v) = \text{at}_j \in \text{Atts}^*$, CS calls ABKS.Search to compute E_v by (15). It is worth noticing that λ_k can be computed by (3), where the set is $\{id_1, \dots, id_d, id_k\}$. If v is a non-leaf node, CS calls the following recursive algorithm to compute $e(g, w)^{rzq_v(0)}$. Specifically, Let S_v be k_v -sized child nodes (e.g., node v') of v , CS computes $e(g, w)^{rzq_v(0)} = \prod_{v' \in S_v} e(g, w)^{rzq_{v'}(0) \Delta_{i, S_v'}(0)}$, where $i = \text{index}(v')$, $S_v' = \{\text{index}(v') | v' \in S_v\}$, $\Delta_{i, S_v'}(0) = \prod_{j \in S_v', j \neq i} \frac{i}{i-j}$. Thus, CS can derive $E_{\text{root}} = e(g, w)^{rzs}$

$$E_v = \frac{e(D_{\text{att}(v)}^z, I_v)}{e((D'_{\text{att}(v)})^z, I_v^{\lambda_k}) e((D'_{\text{att}(v)})^z, I_v')} = e(g, w)^{rzq_v(0)}. \quad (15)$$

- For the time attribute, if $t_c \in [t_i, t_j] \wedge t_c \in [t'_i, t'_j]$, CS can compute E_{time} as follows:

$$E_{\text{time}} = \frac{e((D_{\text{time}})^z, I_{\text{time}}^*) e((D_{\text{time}}^1)^z, (D_{\text{time}}^3)^z, I_{\text{time}}^2)^{-1}}{e((D_{\text{time}}^2)^z, (D_{\text{time}}^4)^z, I_{\text{time}}^1)^{\lambda_k}} = e(g, w)^{rzt_{\text{time}}}$$

- CS checks whether tk matches with the index for keyword kw by (16). If (16) holds, CS returns the query results to DU. Otherwise, CS returns $\perp$

$$e(I_0, \text{tok}_2) \stackrel{?}{=} e(I, \text{tok}_1) \cdot E_{\text{root}} \cdot E_{\text{time}} \cdot e(I', \text{tok}_3). \quad (16)$$

Remark: To support efficient user revocation, our enhanced framework uses the secret sharing technique to construct a d -degree polynomial. We embed the secret $P(0)$ in the secret key of each DU, which will be further hidden in the token. In the index generation process, DO specifies a user revocation list List including at most d revoked DUs, then embeds their shares $\{P(id_1), \dots, P(id_d)\}$ into the indexes. If the identity id_k of query user DU belongs to List, there are at most d secret shares that cannot be used to recover $P(0)$. Otherwise, the $d + 1$ secret shares can recover $P(0)$. Thus, our enhanced framework can prevent revoked DUs to make valid queries. In addition, our enhanced framework does not incur high computation and storage overheads while supporting user revocation, which can be verified by the experimental tests in Section VI. In addition, similar to most encrypted keyword search schemes, our proposed schemes disclose access pattern (i.e., CS learns the matching results between e-health records and queries), which can be obfuscated by using differential privacy [47], [48], [49] to add both false positives and false negatives to the search results.

V. SECURITY ANALYSIS

The security of our proposed frameworks requires that CS should learn nothing about keywords from indexes apart from

search results, which can be guaranteed by the following theorem.

Theorem 1: Given the hash functions H_1, H_2 , our proposed frameworks are secure against CKA in the generic bilinear group model, where H_1 is modelled as a random oracle and H_2 is a one-way hash function.

Proof: In the CKA security game, the adversary $\mathcal{A}$ wants to distinguish $w^{\alpha(u+s+t_{\text{time}})} w^{u\beta H_2(kw_0)}$ from $w^{\alpha(u+s+t_{\text{time}})} w^{u\beta H_2(kw_1)}$. Let $\theta \in \mathbb{Z}_n^*$ be a random element, then $\mathcal{A}$'s advantage of telling w^θ from $w^{\alpha(u+s+t_{\text{time}})} w^{u\beta H_2(kw_0)}$ is equal to that of telling w^θ from $w^{\alpha(u+s+t_{\text{time}})} w^{u\beta H_2(kw_1)}$. Suppose $\mathcal{A}$ has an advantage ϵ in successfully breaking the CKA security game, then it has an advantage $\epsilon/2$ in telling w^θ from $w^{\alpha(u+s+t_{\text{time}})} w^{u\beta H_2(kw_0)}$. The CKA game simulated between $\mathcal{A}$ and challenger $\mathcal{C}$ is shown as follows:

- *Setup:* $\mathcal{C}$ selects random elements $\alpha, \beta, c \in \mathbb{Z}_n^*$ and calls **Setup** to generate the public key $\text{pk} = (\text{BP}, w^\alpha, w^\beta, g^c)$ for $\mathcal{A}$, while it keeps the master key msk . Then, $\mathcal{A}$ chooses an access tree T^* generated from descriptive attributes and sends it to $\mathcal{C}$. $\mathcal{C}$ simulates the random oracle model $\mathcal{O}_{H_1}$ as follows: 1) If the descriptive attribute at_j has been queried before, it selects $\alpha_j \in \mathbb{Z}_n^*$, then adds the tuple (at_j, α_j) in the hash list L_{H_1} and returns w^{α_j} . Otherwise, it returns w^{α_j} by using the term α_j in L_{H_1} .
- *Phase 1:* $\mathcal{A}$ makes queries to the oracles $\mathcal{O}_{\text{KeyGen}}, \mathcal{O}_{\text{Token}}$, then $\mathcal{C}$ responses as follows:
 - $\mathcal{O}_{\text{KeyGen}}$: $\mathcal{C}$ first chooses an element $r \in \mathbb{Z}_n^*$ and computes $D = g^{(\alpha c - r)/\beta}$. Then, for each attribute $\text{at}_j \in \text{Atts}$, $\mathcal{C}$ selects $r_j \in \mathbb{Z}_n^*$ and computes $D_j = g^r g^{\alpha_j r_j}$, $D_j^1 = w^{r_j}$. Finally, $\mathcal{C}$ sends the secret key components $(\text{Atts}, D, \{D_j, D_j^1\})$ to $\mathcal{A}$.
 - $\mathcal{O}_{\text{Token}}$: $\mathcal{C}$ first makes the query to $\mathcal{O}_{\text{KeyGen}}$ for getting the secret key components $(\text{Atts}, D, \{D_j, D_j^1\})$, then selects $z \in \mathbb{Z}_n^*$ and calls **Token** to generate the token components $\text{tok}_1, \text{tok}_2, \text{tok}_3, \{D_j^z, (D_j^1)^z\}$, where $\text{tok}_1 = (w^\alpha w^{\beta H_2(kw')})^z$, $\text{tok}_2 = g^{cz}$, $\text{tok}_3 = D^z$. If Atts satisfies T^* , $\mathcal{C}$ adds the keyword kw' into L_{kw} .
- *Challenge:* $\mathcal{A}$ submits two challenging keywords $kw_0, kw_1 \notin \text{L}_{kw}$ with equal length to $\mathcal{C}$. $\mathcal{C}$ selects random elements $s, u, t_{\text{time}} \in \mathbb{Z}_n^*$ and calculates the secret shares of s for each leaf node $l \in \text{L}$ in T^* . Then, $\mathcal{C}$ selects a random bit $b \in \{0, 1\}$. If $b = 0$, $\mathcal{C}$ computes $I = g^{cu}$, $I' = w^{\beta(s+t_{\text{time}})}$, $I_0 = w^\theta$, $I_l = w^{q_l(0)}$, $I_l' = g^{\alpha_j q_l(0)}$, where $\text{att}(l) = \text{at}_j$. If $b = 1$, $\mathcal{C}$ computes $I = g^{cu}$, $I' = w^{\beta(s+t_{\text{time}})}$, $I_0 = w^{\alpha(s+u+t_{\text{time}})}$, $I_l = w^{q_l(0)}$, $I_l' = g^{\alpha_j q_l(0)}$, where $\text{att}(l) = \text{at}_j$. Finally, $\mathcal{C}$ sends $(I, I', I_0, \{I_l, I_l'\})$ to $\mathcal{A}$.
- *Phase 2:* This phase is similar to that of *Phase 1*, but one restriction is that kw_0, kw_1 cannot be queried again if Atts satisfies T^* .
- *Guess:* $\mathcal{A}$ outputs a guess bit $b' \in \{0, 1\}$ and wins this security game if $b' = b$.

Suppose $\mathcal{A}$ can construct the term $e(g, w)^{\delta\alpha(s+u+t_{\text{time}})}$ for g^δ that can be composed of oracle outputs, then $\mathcal{A}$ utilizes the term to tell w^θ from $w^{\alpha(s+u+t_{\text{time}})}$. Next, we should show that $\mathcal{A}$ just has a negligible advantage in

constructing the term $e(g, w)^{\delta\alpha(s+u+t_{\text{time}})}$ for g^δ , which implies that $\mathcal{A}$ cannot break the CKA security game with non-negligible advantage.

We first recall the generic group model [10], in which two random injective functions ϕ_0, ϕ_1 that map elements from $\mathbb{Z}_n$ into elements with length $\{0, 1\}^\iota$, where $\iota > 3 \log(n)$. It is worth noticing that $\mathcal{A}$ can guess an element in the image of ϕ_0 or ϕ_1 with a negligible advantage. Given the groups $\mathbb{G} = \{\phi_0(x) | x \in \mathbb{Z}_n\}$, $\mathbb{G}_T = \{\phi_1(x) | x \in \mathbb{Z}_n\}$, we show how to construct $e(g, w)^{\delta\alpha(u+s+t_{\text{time}})}$ for the term δ in the following. As the term u only appears in the term cu , δ should contain c for constructing $e(g, w)^{\delta\alpha(s+u+t_{\text{time}})}$. Let $\delta = c\delta'$, then $\mathcal{A}$ needs to construct the term $\delta'\alpha c(s+t_{\text{time}})$ for constructing $e(g, w)^{c\delta'\alpha(s+u+t_{\text{time}})}$. The term $\delta'\alpha c(s+t_{\text{time}})$ can be constructed by using $\beta(s+t_{\text{time}})$ and $(\alpha c - r)/\beta$. Due to $\beta(s+t_{\text{time}}) \cdot (\alpha c - r)/\beta = (s+t_{\text{time}})(\alpha c - r)$, $\mathcal{A}$ still needs to cancel $r(s+t_{\text{time}})$ by using the terms $\alpha_j, r + \alpha_j r_j, q_l(0), \alpha_j q_l(0)$. However, the term $r(s+t_{\text{time}})$ can be constructed iff attributes corresponding $r_j, r + \alpha_j r_j$ match with T^* . Thus, $\mathcal{A}$ just has a negligible advantage in breaking the above CKA security game. In other words, our proposed frameworks are secure against CKA in the generic bilinear group model. This completes the proof of Theorem 1. $\square$

To prevent the adversary $\mathcal{A}$ (modelling CS) from matching tokens with keyword ciphertext of its choice, our proposed frameworks should guarantee the keyword secrecy, which can be proved by the following theorem.

Theorem 2: Given the one-way hash function H_2 , our proposed frameworks achieve the keyword secrecy in the random oracle model.

Proof: We first construct a challenger $\mathcal{C}$ that can simulate the keyword secrecy game by interacting with $\mathcal{A}$:

- **Setup:** $\mathcal{C}$ selects several random elements $\alpha, \beta, c \in \mathbb{Z}_n^*$ and calls Setup to generate the public key $\text{pk} = (\text{BP}, \lambda, \mu, w^\alpha, w^\beta, g^c)$ and master key $\text{msk} = (p, q, n', \alpha, \beta, c)$. Similar to the proof of Theorem 1, $\mathcal{C}$ simulates the random oracle model $\mathcal{O}_{H_1}$ as follows: If the descriptive attribute at_j was not queried before, $\mathcal{C}$ selects an element $\alpha_j \in \mathbb{Z}_n^*$ and adds the tuple (at_j, α_j) into the hash list L_{H_1} , and outputs w^{α_j} . Otherwise, $\mathcal{C}$ uses the term α_j that already exists in L_{H_1} , and then outputs w^{α_j} .
- **Phase 1:** $\mathcal{A}$ adaptively makes polynomial time queries to following oracles:
 - $\mathcal{O}_{\text{KeyGen}}$: $\mathcal{C}$ calls KeyGen to generate the secret key sk corresponding to descriptive attributes Atts , then returns sk to $\mathcal{A}$. Finally, $\mathcal{C}$ adds Atts into the list L_{KeyGen} which is initially empty.
 - $\mathcal{O}_{\text{Token}}$: Upon gaining the secret key components $\text{sk} = (D, \{D_j, D_j^1\})$, $\mathcal{C}$ calls Token to generate the token tk for the keyword kw , then sends tk to $\mathcal{A}$.
- **Challenge:** $\mathcal{A}$ submits an access policy T^* to $\mathcal{C}$. $\mathcal{C}$ chooses an attribute set Atts^* that satisfies T^* , then calls KeyGen to generate the secret key sk^* corresponding to Atts^* . With the tuple $(\text{Atts}^*, \text{sk}^*)$, $\mathcal{C}$ selects a keyword kw^* from the keyword space $\mathcal{W}$, and calls Enc and Token to output the indexes cph^* and token tk^* respectively. $\mathcal{C}$ sends $(\text{cph}^*, \text{tk}^*)$ to $\mathcal{A}$.

- **Guess:** $\mathcal{A}$ guesses a keyword kw' and submits it to $\mathcal{C}$. $\mathcal{C}$ calls Enc to generate the indexes cph' corresponding to kw' and returns it to $\mathcal{A}$. If cph' satisfies the token tk^* , then $\mathcal{A}$ wins this security game; otherwise, it fails.

Assume that $\mathcal{A}$ has queried q^* different keywords before guessing kw' , then the size of remaining keyword space is $|\mathcal{W}| - q^*$. In addition, H_2 is a one-way hash function, which indicates the advantage ϵ in deducing the keyword kw^* from the hash value $H_2(kw^*)$ is negligible. Thus, we derive that the $\mathcal{A}$'s probability of winning the above security game is at most $\frac{1}{|\mathcal{W}| - q^*} + \epsilon$. In other words, our proposed frameworks guarantee the keyword secrecy in the random oracle model. This completes the proof of Theorem 2. $\square$

Similar to the proof of ciphertext-policy ABE [10], our proposed frameworks also can prevent malicious DUs to combine their descriptive attributes to derive new valid secret key component. At a high level, our proposed frameworks embed the share $s \in \mathbb{Z}_n^*$ associated with descriptive attributes into the term $e(g, w)^{rzs}$. To generate the valid token that satisfies the indexes, the adversary must recover $e(g, w)^{rzs}$. However, TA uses a different element $r \in \mathbb{Z}_n^*$ for generating the secret key of DU. Thus, malicious DUs cannot derive the valid secret key component by using malicious DUs' descriptive attribute sets. Besides, our proposed frameworks embed the other share $t_{\text{time}} \in \mathbb{Z}_n^*$ in $e(g, w)^{rzt_{\text{time}}}$. Next, we just show that collusion users cannot derive the valid key component for time attribute time.

Theorem 3: Our proposed schemes are secure against the key collusion attack if following two lemmas hold.

Proof: As the key collusion attack against descriptive attributes has been proved in previous schemes, we mainly focus on the key collusion attack against time attribute. Given two DUs' secret key components $(D, D_{\text{time}}^u, D_{\text{time}}^{u,1}, D_{\text{time}}^{u,2}), (D, D_{\text{time}}^u, D_{\text{time}}^{u,1}, D_{\text{time}}^{u,2})$ corresponding to time attribute ranges $[t_i, t_j], [t_i, t_j]$, where $t_i < t_i' < t_j' < t_j$. Suppose the user u' with time range $\text{range}[t_i', t_j']$ attempts to obtain the secret key component of another user u with time range $[t_i, t_j]$. After issuing the key collusion attack, the user u' obtains the new key component $(g^{(\alpha c - r_{u'})/\beta}, g^{r_{u'} H_1(\text{time})^{r_{\text{time}}}}, w^{r_{\text{time}}}, v_{[t_i, t_j]}^{r_{\text{time}}})$ by replacing the term $g^{r_{u'}}$ or $H_1(\text{time})^{r_{\text{time}}}$ with other known terms. The following lemma shows that our proposed frameworks resist such key collusion attack.

Before presenting the proof of Lemma 1, we first give the definition of co-CDH assumption. Given a quadruple (g_1, g_1^a, g_2, g_2^b) , the goal of co-CDH problem is to compute g_2^{ab} , where $a, b \in \mathbb{Z}_n^*$. We can say that the co-CDH assumption holds if the adversary's advantage in solving the co-CDH problem is negligible.

Lemma 1: There exists no probability polynomial time adversary $\mathcal{A}$ that can derive the term g^{r_u} or $H_1(\text{time})^{r_{\text{time}}}$ with a non-negligible advantage under the co-CDH (Computational Diffie-Hellman) assumption [50].

Proof: Considering that w is the generator of group $\mathbb{G}$, we can set $g^{r_u} = w^{a_1}$, $H_1(\text{time}) = w^{b_1}$, $v_{[t_i, t_j]} = w^{c_1}$, $g^{(\alpha c - r_{u'})/\beta} = w^{d_1}$, where $a_1, b_1, c_1, d_1 \in \mathbb{Z}_n^*$. Then, the co-CDH problem is equivalent to deriving $(w^{a_1}, w^{b_1 r_{\text{time}}})$ from the quadruple

$(w^{d_1}, w^{a_1+b_1r_{\text{time}}^u}, w^{c_1r_{\text{time}}^u}, w^{r_{\text{time}}^u})$. As the element c_1 has nothing to do with co-CDH problem, we reduce the co-CDH problem to deriving $(w^{a_1}, w^{b_1r_{\text{time}}^u})$ from $(w, w^{r_{\text{time}}^u}, w^{b_1}, w^{a_1+b_1r_{\text{time}}^u})$. Assume that the adversary $\mathcal{A}$ can solve the above problem. Given the instance (g_1, g_1^a, g_2, g_2^b) of co-CDH problem, another algorithm $\mathcal{B}$ uses $\mathcal{A}$ as a subroutine to solve the co-CDH problem as follows:

- $\mathcal{B}$ first invokes $\mathcal{A}$ with the input $(w = g_1, w^{r_{\text{time}}^u} = g_1^a, w^{b_1} = g_2^b, w^{a_1+b_1r_{\text{time}}^u} = g_2^\nu)$, where $\nu \in \mathbb{Z}_n^*$.
- Suppose the output of $\mathcal{A}$ is (X, Y) , then $\mathcal{B}$ checks whether both $X \cdot Y = g_2^\nu$ and $e(g_1, Y) = e(g_1^a, g_2^b)$. If these equations hold, $\mathcal{B}$ outputs Y . Otherwise, $\mathcal{B}$ continues to invoke $\mathcal{A}$.

If there exist elements such that $r_{\text{time}}^u = a$, $Y = w^{b_1r_{\text{time}}^u}$, then we have $X \cdot Y = g_2^{a_1+b_1r_{\text{time}}^u} = g_2^\nu$, $e(g_1, Y) = e(g_1, g_2^{b_1r_{\text{time}}^u}) = e(g_1^a, g_2^b)$. In this case, $\mathcal{B}$ can address the co-CDH problem, which conflicts the co-CDH assumption. Thus, there exists no algorithm that can derive the term $g^{r_{\text{time}}^u}$ or $H_1(\text{time})^{r_{\text{time}}^u}$ with non-negligible advantage. $\square$

In addition, the collusion users can derive the new secret key $(g^{(\alpha c - r_{\text{time}}^u)/\beta}, g^{r_{\text{time}}^u} H_1(\text{time})^{r_{\text{time}}^u}, w^{r_{\text{time}}^u}, v_{[t_i, t_j]}^{r_{\text{time}}^u})$ by replacing the term $v_{[t_i', t_j']}$ with $v_{[t_i, t_j]}$. Then we prove that our proposed frameworks can resist this kind of collusion attack based on the following theorem.

For proving the Lemma 2, we present the RSA assumption [51]. Let κ be a security parameter, and N be the product of two primes p, q , where $p+1 = 2s_1p'$, $q+1 = 2s_1q'$. In addition, let e be a random prime between $[2^\kappa, 2^{\kappa+1}-1]$, and QR_N be a group of quadratic residues in $\mathbb{Z}_N^*$ of order $p'q'$. Given a tuple (N, e) and a random element $C \in QR_N$, it is difficult to compute the message M such that $M^e \equiv C \pmod{N}$.

Lemma 2: Given the public parameters $(N = pq, \varphi, \lambda, \mu, [t_i, t_j], v_{[t_i, t_j]}^{r_{\text{time}}^u} = \varphi^{r_{\text{time}}^u \lambda^{t_i} \mu^{Z-t_j}})$, there is no algorithm $\mathcal{A}$ that can derive $\varphi^{r_{\text{time}}^u \lambda^{t_i'} \mu^{Z-t_j'}}$ with non-negligible advantage if $t_i > t_i'$ or $t_j < t_j'$ under the RSA assumption.

Proof: For the time attribute time with time range $[t_i, t_j]$, we assume that there exists $t_i > t_i'$. In addition, there exists an algorithm that can output $\varphi^{r_{\text{time}}^u \lambda^{t_i'} \mu^{Z-t_j'}}$ by using the input $(N, \varphi, \lambda, \mu, [t_i, t_j], v_{[t_i, t_j]}^{r_{\text{time}}^u} = \varphi^{r_{\text{time}}^u \lambda^{t_i} \mu^{Z-t_j}})$. Then we construct an algorithm $\mathcal{B}$ than can derive the plaintext $M = C^{e^{-1}}$, where C is the RSA ciphertext and $1 < e < (p-1)(q-1)$ is the public key of RSA. Then we show how $\mathcal{B}$ breaks the bilinear co-CDH problem using the public keys $(\mathbb{G}, N, e)$.

- $\mathcal{B}$ calls $\mathcal{A}$ with the input $(N, \varphi, e = \lambda', t_i, C = \varphi^{r_{\text{time}}^u \lambda^{t_i} \mu^{Z-t_j}})$, where $\varphi = C^{r_{\text{time}}^u}$ and t_i is an integer.
- $\mathcal{A}$'s outputs are set as (t_i', Y) . If the equations $t_i' - t_i - 1 > 0$ and $Y^{\lambda^{(t_i' - t_i)}} = C$ both hold, $\mathcal{B}$ outputs $M = Y^{e^{(t_i' - t_i - 1)}}$. Otherwise, $\mathcal{B}$ continues to invoke $\mathcal{A}$.

The output of $\mathcal{B}$ is valid due to $M^e = C = Y^{e^{(t_i' - t_i - 1)}}$. In addition, it is difficult to compute $r_{\text{time}} = (r_{\text{time}}^u \lambda^{t_i})^{-1} \pmod{N'}$ due to the hardness of factoring N . From the above discussion, $\mathcal{B}$ can solve the RSA problem if $\mathcal{A}$ is a probabilistic polynomial time algorithm, which conflicts the RSA assumption. With the same analysis, it is also difficult to derive

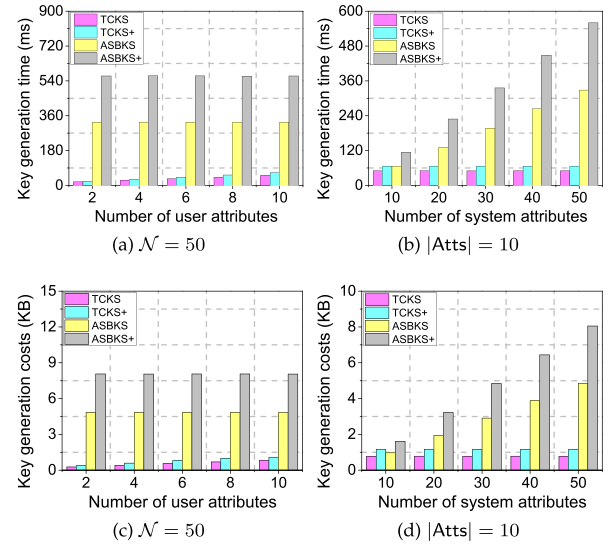


Fig. 4. Performance of KeyGen.

$(\{t_i', t_j'\}, \varphi^{r_{\text{time}}^u \lambda^{t_i'} \mu^{Z-t_j'}})$ if $t_j < t_j'$. Thus, there exists no algorithm that can derive $(\{t_i', t_j'\}, \varphi^{r_{\text{time}}^u \lambda^{t_i'} \mu^{Z-t_j'}})$ if $t_i < t_i', t_j < t_j'$ under the RSA assumption. $\square$

If Lemmas 1 and 2 both hold, then our proposed frameworks can resist the key collusion attack. This completes the proof of Theorem 3. $\square$

VI. PERFORMANCE ANALYSIS

In this section, we assess the performance of our proposed frameworks from two aspects including theoretical complexities and experimental tests using a real-world dataset.

A. Theoretical Complexities

To evaluate the complexities of our proposed frameworks including TCKS and TCKS+, we compare them with the state-of-the-art ASBKS (Attribute Set Based Keyword Search) scheme [40] as it exhibits the most overlapping functionality with our proposed schemes. For example, both ASBKS and our proposed schemes support attribute-based single-keyword search, fine-grained access control based on access tree, time control and user revocation. However, our proposed schemes support more flexible time-controllable keyword search and user revocation than ASBKS. We consider several time-consuming operations including the pairing operation P , modular exponentiation operation E (or E_T) in the groups $\mathbb{G}, \mathbb{G}_{s'}, \mathbb{G}_{n'}$ (or $\mathbb{G}_T$) and hash function H which maps an arbitrary string into the subgroup $\mathbb{G}_{s'}$. It is worth noticing that we neglect the hash function (which maps an arbitrary string into the field $\mathbb{Z}_n$) as it is very efficient. Let $|\mathbb{G}|, |\mathbb{G}_T|, |\mathbb{G}_{s'}|, |\mathbb{G}_{n'}|$ be the lengths of elements in groups $\mathbb{G}, \mathbb{G}_T, \mathbb{G}_{s'}, \mathbb{G}_{n'}$ respectively, note that $|\mathbb{G}_T| = 2|\mathbb{G}| = 2|\mathbb{G}_{s'}| = 2|\mathbb{G}_{n'}|$. The theoretical complexities of our proposed frameworks and ASBKS are shown in Table II.

From Table II we notice that the performance of our proposed frameworks is mainly affected by the number $|\text{Atts}|$ of user attributes, while that of ASBKS is influenced by the number N

TABLE II
THEORETICAL COMPLEXITIES OF OUR PROPOSED FRAMEWORKS AND PREVIOUS SCHEME: A COMPARATIVE SUMMARY

Algorithms		TCKS	TCKS+	ASBKS [40]	ASBKS+ [40]
KeyGen	CC	$5E + H + (2E + H) \text{Atts} $	$7E + (3E + H) \text{Atts} $	$(4E + H)\mathcal{N}$	$(6E + 2H)\mathcal{N}$
	SC	$4 \mathbb{G} + 2 \mathbb{G} \text{Atts} $	$6 \mathbb{G} + 3 \mathbb{G} \text{Atts} $	$3 \mathbb{G} \mathcal{N}$	$5 \mathbb{G} \mathcal{N}$
Enc	CC	$H + 6E + E W + (H + 2E)L$	$7E + H + E W + (H + 3E)L$	$4E + E W + (H + 2E)\mathcal{N}$	$4E + E W + (H + 2E)\mathcal{N}$
	SC	$5 \mathbb{G} + W \mathbb{G} + 2 \mathbb{G} L $	$6 \mathbb{G} + W \mathbb{G} + 3 \mathbb{G} L $	$3 \mathbb{G} + W \mathbb{G} + 2 \mathbb{G} \mathcal{N}$	$3 \mathbb{G} + W \mathbb{G} + 2 \mathbb{G} \mathcal{N}$
Token	CC	$5E + E_q + 2E \text{Atts} $	$7E + E_q + 3E \text{Atts} $	$2E + E_q + 3E\mathcal{N}$	$2E + E_q + 5E\mathcal{N}$
	SC	$5 \mathbb{G} + \mathbb{G} q + 2 \mathbb{G} \text{Atts} $	$7 \mathbb{G} + \mathbb{G} q + 3 \mathbb{G} \text{Atts} $	$ \mathbb{G} + \mathbb{G} q + 3 \mathbb{G} \mathcal{N}$	$ \mathbb{G} + \mathbb{G} q + 5 \mathbb{G} \mathcal{N}$
Search	CC	$P(W + q + 3) + 2P \text{Atts} $	$P(W + q + 4) + E_T + (3P + E_T) \text{Atts} $	$P(W + q) + 3P\mathcal{N}$	$P(W + q) + 5P\mathcal{N}$
	SC	$ \mathbb{G}_T (W + q + 3) + \mathbb{G}_T \text{Atts} $	$ \mathbb{G}_T (W + q + 3) + \mathbb{G}_T \text{Atts} $	$ \mathbb{G}_T (W + q) + 2 \mathbb{G}_T \mathcal{N}$	$ \mathbb{G}_T (W + q) + 2 \mathbb{G}_T \mathcal{N}$

Notes. We assume that each attribute has one value in ASBKS, but has two values in ASBKS+. In addition, the performance of ASBKS and ASBKS+ is affected by the number of system attributes, namely the variable $\mathcal{N}$.

“CC”: Computation costs; “SC”: Storage costs; “Atts”: Number of descriptive attributes; “L”: Number of leaf nodes in the access policies used in TCKS and TCKS+; “q”: Number of queried keywords; “|W|”: Number of keywords in dictionary.

of system attributes. When the value of $|\text{Atts}|$ is small, the key generation time and costs of our proposed frameworks including TCKS and TCKS+ are much fewer than those of ASBKS and ASBKS+. Due to the support for user revocation, TCKS+ has higher key generation burdens than TCKS. In addition, ASBKS+ has higher key generation overheads than ASBKS as its attribute has two values. In Enc, as TCKS+ needs to generate extra index components ($\{I_L\}, I_{\text{time}}^2$), thus it has higher encryption time and costs than those of TCKS. ASBKS and ASBKS+ have the same index computation cost $4E + E|W| + (H + 2E)\mathcal{N}$ and storage cost $3|\mathbb{G}| + |W||\mathbb{G}| + 2|\mathbb{G}|\mathcal{N}$. The encryption time and costs of our proposed frameworks are related with the number of leaf nodes in the access policy and the number $|W|$ of keywords, while those of ASBKS and ASBKS+ depend on the variables $\mathcal{N}, |W|$. In Token, the token generation time and costs of TCKS and TCKS+ are affected by the number q of queried keywords and the variable $|\text{Atts}|$, while those of ASBKS and ASBKS+ are influenced by variables $q, \mathcal{N}$. Due to $|\text{Atts}| \ll \mathcal{N}$, our proposed frameworks are efficient than ASBKS and ASBKS+ in terms of token generation time and costs. In Search, the search time and costs of TCKS and TCKS+ are related to the variables $|W|, q, |\text{Atts}|$, but those of ASBKS and ASBKS+ are susceptible to variables $|W|, q, \mathcal{N}$. For the same reason in Token, our proposed frameworks incur less computation and storage costs in the ciphertext retrieval process compared with ASBKS and ASBKS+.

B. Experimental Tests

To assess the actual performance of our proposed frameworks, ASBKS and ASBKS+, we conduct extensive experiments on macOS Catalina 10.15.4 with Intel Core i7 CPU 2.9 GHz by using Python3 language and Paring Based Cryptography (PBC) Library. The real-world dataset used in these empirical experiments comes from the public Enron Email Dataset³, and has a size of 422 MB emails distributed in 3500 folders. For simplicity, we set $|s'| = |n'| = 128$ bits.⁴ Thus, we get $|\mathbb{Z}_n| = 256$ bits, $|\mathbb{G}| = |\mathbb{G}_{s'}| = |\mathbb{G}_{n'}| = 264$ bits

³[Online]. Available: <http://www.cs.cmu.edu/~enron/>

⁴In our experiments, we set $|s'| = |n'| = 128$ bits for comparison. In practice, we can set the larger values of $|s'| = |n'|$, but will incur much more computation overheads. Thus, setting the values of $|s'| = |n'|$ is tradeoff between security and efficiency.

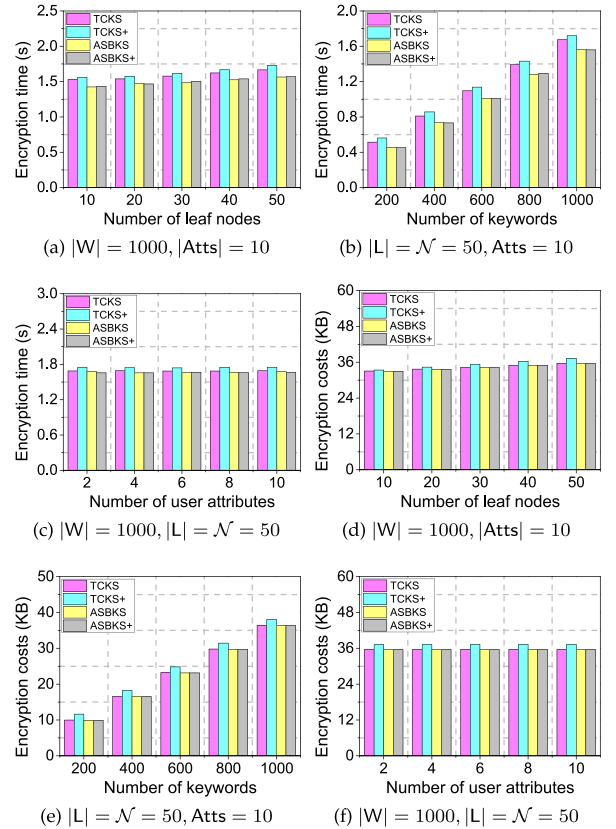


Fig. 5. Performance of Enc.

and $|\mathbb{G}_T| = 528$ bits. In addition, we set $q = |\text{Atts}| \in [1, 10]$, $|W| \in [1, 1000]$, $L = \mathcal{N} \in [1, 50]$ in the experimental tests.

Fig. 4 shows that the key generation time and costs of TCKS and TCKS+ linearly increase with the number $|\text{Atts}| \in [1, 10]$ of user attributes, but keep almost unchanged with the number $\mathcal{N} \in [1, 50]$ of system attributes. On the contrary, the key generation time and costs of ASBKS and ASBKS+ approximately grow as the value of $\mathcal{N}$ increases. In the case that $|\text{Atts}| = \mathcal{N}$, our proposed frameworks still have less key generation time compared with ASBKS and ASBKS+. This is because TCKS and TCKS+ respectively need to conduct about $2E + H$, $3E + H$ operations, but ASBKS and ASBKS+ respectively perform about $4E + H$, $6E + H$ operations. In addition, TCKS+ has slightly higher computation and storage costs than TCKS as it needs

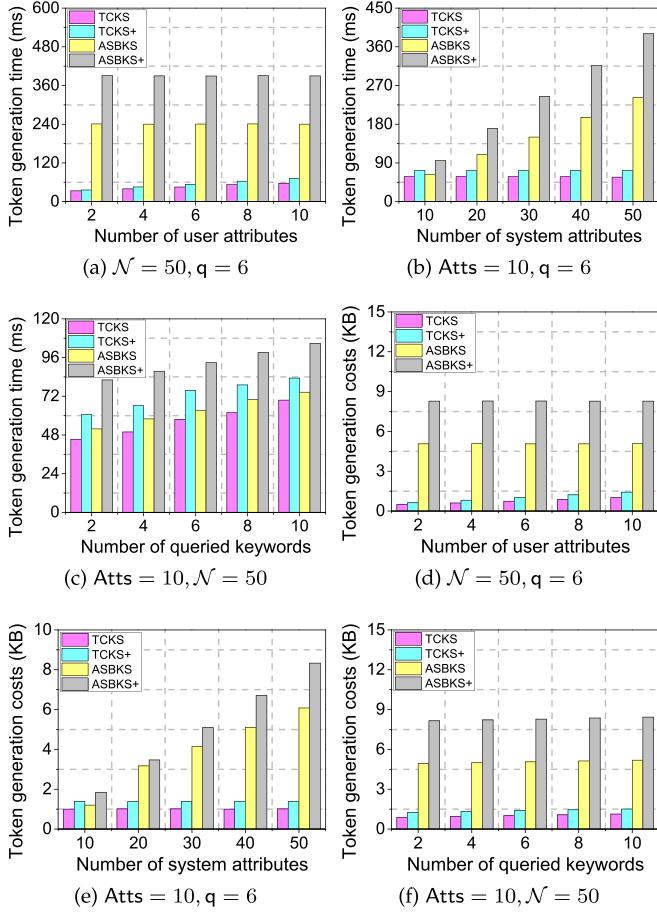


Fig. 6. Performance of Token.

to execute extra computations for achieving user revocation. ASBKS+ incurs higher computation and storage burdens than ASBKS as it computes twice for each attribute.

To facilitate comparison, the number of leaf nodes in the access policy is equal to that of system attributes in Fig. 5, namely $|L| = \mathcal{N}$. We notice that the index generation time and costs of all schemes linearly increase with the number of leaf nodes and the number of keywords, but keep unchanged as the number of user attributes increases. In Fig. 5(a), (b), (d), (e), TCKS and TCKS+ have slightly higher index computation and storage overheads than ASBKS and ASBKS+ as the number of leaf nodes or keywords increases. However, in practice, our proposed frameworks have fewer costs due to $|L| \ll \mathcal{N}$. In Fig. 5(c), (f), both our proposed frameworks and comparison schemes have constant index generation time and costs as the number of user attributes increases. It is worth noticing that TCKS has higher index generation time and cost than TCKS+ as it needs to generate extra ciphertexts for supporting user revocation, and ASBKS and ASBKS+ have the same token generation time and cost. With setting $|W| = 1000$, $|Atts| = 10$, $|L| = \mathcal{N} = 50$, the index generation time and costs of above four schemes are respectively (1.69 s, 35.61 KB), (1.75 s, 37.26 KB), (1.67 s, 35.55 KB), (1.66 s, 35.55 KB). Second level computation costs and KB level storage costs make it reasonable for DO to use resource-limited devices in mobile computing framework. As

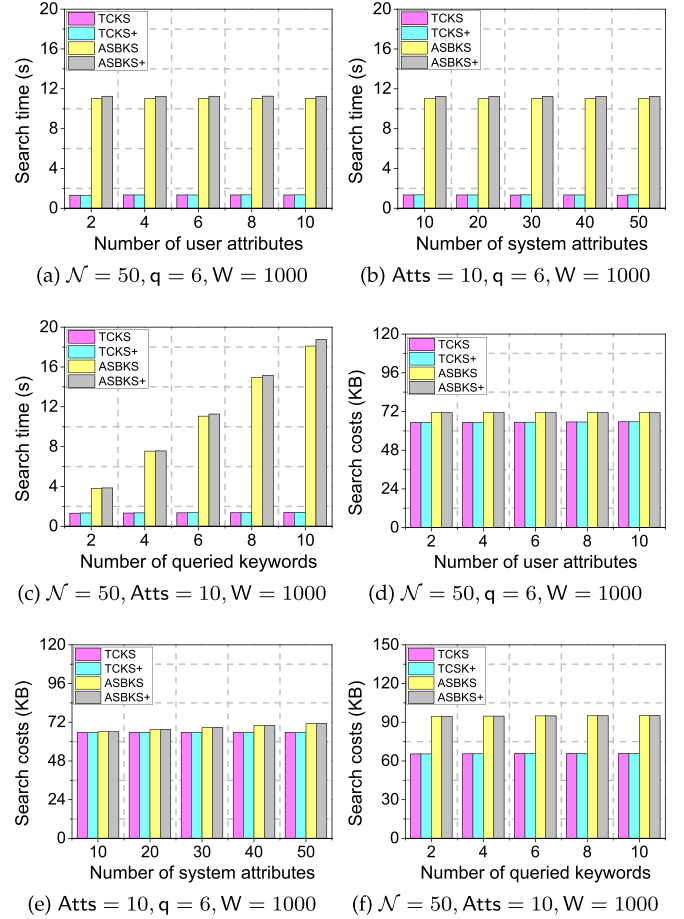


Fig. 7. Performance of Search.

the index building is just a one-time operation, the performance of Enc does not affect the practicability and scalability of our proposed frameworks in practical applications.

From Fig. 6 we notice that the token generation time and costs of TCKS and TCKS+ are much fewer than those of ASBKS and ASBKS+. With setting $\mathcal{N} = 50, q = 6$ in Fig. 6(a), (d), the token generation time and costs of TCKS and TCKS+ linearly grow with the variable $|Atts|$, but those of ASBKS and ASBKS+ are almost constant. In Fig. 6(b), (e), we set $|Atts| = 10, q = 6$, and find that the token generation time and costs of TCKS and TCKS+ are almost constant, while those of ASBKS and ASBKS+ linearly grow with the value of $\mathcal{N} \in [1, 50]$. When setting $|Atts| = 10, \mathcal{N} = 50$, the token generation time and costs of all schemes grow as the value of q increases in Fig. 6(c), (f), but our proposed frameworks outperform ASBKS and ASBKS+. For example, if DU is allowed to query 10 keywords at the same time, the token generation time and costs of TCKS, TCKS+, ASBKS, ASBKS+ are respectively (69.60 ms, 1.13 KB), (83.29 ms, 1.51 KB), (74.55 ms, 5.19 KB), (104.73 ms, 8.43 KB). Millisecond level computation costs and KB level storage costs make it reasonable for DUs to use resource-limited devices in mobile computing framework. Due to the support for user revocation, TCKS+ has slightly higher token generation time and cost than TCKS. As each attribute has

two values in ASBKS+, thus ASBKS+ has higher computation and storage costs than ASBKS in term of token generation.

From Fig. 7 we notice that our proposed TCKS and TCKS+ have less ciphertext retrieval time and costs than ASBKS and ASBKS+. In Fig. 7(a), (d), when setting $\mathcal{N} = 50$, $q = 6$, $W = 1000$, the search time and costs of TCKS and TCKS+ slightly increase with the variable $|\text{Atts}| \in [1, 10]$, but those of ASBKS and ASBKS+ keep almost unchanged. With setting $|\text{Atts}| = 10$, $q = 6$, $W = 1000$ in Fig. 7(b), TCKS and TCKS+ have almost constant search time and costs, but those of ASBKS and ASBKS+ slightly increase with the variable $\mathcal{N} \in [1, 50]$. In Fig. 7(c), (f), the search time of all schemes linearly increases with the variable $q \in [1, 10]$, but the search costs of all schemes are almost constant as the value of q increases. For example, when setting $q = 10$, $W = 1000$, $\mathcal{N} = 50$, $|\text{Atts}| = 10$, the search time and costs of above four schemes are respectively (1.37 s, 65.94 KB), (1.38 s, 65.94 KB), (18.10 s, 95.17 KB), (18.78 s, 95.17 KB). Due to $|\text{Atts}|, \mathcal{N}, q \ll |W|$, the effect of variables $|\text{Atts}|, \mathcal{N}$ on TCKS, TCKS+, ASBKS, ASBKS+ is not obvious.

VII. CONCLUSION

In this paper, we proposed a time-controllable fine-grained keyword search scheme using attribute-based comparable access control, which allows authorized data users to match indexes encrypted within specified time intervals. We then leveraged secret sharing to design an improved framework for supporting efficient user revocation. We formally proved that both proposed frameworks can resist chosen-keyword and key collusion attacks, and guarantee key secrecy. We also performed empirical experiments to demonstrate the feasibility and efficiency of our proposed frameworks using a real-world dataset.

As part of our future work, we will focus on designing multi-attribute comparison for real-world systems, and consider issue of single point of failure for the time server.

REFERENCES

- [1] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy.*, 2000, pp. 44–55.
- [2] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. Int. Conf. Theory Appl. Cryptogr. Techn.*, Springer, 2004, pp. 506–522.
- [3] P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data," in *Proc. Int. Conf. Appl. cryptogr. Netw. Secur.*, Springer, 2004, pp. 31–45.
- [4] P. Jiang, F. Guo, K. Liang, J. Lai, and Q. Wen, "Searchchain: Blockchain-based private keyword search in decentralized storage," *Future Gener. Comput. Syst.*, vol. 107, pp. 781–792, 2020.
- [5] K. Huang, M. Manulis, and L. Chen, "Password authenticated keyword search," in *Proc. IEEE Symp. Privacy-Aware Comput.*, 2017, pp. 129–140.
- [6] H. Li, Q. Huang, J. Shen, G. Yang, and W. Susilo, "Designated-server identity-based authenticated encryption with keyword search for encrypted emails," *Inf. Sci.*, vol. 481, pp. 330–343, 2019.
- [7] Q. Zheng, S. Xu, and G. Ateniese, "VABKS: Verifiable attribute-based keyword search over outsourced encrypted data," in *Proc. IEEE Conf. Comput. Commun.*, 2014, pp. 522–530.
- [8] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in *Proc. Annu. Int. Cryptol. Conf. Adv. Cryptol.*, Springer, 2001, pp. 213–229.
- [9] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2006, pp. 89–98.
- [10] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy.*, 2007, pp. 321–334.
- [11] L. Ou, Z. Qin, S. Liao, J. Weng, and X. Jia, "An optimal noise mechanism for cross-correlated IoT data releasing," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 4, pp. 1528–1540, Jul./Aug. 2020.
- [12] M. Abdalla et al., "Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions," in *Proc. Annu. Int. Cryptol. Conf. Adv. Cryptol.*, Springer, 2005, pp. 205–222.
- [13] M. H. Ameri, M. Delavar, J. Mohajeri, and M. Salmisazadeh, "A key-policy attribute-based temporary keyword search scheme for secure cloud storage," *IEEE Trans. Cloud Comput.*, vol. 8, no. 3, pp. 660–671, Jul./Sep. 2020.
- [14] Y. Yang and M. Ma, "Conjunctive keyword search with designated tester and timing enabled proxy re-encryption function for e-health clouds," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 4, pp. 746–759, Apr. 2015.
- [15] L. Xu, J. Li, X. Chen, W. Li, S. Tang, and H.-T. Wu, "Tc-PEDCKS: Towards time controlled public key encryption with delegatable conjunctive keyword search for Internet of Things," *J. Netw. Comput. Appl.*, vol. 128, pp. 11–20, 2019.
- [16] W. Sun, S. Yu, W. Lou, Y. T. Hou, and H. Li, "Protecting your right: Attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud," in *Proc. IEEE Conf. Comput. Commun.*, 2014, pp. 226–234.
- [17] S. Liu, J. Yu, Y. Xiao, Z. Wan, S. Wang, and B. Yan, "BC-SABE: Blockchain-aided searchable attribute-based encryption for cloud-IoT," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 7851–7867, Sep. 2020.
- [18] J. Wang, X. Yin, J. Ning, and G. S. Poh, "Attribute-based encryption with efficient keyword search and user revocation," in *Proc. Int. Conf. Inf. Secur. Cryptol.*, Springer, 2018, pp. 490–509.
- [19] M. Naor and B. Pinkas, "Efficient trace and revoke schemes," in *Proc. Financial Cryptogr.*, Springer, 2001, pp. 1–20.
- [20] A. Wu, D. Zheng, Y. Zhang, and M. Yang, "Hidden policy attribute-based data sharing with direct revocation and keyword search in cloud computing," *Sensors*, vol. 18, no. 7, 2018, Art. no. 2158.
- [21] J. Cui, H. Zhou, H. Zhong, and Y. Xu, "AKSER: Attribute-based keyword search with efficient revocation in cloud computing," *Inf. Sci.*, vol. 423, pp. 343–352, 2018.
- [22] Y. Zhu, H. Hu, G.-J. Ahn, M. Yu, and H. Zhao, "Comparison-based encryption for fine-grained access control in clouds," in *Proc. ACM Conf. Data Appl. Secur. Privacy*, 2012, pp. 105–116.
- [23] Z. Wang, D. Huang, Y. Zhu, B. Li, and C.-J. Chung, "Efficient attribute-based comparable data access control," *IEEE Trans. Comput.*, vol. 64, no. 12, pp. 3430–3443, Dec. 2015.
- [24] R. Zhang, R. Xue, and L. Liu, "Searchable encryption for healthcare clouds: A survey," *IEEE Trans. Services Comput.*, vol. 11, no. 6, pp. 978–996, Nov./Dec. 2017.
- [25] G. S. Poh, J.-J. Chin, W.-C. Yau, K.-K. R. Choo, and M. S. Mohamad, "Searchable symmetric encryption: Designs and challenges," *ACM Comput. Surveys*, vol. 50, no. 3, pp. 1–37, 2017.
- [26] Q. Huang and H. Li, "An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks," *Inf. Sci.*, vol. 403, pp. 1–14, 2017.
- [27] Y. Miao, Q. Tong, R. Deng, K.-K. R. Choo, X. Liu, and H. Li, "Verifiable searchable encryption framework against insider keyword-guessing attack in cloud storage," *IEEE Trans. Cloud Comput.*, vol. 10, no. 2, pp. 835–848, Apr./Jun. 2022, doi: [10.1109/TCC.2020.2989296](https://doi.org/10.1109/TCC.2020.2989296).
- [28] Y. Lu, G. Wang, J. Li, and J. Shen, "Efficient designated server identity-based encryption with conjunctive keyword search," *Ann. Telecommun.*, vol. 72, no. 5/6, pp. 359–370, 2017.
- [29] J. Lai et al., "Dual trapdoor identity-based encryption with keyword search," *Soft Comput.*, vol. 21, no. 10, pp. 2599–2607, 2017.
- [30] 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," *Inf. Sci.*, vol. 494, pp. 193–207, 2019.
- [31] J. Shu, K. Yang, X. Jia, X. Liu, C. Wang, and R. Deng, "Proxy-free privacy-preserving task matching with efficient revocation in crowdsourcing," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 1, pp. 117–130, Jan./Feb. 2021, doi: [10.1109/TDSC.2018.2875682](https://doi.org/10.1109/TDSC.2018.2875682).
- [32] Y. Miao et al., "Privacy-preserving attribute-based keyword search in shared multi-owner setting," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 3, pp. 1080–1094, May/Jun. 2021, doi: [10.1109/TDSC.2019.2897675](https://doi.org/10.1109/TDSC.2019.2897675).
- [33] Y. Miao, R. Deng, X. Liu, K.-K. R. Choo, H. Wu, and H. Li, "Multi-authority attribute-based keyword search over encrypted cloud data," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 4, pp. 1667–1680, Jul./Aug. 2021, doi: [10.1109/TDSC.2019.2935044](https://doi.org/10.1109/TDSC.2019.2935044).

- [34] S. Qiu, J. Liu, Y. Shi, and R. Zhang, "Hidden policy ciphertext-policy attribute-based encryption with keyword search against keyword guessing attack," *Sci. China Inf. Sci.*, vol. 60, no. 5, pp. 1–12, 2017.
- [35] H. Wang, X. Dong, and Z. Cao, "Multi-value-independent ciphertext-policy attribute based encryption with fast keyword search," *IEEE Trans. Services Comput.*, vol. 13, no. 6, pp. 1142–1151, Nov./Dec. 2020, doi: [10.1109/TSC.2017.2753231](https://doi.org/10.1109/TSC.2017.2753231).
- [36] K. Xue, J. Hong, Y. Xue, D. S. Wei, N. Yu, and P. Hong, "CABE: A new comparable attribute-based encryption construction with 0-encoding and 1-encoding," *IEEE Trans. Comput.*, vol. 66, no. 9, pp. 1491–1503, Sep. 2017.
- [37] Y. Yu, J. Ni, H. Yang, Y. Mu, and W. Susilo, "Efficient public key encryption with revocable keyword search," *Secur. Commun. Netw.*, vol. 7, no. 2, pp. 466–472, 2014.
- [38] J. Zhang and J. Mao, "Efficient public key encryption with revocable keyword search in cloud computing," *Cluster Comput.*, vol. 19, no. 3, pp. 1211–1217, 2016.
- [39] T. Liu et al., "Time-controlled hierarchical multikeyword search over encrypted data in cloud-assisted IoT," *IEEE Internet Things J.*, vol. 9, no. 13, pp. 11017–11029, Jul. 2022.
- [40] L. Xu et al., "ASBKS: Towards attribute set based keyword search over encrypted personal health records," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 6, pp. 2941–2952, Nov./Dec. 2021, doi: [10.1109/TDSC.2020.2970928](https://doi.org/10.1109/TDSC.2020.2970928).
- [41] J. J. Kang, K. Fahd, and S. Venkatraman, "Trusted time-based verification model for automatic man-in-the-middle attack detection in cybersecurity," *Cryptography*, vol. 2, no. 4, 2018, Art. no. 38.
- [42] A.-F. Chan and I. F. Blake, "Scalable, server-passive, user-anonymous timed release cryptography," in *Proc. IEEE 25th Int. Conf. Distrib. Comput. Syst.*, 2005, pp. 504–513.
- [43] Z. Wan and R. H. Deng, "VPSearch: Achieving verifiability for privacy-preserving multi-keyword search over encrypted cloud data," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 6, pp. 1083–1095, Nov./Dec. 2018.
- [44] Q. Liu, X. Nie, X. Liu, T. Peng, and J. Wu, "Verifiable ranked search over dynamic encrypted data in cloud computing," in *Proc. IEEE/ACM 25th Int. Symp. Qual. Serv.*, 2017, pp. 1–6.
- [45] H. Wang, C. Xu, C. Zhang, J. Xu, Z. Peng, and J. Pei, "vChain+ : Optimizing verifiable blockchain boolean range queries," in *Proc. IEEE 38th Int. Conf. Data Eng.*, 2022, pp. 1927–1940.
- [46] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.
- [47] G. Chen, T.-H. Lai, M. K. Reiter, and Y. Zhang, "Differentially private access patterns for searchable symmetric encryption," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 810–818.
- [48] L. Ou, Z. Qin, S. Liao, Y. Hong, and X. Jia, "Releasing correlated trajectories: Towards high utility and optimal differential privacy," *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 5, pp. 1109–1123, Sep./Oct. 2018.
- [49] L. Ou, Z. Qin, S. Liao, T. Li, and D. Zhang, "Singular spectrum analysis for local differential privacy of classifications in the smart grid," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5246–5255, Jun. 2020.
- [50] D. Boneh, D. Freeman, J. Katz, and B. Waters, "Signing a linear subspace: Signature schemes for network coding," in *Proc. Int. Workshop Public Key Cryptogr.*, Springer, 2009, pp. 68–87.
- [51] S. Hohenberger and B. Waters, "Synchronized aggregate signatures from the RSA assumption," in *Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Techn.*, Springer, 2018, pp. 197–229.



Yinbin Miao (Member, IEEE) received the BE degree from the Department of Telecommunication Engineering, Jilin University, Changchun, China, in 2011, and the PhD degree from the Department of Telecommunication Engineering, Xidian University, Xi'an, China, in 2016. He is also a postdoctor with Nanyang Technological University from September 2018 to September 2019, and a postdoctor with the City University of Hong Kong from December 2019 to December 2021. He is currently an Associate Professor with the Department of Cyber Engineering in Xidian University, Xi'an, China. His research interests include information security and applied cryptography.



Feng Li received the BE degree from the School of Cyberspace Security, Hangzhou Dianzi University, Hangzhou, China, in 2018. He is currently working toward the PhD degree with the School of Cyber Engineering, Xidian University. His current research interests include cloud computing security.



Xinghua Li received the ME and PhD degrees in computer science from Xidian University, Xian, China, in 2004 and 2007, respectively. He is currently a professor with the School of Cyber Engineering, Xidian University, China. His research interests include wireless networks security, privacy protection, cloud computing, and security protocol formal methodology.



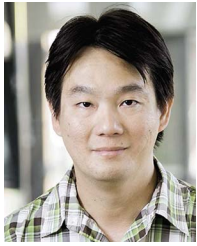
Zhiquan Liu (Member, IEEE) received the BS degree from the School of Science, Xidian University, Xi'an, China, in 2012, and the PhD degree from the School of Computer Science and Technology, Xidian University, Xi'an, China, in 2017. He is currently an Associate Professor with the College of Cyber Security, Jinan University, Guangzhou, China, and his current research focuses on trust management and privacy preservation in vehicular networks and UAV networks.



Jianting Ning (Member, IEEE) received the PhD degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, in 2016. He was a research fellow with the Department of Computer Science, National University of Singapore. His research interests include applied cryptography and information security, in particular, public key encryption, attribute-based encryption, and secure multi-party computation.



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



Kim-Kwang Raymond Choo (Senior Member, IEEE) received the PhD degree in information security from the Queensland University of Technology, Australia, in 2006. He currently holds the Cloud Technology Endowed Professorship with the University of Texas at San Antonio. He is the founding co-editor-in-chief of ACM Distributed Ledger Technologies: Research & Practice, and the founding chair of IEEE Technology and Engineering Management Society Technical Committee (TC) on Blockchain and distributed ledger Technologies. He is also the recipient of the 2022 IEEE Hyper-Intelligence TC Award for Excellence in Hyper-Intelligence Systems (Technical Achievement award), the 2022 IEEE TC on Homeland Security Research and Innovation Award, the 2022 IEEE TC on Secure and Dependable Measurement Mid-Career Award, and the 2019 IEEE TC on Scalable Computing Award for Excellence in Scalable Computing (Middle Career Researcher).



Robert H. Deng (Fellow, IEEE) is a AXA chair professor of cybersecurity and a professor of information systems with the School of Information Systems, Singapore Management University since 2004. His research interests include data security and privacy, multimedia security, network and system security. He served/is serving on the editorial boards of many international journals, including TIFS, TDSC. He has received the Distinguished Paper Award (NDSS 2012), Best Paper Award (CMS 2012), Best Journal Paper Award (IEEE Communications Society 2017).