1-2020

Lightweight sharable and traceable secure mobile health system

Yang YANG
Fuzhou University

Ximeng LIU
Fuzhou University

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

Yingjiu LI
Singapore Management University, yjli@smu.edu.sg

Published in IEEE Transactions on Dependable and Secure Computing, 2017 July, Volume PP, Issue 99, Pages 1-14
http://doi.org/10.1109/TDSC.2017.2729556

Lightweight Sharable and Traceable Secure Mobile Health System

Yang Yang, *Member, IEEE,* Ximeng Liu, *Member, IEEE,* Robert H. Deng, *Fellow, IEEE,* Yingjiu Li, *Member, IEEE,*

Abstract—Mobile health (mHealth) has emerged as a new patient centric model which allows real-time collection of patient data via wearable sensors, aggregation and encryption of these data at mobile devices, and then uploading the encrypted data to the cloud for storage and access by healthcare staff and researchers. However, efficient and scalable sharing of encrypted data has been a very challenging problem. In this paper, we propose a Lightweight Sharable and Traceable (LiST) secure mobile health system in which patient data are encrypted end-to-end from a patient's mobile device to data users. LiST enables efficient keyword search and fine-grained access control of encrypted data, supports tracing of traitors who sell their search and access privileges for monetary gain, and allows on-demand user revocation. LiST is lightweight in the sense that it offloads most of the heavy cryptographic computations to the cloud while only lightweight operations are performed at the end user devices. We formally define the security of LiST and prove that it is secure without random oracle. We also conduct extensive experiments to access the system's performance.

Index Terms—access control, searchable encryption, traceability, user revocation, mobile health system.

1 Introduction

Mobile health (mHealth) encompasses mobile devices and wireless communication technology to collect clinical health data and deliver them to healthcare providers [1]. The emergence of wireless body sensor network (WBSN) accelerates the development of mHealth. Implantable or wearable medical sensors are placed on patients to monitor and collect the physiological symptoms. These medical data are aggregated at a mobile device (such as a smart phone) and transmitted to the cloud via wireless networks for remote storage and access. The two major benefits that brought by mHealth are improved patient care and improved data access. "Improved patient care" means that mHealth could realize telemedicine since the patient's conditions can be measured remotely instead of face-to-face in the hospital. "Improved data access" means that healthcare providers can access critical electronic health record (EHR) at the point of care or at a remote location using a mobile terminal to provide in time medical treatment.

Mobile devices however have limited computation, storage and battery powers. It is not economical and practical for a hospital to equip thousands of healthcare staff with high performance mobile devices. Moreover, the busy work schedule in medical institutions also does not allow their physicians to wait for the charging of portable devices when their batteries are drain. Thus, it is critical to keep operations in all mobile devices lightweight in a mHealth system.

Apart from the performance concerns in mobile devices, data security and privacy concerns [2], [3] have been the major obstacle that hinders the wide spread adoption of mHealth systems. According to the Office of Civil Rights under Health and Human services of U.S., more than 113 million medical records were compromised in 2015 [4]. In mHealth systems, EHRs are outsourced to public cloud, data owners would not have direct control of the software and hardware platforms used to store their data. To mitigate security and privacy concerns about EHRs, a common solution is to provide end-to-end encryption by storing EHRs in encrypted form so that they remain private and secure, even if the cloud is not trusted or compromised. The encrypted EHRs, however, must be amenable to sharing and access control. Attribute based encryption (ABE) is an effective mechanism to provide fine-grained access control on encrypted data, in which secret keys of users and ciphertexts are dependent upon attributes. In ciphertext-policy ABE, which we will adopt, an access policy is associated with a ciphertext and a user's secret key is associated with a set of attributes. The ciphertext can be decrypted only if the set of attributes associated with the user's secret key satisfies the policy. In addition to fine-grained access control, effective keyword search over encrypted EHRs is an extremely useful feature in practice.

The high value of EHRs might motivate certain rogue healthcare staff, called traitors, to sell their secret keys for financial gains. Hence, it is imperative that the identity of a key owner who maliciously sells his/her secret key in the black market be traceable in mHealth systems. Furthermore, a mHealth system should be able to revoke authorized users' access privileges when the users misbehave or when their secret keys are being compromised. Most existing ABE-based data encryption systems require large scale periodic key update or ciphertext update to accomplish user revocation, which incur too much operational overhead for

Y. Yang and X. Liu are with College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China, 350116; and School of Information Systems, Singapore Management University, Singapore 188065. (email: yang.yang.research@gmail.com, snbnix@gmail.com)

R. H. Deng and Y. Li are with School of Information Systems, Singapore Management University, Singapore 188065. (email: robertdeng@smu.edu.sg, yjli@smu.edu.sg)

Y. Yang is also with Fujian Provincial Key Laboratory of Information Processing and Intelligent Control (Minjiang University), Fuzhou, China, 350121.

X. Liu is the corresponding author.

mHealth systems.

1.1 Our Contributions

In this paper, we propose a novel **Li**ghtweight **S**harable and **T**raceable (**LiST**) secure mHealth system. In addition to using ABE to achieve fine-grained access control of encrypted EHRs, LiST also supports keyword search over encrypted EHRs, efficient traitor tracing, and scalable user revocation. As mentioned before, the resource constraint feature of mobile devices in mHealth system requires that operations in mobile devices be lightweight. Maintaining lightweight computation and efficient storage in mobile devices throughout the LiST system operations is our overriding design objective. We achieve this by elaborately outsourcing most of the expensive cryptographic computations to the cloud without leaking the sensitive information. In particular, LiST provides the following functionalities.

- ***Lightweight encryption***. In the encryption algorithm, most of the ABE encryption computations are offloaded to the public cloud and only a few exponentiation operations are performed in a data owner's mobile device. Encrypted EHRs are uploaded to the public cloud for storage.
- ***Lightweight keyword trapdoor generation***. To obtain encrypted EHRs containing a certain keyword from the public cloud, a data user generates a keyword trapdoor and sends it to the cloud. In the keyword trapdoor generation algorithm, only a few lightweight multiplication, division and inversion operations are done in the data user's device.
- ***Lightweight test algorithm***. Upon receiving a keyword trapdoor from a data user, the cloud runs a test algorithm to retrieve encrypted EHRs containing the underlying keyword. Only three bilinear computations are required for the cloud storage provider to complete a test operation. As will be discussed later, the existing attribute based searchable encryption systems require a huge number of time consuming bilinear operations.
- ***Lightweight decryption and verification***. In the decryption algorithm, most of the ABE decryption operations are outsourced to the public cloud. That is, the cloud first transforms an encrypted EHR into an intermediate ciphertext and sends it to a data user. The data user's device only needs to perform one exponentiation computation to obtain the underlying EHR and verifies that the transformation done by the cloud is correct.
- ***Lightweight user revocation***. Instead of expensive periodic large-scale secret key update or ciphertext re-encryption, an exquisite design in LiST guarantees an ultra-lightweight user revocation mechanism.
- ***lightweight traitor tracing***. Due to the one-to-many encryption characteristic of ABE, decryption privileges can be shared by a group of users who own the same set of attributes. It is extremely hard to reveal the original secret key owner's identity from an exposed secret key since most existing ABE schemes allow key randomization. LiST supports lightweight traitor tracing, only three bilinear operations are involved in the traitor tracing algorithm, and no additional storage or identity table is required.

We provide a thorough analysis of the security of LiST and a detailed performance comparison of LiST with existing schemes. Extensive simulations and experiments are conducted on both fixed and mobile platforms to validate the performance of LiST. Our results indicates that LiST is promising for practical applications.

1.2 Related Work

To realize fine-grained access control for outsourced data, ABE provides a cryptographically approach to achieve one-to-many data encryption and sharing. The notion of ABE was first put forth by Goyal et al [5]. They proposed the first key policy ABE (KP-ABE) scheme and the first ciphertext policy ABE (CP-ABE) scheme based on access tree. Ostrovsky et al [6] introduced a new KP-ABE scheme such that user's private key can represent any Boolean access formula over attributes. To remove the trusted central authority, [7] and [8] present multi-authority system to realize decentralized ABE. However, these schemes suffer from a large computation overhead.

In order to reduce the computation operations at an end user's device, Green et al. [9] introduced outsourcing decryption mechanism to ABE system, which allows a proxy to transform a ciphertext into another form so that the user can recover the message efficiently. However, the correctness of transformation in [9] can not be verified. Later, Lai et al. [10] presented a verifiable outsourced decryption (VOD) ABE scheme by appending a redundant message as the auxiliary verification information. Although verifiability is achieved in [10], it doubles the length of ciphertext and introduces significant overhead in encryption operation. The VOD issue is further discussed in schemes [11], [12]. The decryption computation overhead is reduced in these schemes, but the encryption cost still grows with the complexity of access structure. Furthermore, these schemes can not provide search function on ciphertexts.

Another problem in the ABE mechanism is that a user's secret key is associated with a set of attributes rather than the user's identity. The same set of attributes can be shared by a group of users. If a malicious authorized user sells his secret key for financial gain, it would be impossible to identify the suspect in the traditional ABE schemes. The problem of tracing the original user from a secret key is named as *white-box traceability* [13], [14]. If the leakage is the decryption equipment instead of the secret key, this stronger tracing notion is called *black-box traceability* [15]. Existing traitor tracing schemes [13], [14], [15] either requires the maintenance of a user list or incurs a large computation overhead. In this paper, we provide a solution for lightweight white-box traceability.

Although ABE encryption could prevent a storage provider or outside attacker from revealing sensitive EHRs, it still faces the problem of data usability. The encryption algorithm exerts unreadability to the medical files and prohibits users from performing operations on them, such as the most commonly used information retrieval. As a seminal work, Song et al. [16] proposed the first scheme

to enable keyword search over encrypted files. Following this work, a lot of searchable encryption schemes have been proposed, which can be classified into two categories: symmetric searchable encryption (SSE) [17], [18], [19] and public key encryption with keyword search (PEKS) [20], [21], [22], [23].

In SSE schemes, a user Alice uploads its encrypted data to a remote server and keeps the secret key private from the server. It allows another user Bob to search on the private files using a specified keyword under the premise that the secret key is shared between Alice and Bob. SSE is only suitable for the scenario where a group of users fully trust each other.

It is obvious that this assumption is not suitable for a mHealth system. In order to provide searchable encryption ability in the public key setting, Boneh et al. [24] introduced PEKS to enable searchable data sharing between untrusted parties. Later on, Curtmola et al. [25] proposed a dynamic searchable encryption scheme using the inverted index. Boneh and Waters et al. [26] presented a novel PEKS scheme that supports conjunctive, subset and range keyword queries but the scheme suffers from large computation and storage overhead. To provide multiple users the search ability, authorized searchable encryption is desirable. In [27], [28], [29], the ABE mechanism is introduced to searchable encryption system. The outsourced files can be contributed from multiple data owners and searchable by multiple users [28]. A data owner could enforce access policy on the index of the documents to realize search authorization. However, the scheme in [28] requires the system to re-encrypt the encrypted files and update users' secret keys to revoke a user. This is not suitable for a large mHealth system, which has a massive amount of EHRs and a large number of users. Moreover, the computation costs of these schemes [27], [28], [29] increase with the complexity of access structures. More recently, PEKS schemes dealing with post quantum attacks, keyword guessing attacks, and key-escrow problems have been proposed in [30], [33], [31], and [32], respectively. However, these schemes do not provide the access control function.

2 PRELIMINARIES

In this section, some basic notations and definitions used in LiST are introduced.

2.1 Access Policy

Definition 1 (Access Structure [34]). Let $\{P_1, \cdots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1,\cdots,P_n\}}$ is monotone if $\forall B$ and C: if $B \in \mathbb{A}$ and $B \subseteq C$ then $C \in \mathbb{A}$. An access structure (respectively, monotone access structure) is a collection (resp. monotone collection) $\mathbb{A}$ of non-empty subsets of $\{P_1, \cdots, P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1,\cdots,P_n\}} \backslash \{\phi\}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.

The role of parties is taken by attributes in ABE scheme. Thus, an access structure $\mathbb{A}$ contains the authorized sets of attributes. As shown in [34], any monotone access structure can be represented by a linear secret sharing scheme.

Definition 2 (Linear Secret Sharing Scheme (LSSS) [34]). A secret-sharing scheme Π over a set of parties $\mathcal{P}$ is called linear (over $\mathbb{Z}_p$) if

- The shares for each party form a vector over $\mathbb{Z}_p$.
- There exists a matrix M with l rows and n columns called the share-generating matrix for Π. For all $i = 1, \cdots, l$, the ith row of M is labeled by a party $\rho(i)$ (ρ is a function from $\{1, \cdots, l\}$ to $\mathcal{P}$). When we consider the column vector $v = (s, r_2, \cdots, r_n)$, where $s \in \mathbb{Z}_p$ is the secret to be shared and $r_2, \cdots, r_n \in \mathbb{Z}_p$ are randomly chosen, then Mv is the vector of l shares of the secret s according to Π. The share $(Mv)_i$ belongs to party $\rho(i)$.

Every LSSS according to the definition achieves the linear reconstruction property [34]. Suppose that Π is an LSSS for the access structure $\mathbb{A}$. Let $S \in \mathbb{A}$ be any authorized set and $I \subset \{1, \cdots, l\}$ be defined as $I = \{i : \rho(i) \in S\}$. Then, there exists constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ such that, if $\{\lambda_i\}_{i \in I}$ are valid shares of any secret s according to Π, then $\sum_{i \in I} \omega_i \lambda_i = s$. Furthermore, it is shown in [34] that these constants $\{\omega_i\}_{i \in I}$ can be found in time polynomial in the size of the share-generating matrix M. For unauthorized sets, no such constants exist. In this paper, an LSSS matrix (M, ρ) will be used to express an access policy associated to a ciphertext.

2.2 Bilinear Groups

Let $\mathcal{G}_p$ be an algorithm that on input the security parameter λ, outputs the parameters of a prime order bilinear map as $(p, g, \mathbb{G}, \mathbb{G}_T, e)$, where $\mathbb{G}$ and $\mathbb{G}_T$ are multiplicative cyclic groups of prime order p and g is a random generator of $\mathbb{G}$. The mapping $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a bilinear map. The bilinear map e has three properties: (1) *bilinearity*: $\forall u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(uv)^{ab}$. (2) *non-degeneracy*: $e(g, g) \neq 1$. (3) *computability*: e can be efficiently computed.

2.3 Assumptions

The security of LiST is based on the following assumptions.

Assumption 1 (q-SDH assumption [35]). Let $\mathbb{G}$ be a bilinear group of prime order p and g be a generator of $\mathbb{G}$, the q-Strong Diffie-Hellman (q-SDH) problem in $\mathbb{G}$ is defined as follows: given a $(q+1)$-tuple $(g, g^x, g^{x^2}, \cdots, g^{x^q})$ as inputs, output a pair $(c, g^{1/(c+x)}) \in \mathbb{Z}_p \times \mathbb{G}$. An algorithm $\mathcal{A}$ has advantage ϵ in solving l-SDH in $\mathbb{G}$ if $Pr[\mathcal{A}(g, g^x, g^{x^2}, \cdots, g^{x^q}) = (c, g^{1/(c+x)})] \geq \epsilon$, where the probability is over the random choice of x in $\mathbb{Z}_p^*$ and the random bits consumed by $\mathcal{A}$.

Definition. We say that the (q, t, ϵ)-SDH assumption holds in $\mathbb{G}$ if no t-time algorithm has advantage at least ϵ in solving the q-SDH problem in $\mathbb{G}$.

Assumption 2 (decisional bilinear Diffie-Hellman assumption). Let $\mathbb{G}$ be a bilinear group of prime order p and g be a generator of $\mathbb{G}$. Let $a, b, s \in \mathbb{Z}_p^*$ be chosen at random. If an adversary $\mathcal{A}$ is given $\vec{y} = (g, g^a, g^b, g^s)$, it is hard for the attacker $\mathcal{A}$ to distinguish $e(g, g)^{abs} \in \mathbb{G}_T$ from an element R that is randomly chosen from $\mathbb{G}_T$.

The adversary $\mathcal{A}$ has advantage ϵ in solving the above assumption if

$$\Big| Pr[\mathcal{A}(\vec{y}, T = e(g, g)^{abs}) = 0] - Pr[\mathcal{A}(\vec{y}, T = R) = 0] \Big| \geq \epsilon.$$

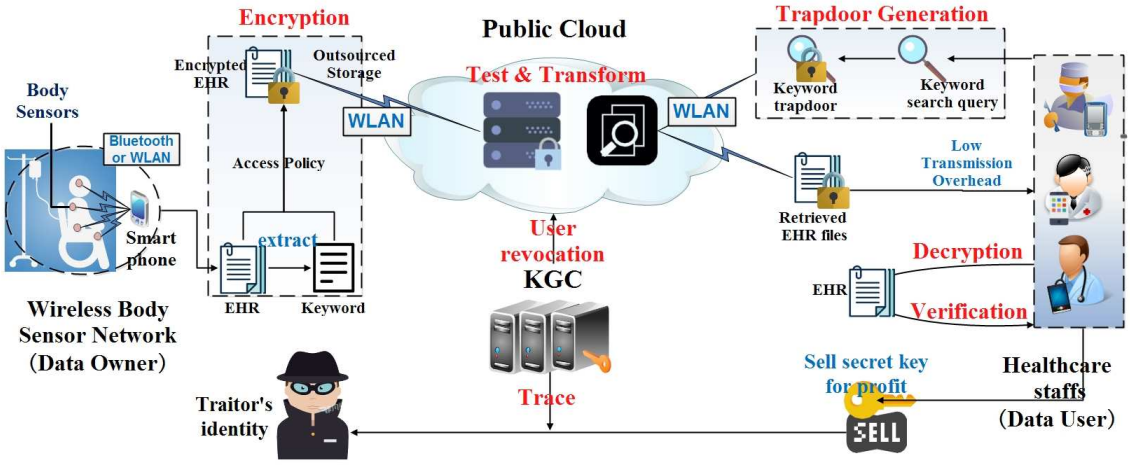


Fig. 1: LiST System Architecture

3 System Model

In this section, we describe the system architecture, threat model and security requirements of LiST.

3.1 System Architecture

As shown in Fig. 1, the architecture of LiST mainly consists of four types of parties: the wireless body sensor network (WBSN) which acts as the function of data owner, healthcare staff which is deemed as a data user, the public cloud and the key generation center (KGC). The characteristic and function of each party are described below.

• ***WBSN (data owner).*** WBSN involves tiny wireless sensors that are embedded inside or surface-mounted on the body of a patient. These sensors continuously monitor the vital physiology parameters of the patient suffering from chronic diseases such as diabetes, asthma and heart problems. Collected personal health data are aggregated and transmitted to a mobile device via wireless interface, such as bluetooth or WLAN. Keyword to depict the health information is extracted from the health record. Then, the keyword and EHR are encrypted into a ciphertext under a specific access policy. These encrypted EHRs are outsourced to public cloud server for remote storage. The encryption algorithm should be lightweight since the personal wireless terminal has low computation capability and battery power.

• ***Healthcare staff (data user).*** Healthcare staff are the data users in mHealth network. Each data user has a set of attributes, such as affiliation, department and type of healthcare staff, and is authorized to search on encrypted EHRs based on his set of attributes. In mHealth system, a data uses resource-limited mobile terminals to generate keyword trapdoors and conduct the information retrieval operation. The trapdoors are sent to the public cloud via wireless channel and the retrieved EHR files are returned. Then, the data user decrypts the EHR files and verifies the correctness of decryption. In LiST, the trapdoor generation, decryption and verification are all lightweight operations.

• ***Public cloud.*** The public cloud has almost unlimited storage and computing power to undertake the EHR remote storage task and respond on data retrieval requests. Lightweight test algorithm is designed in our proposed system to improve performance. In addition, the public cloud helps to convert the retrieved ciphertext into a transformed one so that the data user can decrypt it by lightweight computation.

• ***KGC.*** KGC generates public parameters for the entire system and distributes secret keys to data users. A data user's set of attributes is embedded in his secret key in LiST to realize access control. If a traitor sells his secret key for financial gain, the KGC is able to trace the identity of the malicious user and revoke his secret key. Both traitor tracing and user revocation algorithms in LiST are lightweight.

The formal definition of LiST is described in Supplemental Materials A.

3.2 Threat Model

We assume that the KGC is a fully trusted entity. In our system, the public cloud server is deemed as semi-honest and curious. It follows the pre-defined operations to conduct search on EHRs on behalf of data users but is curious in the sense that it tries to derive sensitive information from the stored EHRs or the plaintext keywords from keyword trapdoors submitted by users. Moreover, the public cloud may try to save its computation resource or bandwidth by returning incorrectly transformed ciphertexts to users. Data owners are supposed to honestly encrypt and upload their EHRs. Data users are not trusted, who may sell their secret keys for financial gain. We assume that the public cloud does not collude with revoked users in order to obtain unauthorized data or gain decryption privilege. All attackers are assumed to have polynomial time bounded computation ability such that they can not solve the hardness problems mentioned in Section 2.3.

3.3 Security Requirements

In order to guarantee security of the keyword and ciphertext of an EHR, a secure searchable data sharing system should satisfy the following requirement: indistinguishable

against chosen keyword and chosen ciphertext attack (IND-CKCCA) [23], [27], [28]. This requirement guarantees that an attacker (either the cloud server or an outside adversary) is not able to distinguish two challenge ciphertexts (given two corresponding plaintext messages and keywords). Lots of training opportunities will be given to the adversary before and after the challenge phase.

Another security requirement is on the traceability [13], [14]. This requirement ensures that any adversary can not forge a valid secret key without knowing the master secret key of the KGC. It also guarantees that a traitor can be traced if a well-formed secret key is sold for financial gain.

The concrete definitions of security models of IND-CKCCA and traceability can be found in Supplemental Materials B.

4 Proposed System

4.1 System Overview

A highlight of LiST is the lightweight computation at user's mobile device. The basic approach to achieve this is to offload most of the computation intensive operations to the cloud server such that only some marginal operations are left to the user device.

The workflow of the system architecture (shown in figure 1) is described as follows.

(1) When an EHR is generated from a wireless body sensor network, the data owner extracts a keyword to describe the EHR. Then, both the EHR and keyword are encrypted using a lightweight encryption algorithm. During the encryption process, the access policy specified by the data owner is embedded in the encrypted EHR. Then, the ciphertext is outsourced to the public cloud.

(2) When an authorized healthcare staff (data user) intends to issue a search query, he generates a keyword trapdoor using a lightweight trapdoor generation algorithm, and sends the trapdoor to the public cloud.

(3) Upon receiving the data retrieval request, the public cloud executes a lightweight test algorithm to find the matched ciphertexts. Then, the public cloud transforms the matched ciphertexts into outsourced ciphertexts, and sends them to the healthcare staff.

(4) Upon obtaining the transformed ciphertexts, the healthcare staff recovers the plaintext EHRs with a lightweight decryption algorithm and checks the correctness of the decryption output using a lightweight verification algorithm.

(5) When a secret key is found in the underground market, the KGC firstly verifies whether the secret key is a valid key generated by itself. If it is a valid key, the KGC runs a lightweight trace algorithm to reveal the identity of the key owner.

(6) To protect privacy of EHRS, efficient user revocation is essential. The KGC uses a lightweight revocation mechanism to remove revoked users' data retrieval and decryption privileges.

4.2 Concrete Construction

In this subsection, we present a concrete construction of LiST. Important notations are summarized in Table 1 for ease of reference. High level interactions among various entities in the concrete construction are illustrated in Fig. 2.

TABLE 1: Summary of Notations

Notation	Description
PP/MSK	public parameter/master secret key
$S/(A,\rho)$	attribute set/access structure
PK/SK	public key/ secret key pair of user
KW/T_{KW}	keyword/keyword trapdoor
CT/C_m	index/message ciphertext
CT_{out}	outsourced ciphertext
$SEnc/SDec$	symmetric encryption/decryption pair
$\mathcal{K}$	key space of symmetric encryption

4.2.1 System Setup

The KGC takes the security parameter 1^λ as input. It outputs the public parameter PP of the whole system and keeps secret the generated master secret key MSK.

- $Setup(1^\lambda) \rightarrow (PP, MSK)$. Run $\mathcal{G}_p(1^\lambda) \rightarrow (p, \mathbb{G}, \mathbb{G}_T, e)$. Let $g \in \mathbb{G}$ be the generator of group $\mathbb{G}$. Select random $\alpha, \lambda, \tau \in_R \mathbb{Z}_p^*$ and $k_1, k_2 \in_R \mathcal{K}$. Compute $f = g^\tau, Y = e(g,g)^\alpha, Y_0 = e(g,f), h = g^\lambda$. The public parameter is $PP = (g, h, f, Y, Y_0)$ and the master secret key is $MSK = (\alpha, \lambda, \tau, k_1, k_2)$. The KGC also defines two hash functions: $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ and $H_1 : \{0,1\}^* \rightarrow \mathcal{K}$. We will omit PP in the expressions of the following algorithms.

4.2.2 Key Generation

As shown in Fig. 2, the KGC and data user are involved in the following key generation protocol. The KGC generates the public/secret key pair for each data user using $KeyGen$ algorithm. The identity id and attribute set S of user are embedded in the created secret key $SK_{id,S}$.

- $KeyGen(MSK, id, S) \rightarrow (PK_{id,S}, SK_{id,S})$. The key generation algorithm takes the master secret key MSK, the user's identity id and an attribute set $S = \{\xi_1, \xi_2, \cdots, \xi_k\} \subseteq \mathbb{Z}_p^*$ as input. Choose $a, r, \theta, \varrho, s', s'', u', u'', u''' \in_R \mathbb{Z}_p^*$. Compute $\zeta = SEnc_{k_1}(id), \delta = SEnc_{k_2}(\zeta||\theta)$. The public key $PK_{id,S}$ and secret key $SK_{id,S}$ are constructed as:

$$D_1 = g^{\frac{\alpha - ar}{\lambda + \delta}}, D_2 = \delta, D_{3,i} = g^{ar(\xi_i + \tau)^{-1}}, D_4 = \varrho,$$

$$\Psi_1 = (D_1^\varrho)^{u'}, \Psi_2 = Y_0^{u'''}, \Psi_{3,i} = (D_{3,i}^\varrho)^{u''},$$

$$\Psi_4 = g^{s'}, \Psi_5 = f^{s''},$$

$$PK_{id,S} = (\Psi_1, \Psi_2, \{\Psi_{3,i}\}_{i \in [k]}, \Psi_4, \Psi_5),$$

$$SK_{id,S} = (D_1, D_2, \{D_{3,i}\}_{i \in [k]}, D_4, s', s'', u', u'', u''').$$

Note that $1/(\lambda + \delta)$ is computed modulo p. If $gcd(\lambda + \delta, p) \neq 1$, the $KeyGen$ algorithm chooses another $\theta \in_R \mathbb{Z}_p^*$ and repeat the computation $\delta = SEnc_{k_2}(\zeta||\theta)$.

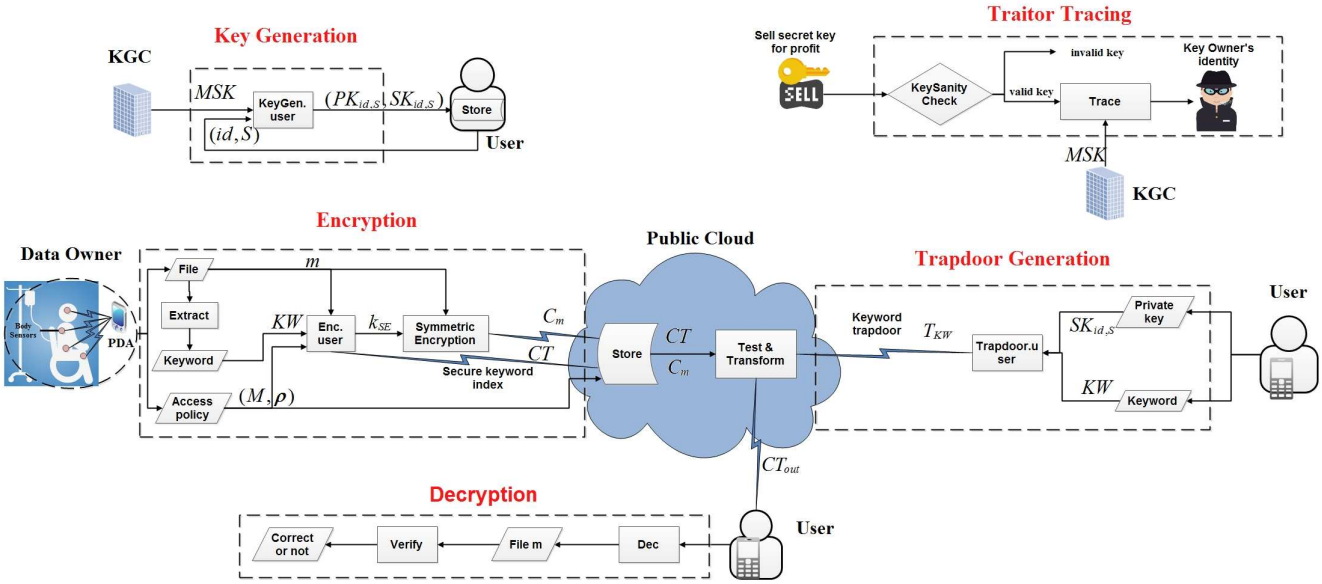


Fig. 2: High level interactions in the concrete construction

4.2.3 EHR and Keyword Encryption

The data owner performs the following steps to encrypt an EHR. First, the keyword used to depict the file (such as the disease name) is extracted. Secondly, the data owner selects a random number and calculates its hash value, which is used as a symmetric key to encrypt EHR. In order to support decryption verification, the data owner appends a string of zeros to EHR and generates the message ciphertext. Thirdly, the data owner specifies the access policy of the EHR. Lastly, he encrypts the keyword and random number using the access policy.

- $Enc(m, (M, \rho), KW) \rightarrow CT$. Let M be an $l \times n$ matrix and ρ be the function that associates rows of M to attributes. Select random $\Upsilon \in_R \mathbb{G}_T^*$. Set $k_{SE} = H_1(\Upsilon)$. The data owner selects a positive integer ϖ and concatenates ϖ-bit 0 string after the message m, which is used for outsourced decryption verification. Then, compute $C_m = SEnc_{k_{SE}}(m||0^{\varpi})$, where $||$ denotes concatenation of a string. The data owner picks a random $s \in_R \mathbb{Z}_p^*$ and then choose a random vector $\vec{v} = (s, y_2, \cdots, y_n)^{\top} \in \mathbb{Z}_p^n$ which is used to share s. For $i \in [l]$, compute $s_i = M_i \cdot \vec{v}$, where M_i is the vector corresponding to the ith row of M. Compute the ciphertext CT as:

$$C_0 = \Upsilon \cdot Y^s, C_1 = g^s, C_2 = h^s,$$

$$C_{3,i} = \rho(i) s_i / [s' H(KW)], C'_{3,i} = s_i / [s'' H(KW)],$$

$$C_4 = Y_0^{H(KW)} Y^{s/H(KW)}.$$

Then, the access policy (M, ρ) and the ciphertext $CT = (C_0, C_1, C_2, \{C_{3,i}, C'_{3,i}\}_{i \in [l]}, C_4, C_m)$ are outsourced to cloud.

Remark: In the LiST system, the encryption algorithm is suitable for all kinds of EHRs, such as the X-Ray pictures and MRI scan files. These EHRs are encrypted using the symmetric encryption algorithm $SEnc$ and the file ciphertext is denoted as $C_m = SEnc_{k_{SE}}(m||0^{\varpi})$, where m represents the EHR. No matter what type the EHR is, the data owner should extract keyword KW to describe the EHR. The keyword is encrypted to index. Then, the file ciphertext and encrypted keyword index are outsourced to cloud. The keyword search algorithm and decryption algorithm work regardless of the types of EHR.

4.2.4 Keyword Trapdoor Generation

The data user generates the keyword trapdoor T_{KW} using the following $Trapdoor$ algorithm. The attribute set S is also implicitly included in the generated trapdoor T_{KW}, which is transmitted to public cloud server via wireless channel.

- $Trapdoor(SK_{id,S}, KW) \rightarrow T_{KW}$. The data user chooses $u, u_0 \in \mathbb{Z}_p^*$ and computes the keyword trapdoor $T_{KW} = (T_0, T_1, T_2, T_3, T'_3, T_4, T_5)$ as:

$$T_0 = u \cdot (u')^{-1}, T_1 = u_0 / [u' H(KW)], T_2 = D_2,$$

$$T_3 = u_0 \cdot (u'')^{-1}, T'_3 = uH(KW) \cdot (u'')^{-1},$$

$$T_4 = u_0 D_4, T_5 = u_0 D_4 \cdot H(KW) \cdot (u''')^{-1}.$$

4.2.5 Data Retrieval

Receiving the data retrieval request from data user, the public cloud server responds on the request and searches on the stored encrypted EHRs to look for matching files. The cloud server provider leverages on the test and transform phases to complete the process: $Test\&Transform(CT, T_{KW}, PK_{id,S}) \rightarrow CT_{out}/\perp$.

In the $Test$ algorithm, the public cloud server searches for the matching encrypted EHRs if the ciphertext satisfies the following two requirements: the attribute set S (implicitly included in keyword trapdoor) satisfies the access

structure defined in the encrypted EHR; the keyword contained in the keyword trapdoor is in accordance with that in ciphertext.

- $Test(CT, T_{KW}, PK_{id,S}) \rightarrow 1/0$. Suppose CT associate with keyword KW' and T_{KW} with KW. The algorithm verifies whether S associated with T_{KW} satisfies (M, ρ) associated with CT. If not, it outputs 0. Otherwise, let $I \subset [l]$ be defined as $I = \{i : \rho(i) \in S\}$. There exists a set of constants $\{w_i \in \mathbb{Z}_p\}_{i \in I}$ so that $\sum_{i \in I} w_i M_i = (1, 0, \cdots, 0)$. The algorithm computes

$$\Lambda = e[\Psi_4, \prod_{i \in I} \Psi_{3,i}^{C_{3,i} \cdot w_i}] \cdot e[\Psi_5, \prod_{i \in I} \Psi_{3,i}^{C'_{3,i} \cdot w_i}],$$

$$\Gamma = e(\Psi_1, C_1^{T_2} C_2), \quad \Gamma_2 = \Gamma^{T_1}, \Lambda_2 = \Lambda^{T_3}.$$

Then, the algorithm verifies whether the following equation holds

$$\Psi_2^{T_5} \cdot (\Gamma_2 / \Lambda_2) = C_4^{T_4}.$$

If the equation does not hold, it outputs 0 indicating that $KW' \neq KW$. Otherwise, it outputs 1.

In the $Transform$ algorithm, the public cloud server transforms the matched ciphertext CT into CT_{out} such that the data user could use lightweight decryption algorithm to recover the plaintext.

- $Transform(CT, T_{KW}, PK_{id,S}) \rightarrow CT_{out}/\perp$. If the output of "$Test$" algorithm is 0, it outputs $\perp$. Otherwise, it computes $CT_{out} = (C_0, \Gamma_1, \Lambda_1, C_m)$, in which $\Gamma_1 = \Gamma^{T_0}, \Lambda_1 = \Lambda^{T'_3}$.

4.2.6 Data Decryption and Verification

Receiving the transformed ciphertext CT_{out}, the data user performs only one modular exponentiation computation to recover the random number Υ, which can be computed as symmetric key to recover the EHR m. In order to verify whether the received CT_{out} is correctly transformed from the original CT, the data user checks whether a string of zeros is appended to m.

- $Dec(CT_{out}, SK_{id,S}) \rightarrow m/\perp$. Compute

$$\frac{C_0}{(\Gamma_1 / \Lambda_1)^{1/(uD_4)}} = \Upsilon.$$

Then, the user computes $k_{SE} = H_1(\Upsilon)$ and $m' = SDec_{k_{SE}}(C_m)$. The user checks whether a redundancy 0^{ϖ} is appended after the recovered message. If so ($m' = m||0^{\varpi}$), the message m can be obtained by truncating ϖ-bit 0 string. Otherwise, the cloud server is dishonest to return an incorrect transformed ciphertext and the algorithm outputs $\perp$.

4.2.7 Traitor Tracing

A highlight of LiST is that the traitor can be traced if a sold secret key is found in market. The KGC firstly verifies whether the sold key is a well-formed key via $KeySanityCheck$ algorithm.

- $KeySanityCheck(SK_{id,S}) \rightarrow 1/0$. Suppose $S = \{\xi_1, \xi_2, \cdots, \xi_k\}$. The key sanity check of secret key $SK_{id,S}$ consists of two steps. Firstly, the KGC checks whether $SK_{id,S}$ is in the form of $(D_1, D_2, \{D_{3,i}\}_{i \in [k]}, D_4, \ s', s'', u', u'', u''')$ and $D_2, D_4, s', s'', u', u'', u''' \in \mathbb{Z}_p^*, D_1, D_{3,i} \in \mathbb{G}$. Then, the KGC verifies whether the equation holds

$$e(D_1, h \cdot g^{D_2})^k \cdot e(g, \prod_{i \in [k]} D_{3,i}^{\xi_i}) \cdot e(f, \prod_{i \in [k]} D_{3,i}) = Y^k.$$

If $SK_{id,S}$ passes the key sanity check, the algorithm outputs 1. Otherwise, it outputs 0.

If the sold key is proved valid, the identity of the true key hold can be easily discovered through two decryption computations on the component D_1 of the secret key $SK_{id,S}$. The $Trace$ algorithm is implemented by KGC using the master secret key MSK.

- $Trace(MSK, SK_{id,S}) \rightarrow id/\perp$. If the output of $KeySanityCheck$ algorithm is 0, it means that $SK_{id,S}$ is not a well-formed secret key and not worth to be traced. The $Trace$ algorithm outputs $\perp$. Otherwise, $SK_{id,S}$ is a well-formed secret key. The algorithm extracts $(\zeta||\theta) = SDec_{k_2}(D_1)$. The malicious user's identity id can be recovered by computing $id = SDec_{k_1}(\zeta)$.

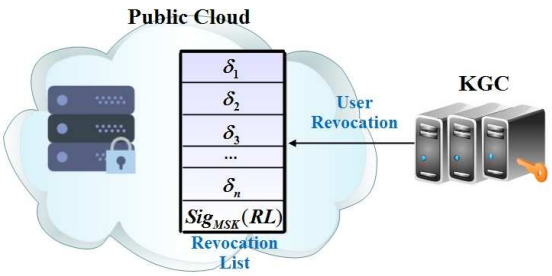


Fig. 3: User Revocation

4.2.8 User Revocation

After the traitor is identified in the tracing algorithm, an important issue is how to revoke the search and decryption privilege of the traced secret key. Taking the advantage of the elaborate secret key design, the KGC can easily revoke user's access right in LiST. The component $D_2 = \delta$ of secret key contains the identity information of user. Moreover, it must be submitted to public cloud server as a component $T_2 = D_2$ of keyword trapdoor to issue a data retrieval request. As shown in Fig. 3, the KGC could simply put $D_2 = \delta$ into the revocation list to realize the user revocation. The revocation list should be stored (together with a signature

signed by KGC) in the public cloud server. When the public cloud receives a keyword trapdoor T_{KW}, it should firstly check whether $T_2 = D_2 = \delta$ is included in the revocation list. If yes, the data retrieval request is rejected.[1]

Discussion: To realize traitor tracing function in LiST, an identification element $D_2 = \delta$ is generated from user's identity. It is also implicitly embedded into another element $D_1 = g^{\frac{\alpha - ar}{\lambda + \delta}}$ of secret key. When a malicious user wants to sell his secret key without being traced, he may intend to re-randomize the element D_2 to remove his identifier information from the secret key. Although he can easily generate a new $D_2' = \delta'$, it is impossible for him to generate a valid $D_1' = g^{\frac{\alpha - ar}{\lambda + \delta'}}$ without the master secret keys α and λ. The construction of element D_1 frustrates his vicious attempt.

According to the performance analysis in Section 6, the LiST system is much more efficient than the related crypto schemes [6], [7], [9]-[14], [27], [28]. If one simply combines several crypto schemes together, the performance of the combined scheme will not be better than the original underlying schemes, as adding extra functions to an existing scheme introduces extra computation cost. Since the performance of the original scheme is worse than the LiST scheme (shown in Section 6), the combined scheme will perform even worse. On the other hand, the security of a combined scheme cannot be guaranteed unless a formal proof is provided. Since different public key cryptosystems may be proved secure based on different hardness problems, it is challenging to prove that a combined scheme is secure based on a single hardness problem. Moreover, simply combining an existing public key encryption scheme with keyword search is vulnerable to ciphertext swapping attack [40], in which an attacker swaps the encrypted keyword indices associated with encrypted files. Since different crypto schemes utilize different random numbers to encrypt data, the encrypted keyword index and file are bound together. The ciphertext swapping attack is inevitable in such a combined system.

In fact, the LiST system leverages the systematic design philosophy to construct a lightweight sharable and traceable secure system that is tailored for the mobile health application. It not only has great advantages of computation and communication overheads over the other crypto primitives, but also ensures the security of the whole system. These merits are achieved by the integrated architecture and delicate algorithm design, which can not be realized by the simple combination.

5 Security Analysis

In accordance with the security requirements defined in Section 3.3, we prove that LiST is IND-CKCCA secure and satisfies the traceability. Both security requirements are formally proved without random oracle. Then, we analyze that the LiST system is secure against collusion attack.

1. The identities of the revoked users do not need to be stored in the revocation list since the element D_2 can be utilized to identify the traitor.

Theorem 1. If the decisional bilinear Diffie-Hellman (DBDH) assumption holds, LiST is IND-CKCCA secure.

Proof: Due to the length limitation, the concrete proof of Theorem 1 is given in Supplemental Materials C.

High Level Idea of the Proof: In the security proof of theorem 1, the challenger $\mathcal{C}$ and polynomial time adversary $\mathcal{A}$ interact with each other. If $\mathcal{A}$ could break the IND-CKCCA security of the LiST system, $\mathcal{C}$ then utilizes the interactive game with $\mathcal{A}$ to solve the DBDH problem. The game is briefly described as below.

(1) In the setup phase, $\mathcal{C}$ is given an instance of the DBDH assumption $(\vec{y}, T)$, where $\vec{y} = (g, g^a, g^b, g^s)$, and $T = e(g,g)^{abs}$ or T is a random element in $\mathbb{G}_T$. Then, $\mathcal{C}$ utilizes the received elements to construct the public parameter of the system.

(2) In the query phase 1, $\mathcal{A}$ adaptively issues the secret key queries and trapdoor queries. $\mathcal{C}$ responds on these queries and returns the corresponding secret keys or trapdoors to $\mathcal{A}$.

(3) In the challenge phase, $\mathcal{A}$ sends the challenge access policy (M^*, ρ^*), two keywords (KW_0^*, KW_1^*) and two messages (m_0^*, m_1^*) to $\mathcal{C}$. The restriction is that the secret key of attribute set S that satisfies (M^*, ρ^*) has not been queried in query phase 1. Then, $\mathcal{C}$ flips random coins $\mu_1, \mu_2 \in \{0,1\}$, and encrypts the message $m_{\mu_1}^*$ and the keyword $KW_{\mu_2}^*$. A key point is that the element T should be embedded in the generated challenge ciphertext CT^*, which is sent to $\mathcal{A}$.

(4) Receiving the challenge ciphertext CT^*, the adversary $\mathcal{A}$ continues to issue queries as in phase 1. The restriction is that $KW \notin \{KW_0^*, KW_1^*\}$ and S does not satisfy (M^*, ρ^*).

(5) In the guess phase, $\mathcal{A}$ outputs a guess $\mu_1', \mu_2' \in \{0,1\}$. If $\mu_1' = \mu_1, \mu_2' = \mu_2$, then $\mathcal{C}$ outputs 1 meaning $T = e(g,g)^{abs}$. Otherwise, it outputs 0 meaning T is a random element in $\mathbb{G}_T$.

If $\mathcal{C}$ wins the game with non-negligible probability, then $\mathcal{A}$ could utilize the guess of $\mathcal{C}$ to solve the DBDH problem with non-negligible probability. However, since DBDH problem is intractable by polynomial time algorithm, then the system is IND-CKCCA secure. □

Theorem 2: The proposed LiST system is $(\tilde{t}, \epsilon)$ traceable under the $(q, \tilde{t}', \epsilon')$-SDH assumption with $\epsilon' = \epsilon, \tilde{t}' \geq \tilde{t} + t_e \cdot [\ O(|S|)q_{sk}\ ]$, where q_{sk} denotes the total numbers of user secret key queries, t_e denotes the running time of an exponentiation, $|S|$ is the number of attributes in a set S.

Proof: The concrete proof of Theorem 2 is given in Supplemental Materials C.

High Level Idea of the Proof: In the security proof of theorem 2, the challenger $\mathcal{C}$ and polynomial time adversary $\mathcal{A}$ interact with each other. If $\mathcal{A}$ could break the traceability of the LiST system, $\mathcal{C}$ then utilizes $\mathcal{A}$ to solve the q-SDH problem. The game is briefly described as below.

(1) In the setup phase, $\mathcal{C}$ is given an instance of the q-SDH assumption $(\hat{g}, \hat{g}^{\lambda}, \hat{g}^{\lambda^2}, \cdots, \hat{g}^{\lambda^q})$. Then, $\mathcal{C}$ utilizes the received elements to construct the public parameter of the system.

(2) In the query phase 1, $\mathcal{A}$ adaptively issues the secret key queries. $\mathcal{C}$ responds on these queries and returns the corresponding secret keys to $\mathcal{A}$.

(3) In the challenge phase, $\mathcal{A}$ sends a challenge secret key SK^* to $\mathcal{C}$. If SK^* passes the key sanity check, it indicates that $\mathcal{A}$ successfully forges a valid secret key and breaks the traceability of the LiST system. Then, $\mathcal{C}$ utilizes the SK^* to construct a tuple (c^*, w^*) to solve the q-SDH problem.

If $\mathcal{C}$ wins the game with non-negligible probability, then $\mathcal{A}$ could utilize the the forged secret key of $\mathcal{C}$ to solve the q-SDH problem with non-negligible probability. However, since q-SDH problem is intractable by polynomial time algorithm, then the system is IND-CKCCA secure. □

Keyword Guessing Attack: The proposed LiST system (as well as [27], [28]) can not resist such attack. However, [36], [37] proposed an effective way to prevent the attack. In their schemes [36], [37], the storage server has its own public/secret key pair. The server's public key is involved in the keyword trapdoor generation algorithm such that the test algorithm can only be executed by the server with the help of its secret key. The similar skill can also be leveraged in LiST to resist keyword guessing attack.

Resistance to collusion attack. Collusion attack is an important type of attack in multi-user system. The authorized users may collude with each other in order to get extra privileges. However, our system is not vulnerable to such attack. In the key generation algorithm, the KGC selects a set of random numbers to create user's secret key. The collusive users are not able to combine their secret keys to generate a new valid secret key, since the secret keys generated from different random numbers are not compatible with each other. Therefore, the LiST system is secure against collusion attack.

6 Performance Analysis

In this section, we compare LiST with other existing schemes in terms of storage overhead and computation cost. The proposed LiST is also implemented using the PBC library [38] on both PC and mobile device platforms.

TABLE 2: Function Comparison with Other Schemes

Sch.	Access Control	Search	Out. Dec.	Ver. Dec.	W.B. Trace	User Revoke
[6]	√	×	×	×	×	×
[7]	√	×	×	×	×	×
[9]	√	×	√	×	×	×
[10]	√	×	√	√	×	×
[11]	√	×	√	√	×	×
[12]	√	×	√	√	×	×
[13]	√	×	×	×	√	×
[14]	√	×	×	×	√	×
[27]	√	√	×	×	×	×
[28]	√	√	×	×	×	√
LiST	√	√	√	√	√	√

6.1 Comparison

Tables 2-4 compare the function, storage and computation overhead of LiST with other schemes that could exert access control on the encrypted data.

Table 2 indicates that the schemes [27], [28] provide keyword search function. Outsourced decryption (for short, Out. Dec.) is achieved in [9] and verifiable decryption (for short, Ver. Dec.) is dealt with in [10], [11], [12]. The schemes in [13], [14] achieves white-box (for short, W. B.) traceability. The user revocation is realized in [28] using a large scale key update approach. In LiST, all these functions are supported with much less transmission and computation cost, which will be analyzed in the following.

First of all, we define the notations used in Table 3-4. Let $|PP|, |SK|, |CT|, |T_{KW}|$ be the sizes of the public parameter, secret key of user, the ciphertext and the keyword trapdoor, respectively. Let l be the number of rows in matrix M of access structure, $|S|$ be the size of attribute set S and $|U|$ be the size of the universe attribute set U. $|\mathbb{G}|$, $|\mathbb{G}_T|$ and $|\mathbb{Z}_p|$ represent the bit length of an element in group $\mathbb{G}$, $\mathbb{G}_T$ and $\mathbb{Z}_p$, respectively. Denote t_{e1}, t_{e2} and t_p as the times consumed for a modular exponentiation on group $\mathbb{G}$, a modular exponentiation on group $\mathbb{G}_T$ and a bilinear pairing operation, respectively.

The storage overhead comparison is shown in Table 3. The comparison shows that LiST has smaller public parameter size, secret key size, ciphertext size and trapdoor size. The detailed analyzing is listed as following.

- ***Public Parameter Size***: It is easy to find that LiST and [9], [14] supports unbounded number of attributes in the system, which is also referred to as "large universe" in ABE schemes. This feature will be very helpful for large scale mHealth network since the size of public parameter is immutable with the size of attribute set. However, the public parameter sizes of the other schemes [6], [7], [10], [11], [12], [13], [27], [28] linearly grow with $|U|$, which is the total number of the attributes in the system. With the expansion of the system, more and more new attributes will emerge. These schemes have to rebuilt the whole system to accommodate these new attributes. Thus, they are not practical for the mHealth system.
- ***Secret Key Size***: The secret key of user in LiST consists of $|S| + 1$ elements in group $\mathbb{G}$ and seven elements in $\mathbb{Z}_p$. Generally speaking, $|\mathbb{Z}_p|$ is always smaller than $|\mathbb{G}|$, which will be further analyzed in the experiments in Sec. 6.2. It is obvious that the size of user's secret key in LiST is smaller compared with the schemes in [6], [14], [27], [28]. Smaller secret key size also means smaller storage overhead in user's resource-limited mobile devices.
- ***Ciphertext Size***: The ciphertext generated in LiST has 2 elements in group $\mathbb{G}$, 2 elements in group $\mathbb{G}_T$ and $2l$ elements in $\mathbb{Z}_p$. Since the $|\mathbb{Z}_p|$ is typically at least one sixth of $|\mathbb{G}|$, the $|CT|$ in LiST is smaller than all the other schemes in Table 3. Thus, LiST requires smaller storage overhead in the public cloud and lower transmission cost between data owner and public cloud.
- ***Trapdoor Size***: The schemes in [6], [7], [9], [10], [11], [12], [13], [14] do not support keyword search on encrypted data and do not have trapdoor generation algorithm. In LiST, the keyword trapdoor generated by user consists of only 7 group elements in $\mathbb{Z}_p$, which is much smaller than the schemes in [27], [28]. From another perspective, it will greatly reduce the transmission overhead between data user and public

TABLE 3: Storage Overhead Comparison

Scheme	$\|PP\|$	$\|SK\|$	$\|CT\|$	$\|T_{KW}\|$
[6]	$(2\|U\|+3)\|\mathbb{G}\|$	$(5\|S\|)\|\mathbb{G}\|$	$(2l+1)\|\mathbb{G}\|+\|\mathbb{G}_T\|$	$\perp$
[7]	$(2\|U\|+4)\|\mathbb{G}\|+\|\mathbb{G}_T\|$	$(\|S\|+6)\|\mathbb{G}\|$	$(5l)\|\mathbb{G}\|+\|\mathbb{G}_T\|$	$\perp$
[9]	$2\|\mathbb{G}\|+\|\mathbb{G}_T\|$	$(\|S\|+2)\|\mathbb{G}\|$	$(2l+1)\|\mathbb{G}\|+\|\mathbb{G}_T\|$	$\perp$
[10]	$(\|U\|+5)\|\mathbb{G}\|+\|\mathbb{G}_T\|$	$(\|S\|+2)\|\mathbb{G}\|$	$(4l+3)\|\mathbb{G}\|+2\|\mathbb{G}_T\|$	$\perp$
[11]	$(\|U\|+2)\|\mathbb{G}\|+\|\mathbb{G}_T\|$	$(\|S\|+2)\|\mathbb{G}\|$	$(2l+1)\|\mathbb{G}\|+\|\mathbb{G}_T\|$	$\perp$
[12]	$(\|U\|+4)\|\mathbb{G}\|+\|\mathbb{G}_T\|$	$(\|S\|+2)\|\mathbb{G}\|$	$(2l+1)\mathbb{G}\|+\|\mathbb{G}_T\|$	$\perp$
[13]	$(\|U\|+3)\|\mathbb{G}\|+\|\mathbb{G}_T\|$	$(\|S\|+3)\|\mathbb{G}\|+\|\mathbb{Z}_p\|$	$(2l+2)\|\mathbb{G}\|+\|\mathbb{G}_T\|$	$\perp$
[14]	$6\|\mathbb{G}\|+\|\mathbb{G}_T\|$	$(2\|S\|+3)\|\mathbb{G}\|+\|\mathbb{Z}_p\|$	$(3l+2)\|\mathbb{G}\|+\|\mathbb{G}_T\|$	$\perp$
[27]	$(2\|U\|+10)\|\mathbb{G}\|+3\|\mathbb{G}_T\|$	$(3\|S\|)\|\mathbb{G}\|$	$(l+3)\|\mathbb{G}\|+2\|\mathbb{G}_T\|$	$(3\|S\|)\|\mathbb{G}\|$
[28]	$(3\|U\|+1)\|\mathbb{G}\|+\|\mathbb{G}_T\|$	$(2\|S\|+1)\|\mathbb{G}\|+\|\mathbb{Z}_p\|$	$(l+2)\|\mathbb{G}\|$	$(2\|S\|+2)\|\mathbb{G}\|$
LiST	$3\|\mathbb{G}\|+2\|\mathbb{G}_T\|$	$(\|S\|+1)\|\mathbb{G}\|+7\|\mathbb{Z}_p\|$	$2\|\mathbb{G}\|+2\|\mathbb{G}_T\|+2l\|\mathbb{Z}_p\|$	$7\|\mathbb{Z}_p\|$

TABLE 4: Computation Overhead Comparison

Scheme	$KeyGen$	Enc	Dec	$Trapdoor$	$Test$	$KeySanityCheck$ &$Trace$
[6]	$(6\|S\|)t_{e_1}$	$t_p+t_{e_2}$ $+(2l+1)t_{e_1}$	$(3\|S\|)t_p$ $+\|S\|t_{e_1}+2\|S\|t_{e_2}$	$\perp$	$\perp$	$\perp$
[7]	$(\|S\|+9)t_{e_1}$	$t_p+(6l)t_{e_1}$ $+l\cdot t_{e_2}$	$(6\|S\|)t_p$ $+(2\|S\|)t_{e_2}$	$\perp$	$\perp$	$\perp$
[9]	$(\|S\|+2)t_{e_1}$	$t_p+(3l+1)t_{e_1}+t_{e_2}$	t_{e_1}	$\perp$	$\perp$	$\perp$
[10]	$(\|S\|+3)t_{e_1}$	$2t_p+(6l+4)t_{e_1}+2t_{e_2}$	t_{e_1}	$\perp$	$\perp$	$\perp$
[11]	$(\|S\|+3)t_{e_1}$	$t_p+(3l+1)t_{e_1}+t_{e_2}$	t_{e_1}	$\perp$	$\perp$	$\perp$
[12]	$(\|S\|+3)t_{e_1}$	$t_p+(3l+3)t_{e_1}+t_{e_2}$	t_{e_1}	$\perp$	$\perp$	$\perp$
[13]	$(\|S\|+4)t_{e_1}$	$t_p+t_{e_2}+$ $(3\|l\|+2)t_{e_1}$	$(2\|S\|+1)t_p+\|S\|t_{e_2}$ $+(\|S\|+1)t_{e_1}$	$\perp$	$\perp$	$(2\|S\|+5)t_p+t_{e_2}$ $+(\|S\|+3)t_{e_1}$
[14]	$(4\|S\|+4)t_{e_1}$	$t_p+t_{e_2}+$ $(5\|l\|+2)t_{e_1}$	$(3\|S\|+1)t_p+\|S\|t_{e_2}$ $+(\|S\|+1)t_{e_1}$	$\perp$	$\perp$	$(4\|S\|+5)t_p+(\|S\|+1)t_{e_2}$ $+(\|S\|+4)t_{e_1}$
[27]	$4\|S\|t_{e_1}$	$2t_p+2t_{e_2}+$ $(l+5)t_{e_1}$	$4t_p+t_{e_2}+$ $(3l+4)t_{e_1}$	$8\|S\|t_{e_1}$	$2t_p+$ $(2l)t_{e_1}$	$\perp$
[28]	$(2\|S\|+2)t_{e_1}$	$(2\|S\|+2)t_{e_1}$	$\perp$	$(2\|S\|+2)t_{e_1}$	$(\|S\|+1)t_p$ $+t_{e_2}$	$\perp$
LiST	$(\|S\|+1)t_{e_1}$	$2t_{e_1}+3t_{e_2}$	t_{e_1}	$0t_p+0t_{e_1}$ $+0t_{e_2}$	$3t_p+t_{e_2}+$ $(2l+4)t_{e_1}$	$3t_p+2t_{e_2}+$ $(\|S\|+1)t_{e_1}$

cloud and decrease the energy consumption of user's mobile terminal compared with that in [27], [28].

As shown in Table 4, LiST has lower computation overhead in each algorithms compared with other schemes.

- $KeyGen$: In the $KeyGen$ algorithm, the KGC could utilize $|S|+1$ exponentiation operations on group $\mathbb{G}$ to generate user's secret key. All other schemes in Table 3 requires more computation than ours. For instance, [6], [14] and [27] needs as much as $6|S|$, $4|S|+4$ and $4|S|$ exponentiation calculations on group $\mathbb{G}$ in $KeyGen$ computation, respectively.
- Enc: In LiST, the EHR encryption is done by user's energy limited device. In real-time monitoring medical care system, the health information will be continuously generated, which should be immediately encrypted and transmitted to the public cloud. If the encryption computation cost is too large, the power of data owner's wireless terminal will be consumed very quickly. In LiST, no pairing computation is involved in the encryption algorithm. Moreover, only two exponentiations on group $\mathbb{G}$ and three exponentiations on group $\mathbb{G}_T$ are required to generate a ciphertext. Other schemes in Table 3 need much more calculation cost in Enc algorithm, which also grow with the number of attributes.
- Dec: Utilizing the outsourced decryption mechanism, the schemes in [9], [10], [11], [12] and LiST can recover the EHR with only one exponentiation computation on group $\mathbb{G}$. However, they [9], [10], [11], [12] have not realized keyword search function on encrypted data. On the other hand, the schemes in [6], [7], [13], [14], [27] consume a lot of time and energy to execute the large amount of pairing and exponentiation computations.
- $Trapdoor$: In Table 4, it is shown that the computation overhead of $Trapdoor$ algorithm in LiST is $0t_p+0t_{e_1}+0t_{e_3}$. It means that no pairing or exponentiation computations is required in LiST. In fact, only a few lightweight multiplication, division and inversion computations on $\mathbb{Z}_p$ are required in LiST. The computation time of these operations can almost be ignored compared with the pairing and exponentiation operations (shown in Table 6). On the contrary, the schemes in [27], [28] need a large amount of exponentiation calculations to generate a keyword trapdoor, which will consume a lot of energy of user's mobile device.
- $Test$: In the $Test$ algorithm, the scheme in [27] needs a little bit less computations than ours. However, [27] puts heavy computation burden to user's terminal in the decryption phase. The main principle of LiST is alleviating user's computation burden and migrating the heavy computation to public cloud, which processes stronger computation power and continuous energy supply.
- $KeySanityCheck\&Trace$: The schemes in [13] and

[14] take the traitor tracing problem into consideration. However, a large amount of pairing computations are required to recover the traitor's identity in these two schemes [13], [14]. In LiST, three pairing computations is needed in the tracing procedure.

6.2 Experimental Analysis

We conduct experiments on both PC and smart phone to evaluate the performance of LiST. The PC is utilized to simulate the public cloud and KGC, which process relatively high computation capability and endless electricity supply. The smart phone is regarded as the mobile terminals of data owner or data user, which has low computation resources and limited battery.

6.2.1 Experiment Settings

We leverage Stanford Pairing Based Cryptography (PBC) Library [38] on PC to implement LiST and other available schemes used for comparison. C programming language is used for prototyping of the schemes. The PC used for conducting experiment is running Windows 7 64-bit operation system with the following configurations: Intel Core^{TM} i3-2120 CPU @ 3.30 GHz, 4.00 GB RAM.

We use Java Pairing Based Cryptography (JPBC) Library [38] to test LiST on smart phone, which utilizes Java programming language for the scheme coding. The smart phone has a 64-bit 8core CPU processor (4core processor runs at 1.5 GHz and 4core processors runs at 1.2 GHz), 3GB RAM. The experiment is built on the platform Android 5.1.1.

For both PC and smart phone, type A elliptic curve of 160-bit group order in PBC library [38] is chosen for conducting experiment, which is equivalent to 80-bit security level [39]. It has the expression form $E : y^2 = x^3 + x$ over $\mathbb{F}_q$ finite field. Both group $\mathbb{G}$ and group $\mathbb{G}_T$ have order p and are subgroups of $E(\mathbb{F}_q)$. The parameters q and p are equivalent to 512 bits and 160 bits numbers in binary system, respectively. Then, we have $|\mathbb{Z}_p| = 160$ bits, $|\mathbb{G}| = 1024$ bits and $|\mathbb{G}_T| = 1024$ bits.

TABLE 5: Storage Overhead (bits) ($|S| = 100$)

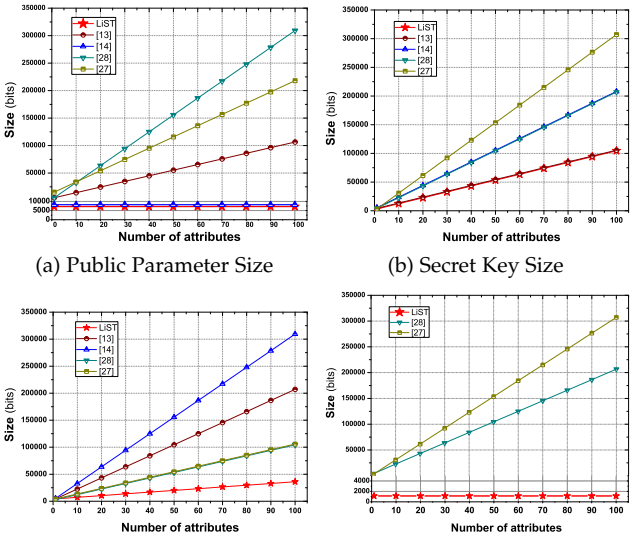
Scheme	[13]	[14]	[27]	[28]	**LiST**
$\|PP\|$	106,496	8,192	218,112	309,248	**7,168**
$\|SK\|$	105,632	208,032	307,200	207,008	**104,544**
$\|CT\|$	207,008	309,408	105,792	104,448	**36,096**
$\|T_{KW}\|$	$\perp$	$\perp$	307,200	206,848	**1,120**

6.2.2 Experiment Results

(1) **Storage and Transmission Efficiency.**

To evaluate the storage and transmission overhead, we compare LiST with the schemes [13], [14], [27], [28] in terms of the public parameter size, secret key size and trapdoor size (bits) in Figure 4 and Table 5. In order to describe the performance, a non-uniform axis is used in Fig. 4(a) to make the different values much clearer.

- Fig. 4(a) shows that LiST requires much less storage space and transmission cost for the public parameter. No matter how many attributes are accommodated in the mHealth system, the public parameter size is 7,168 bits. However, [28] needs 309,248 bits when the total number of attributes in the system grows to 100, which is 43 times of ours parameter size. Since the PP should be stored in each device in the system, the mobile terminals of user has to use a large storage space for the schemes in [13], [27], [28].



(a) Public Parameter Size (b) Secret Key Size

(c) Ciphertext Size (d) Trapdoor Size

Fig. 4: Storage and transmission overhead

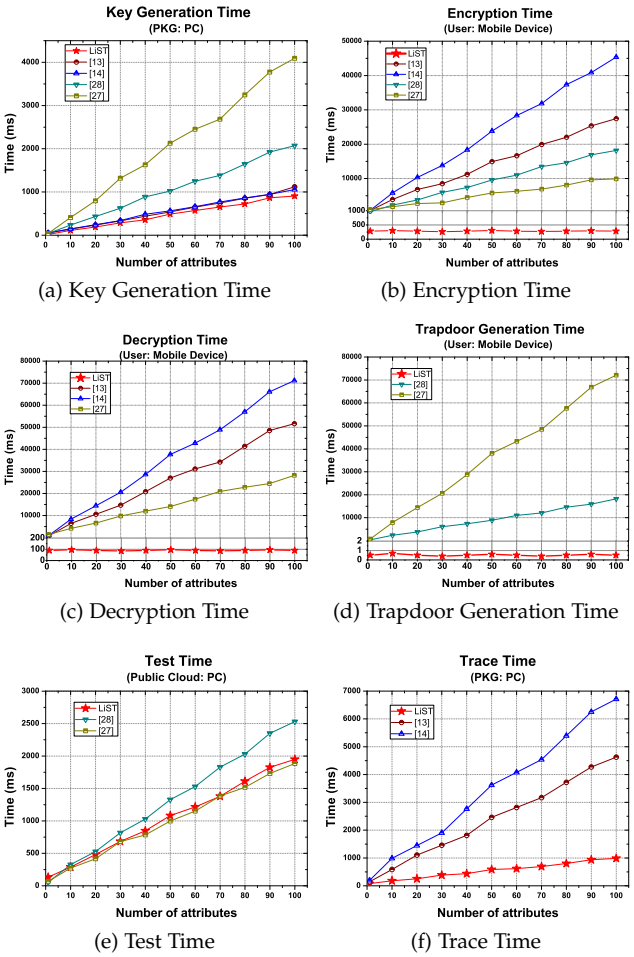
- In Fig. 4(b), it looks like that there are only three lines for five schemes. The fact is that the schemes in [14] and [28] has similar size of SK and these two lines seem overlap. Moreover, LiST and [13] has similar $|SK|$ and the lines overlaps with each other. When $|S| = 100$, the schemes in [27] requires 307,200 bits storage space for the secret key storage, which is tripled of ours.
- Fig. 4(c) indicates that LiST requires the least storage space for ciphertext, which is very useful to save money in the pay-for-use mode of cloud. Moreover, the data owner could consume less battery to transmit the ciphertext to the public cloud and prolong the service time of user's mobile devices.
- In Fig. 4(d), there are only three lines since the schemes in [13], [14] do not provide keyword search function. The keyword trapdoor generated in LiST only has 1,120 bits and will not grow with the number of the attribute set. It is much smaller than that in [28] (206,848 bits) and [27] (307,200 bits) when $|S| = 100$. The expensive transmission overhead in [27], [28] will quickly drain the battery of user's wireless terminal.

TABLE 6: Computation Time on Different Platforms (ms)

	PC	**Smart Phone**
Bilinear Pairing	18.025	195.106
Exponentiation on group $\mathbb{G}$	9.175	90.118
Exponentiation on group $\mathbb{G}_T$	2.784	33.4
Multiplication on $\mathbb{Z}_p$	0.001	0.026
Division on $\mathbb{Z}_p$	0.005	0.093
Inversion on $\mathbb{Z}_p$	0.004	0.057

TABLE 7: Computation Time (ms) ($|S| = 100$)

Scheme	[13]	[14]	[27]	[28]	**LiST**
$KeyGen$	1,118	1,055	4,093	2,072	**906.67**
Enc	27,444	45,377	9,919	18,203	**280.43**
Dec	51,657	71,167	28,209	$\perp$	**90.11**
$Trapdoor$	$\perp$	$\perp$	72,094	18,203	**0.501**
$Test$	$\perp$	$\perp$	1,885	2,530	**1,949**
$Trace$	4,624	6,712	$\perp$	$\perp$	**985.91**



(a) Key Generation Time (b) Encryption Time

(c) Decryption Time (d) Trapdoor Generation Time

(e) Test Time (f) Trace Time

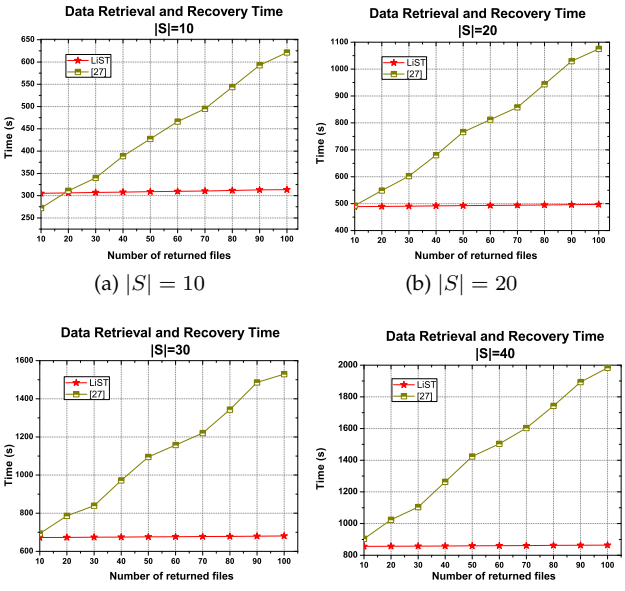
Fig. 5: Computation overhead

(2) **Computation Efficiency.**

Shown in Figure 5, we have implemented each algorithm in LiST and that in [13], [14], [27], [28]. The computations operated by KGC and public cloud are tested on PC, while the calculations of data owner and data user are executed by the smart phone. The experiments on both PC and smart phone (shown in Table. 6) indicate that the same basic computations (such as bilinear paring and exponentiation) executed by PC is about 10 times faster than that on smart phone .

The experimental results shown in Figure. 5 and Table 7 clearly demonstrate that LiST always has least computation time compared with others. Especially for the algorithms that are executed by user's wireless devices, LiST has incomparable efficiency advantage. A non-uniform axis is also used in Fig. 5 for clear description.

- As indicated in Table 7, our encryption time is about 280.43 ms and not varies with the number of attributes. However, the EHR encryption time consumes 27,444 ms, 45,377 ms, 9,919 ms and 18,203 ms in [13], [14], [27], [28], respectively.
- In the Dec algorithm, LiST has a constant computation cost 90.11ms to complete a decryption operation. However, the schemes in [13], [14], [27] require 51,657 ms, 71,167 ms and 28,209 ms to recover a message, respectively. The large computation overhead will quickly consumes the battery of user's resource limited mobile device.
- The $Trapdoor$ algorithm in LiST requires no bilinear paring or exponentiation computations. Only multiplication, division and inversion operations on $\mathbb{Z}_p$ are calculated in smart phone, which consume only 0.026 ms, 0.093 ms and 0.057 ms, respectively. When $|S| = 100$, LiST only requires 0.501 ms to complete a keyword trapdoor generation algorithm, while the schemes in [27], [28] needs 72,094 ms and 18,203 ms.



(a) $|S| = 10$ (b) $|S| = 20$

(c) $|S| = 30$ (d) $|S| = 40$

Fig. 6: Data Retrieval and Recovery Time Comparison

The above analysis shows that LiST has efficiency significantly better than the other schemes. The only exception is that the computation cost of $Test$ algorithm in [27] is a little bit better than ours. However, [27] spends much more time in the decryption time compared with ours. It is important to evaluate the time between sending out a keyword trapdoor query and obtaining the recovered health documents, which is deemed as user's waiting time.

In the following, we compare the data retrieval and recovery time (waiting time) of LiST and that in [27] in Figure 6. Assume the public cloud executes data retrieval operations on 1000 encrypted EHRs. Various quantity of matched files will be returned and decrypted by user's terminal. The number of matched files varies from 10 to 100 in Figure 6. Moreover, distinct values of $|S|$ are considered in Figure 6.a-6.d.

It is notable that the data retrieval and recovery time of LiST is far less than that in [27]. With the increasing of the number of matched files, the waiting time in [27] grows rapidly. For example, when $|S| = 40$ (shown in Figure 6.d), the data retrieval and recovery time in [27] varies from 903 seconds to 1,982 seconds when the number of matching files grows from 10 to 100. On the contrary, the waiting time in LiST varies from 855 seconds to 863 seconds. Needless to say, from the user's viewpoint, LiST has much better performance than that in [27].

7 Conclusion

In this paper, we proposed LiST, a lightweight secure data sharing solution with traceability for mHealth systems. LiST seamlessly integrates a number of key security functionalities, such as fine-grained access control of encrypted data, keyword search over encrypted data, traitor tracing, and user revocation into a coherent system design. Considering that mobile devices in mHealth are resource constrained, operations in data owners' and data users' devices in LiST are kept at lightweight. We formally defined the security of LiST and proved its security without random oracle. The qualitative analysis showed that LiST is superior to most of the existing systems. Extensive experiments on its performance (on both PC and mobile device) demonstrated that LiST is very promising for practical applications.

Acknowledgments

This work is supported by National Natural Science Foundation of China under Grant No. 61402112, 61472307, 61472309, 61502086; the Singapore National Research Foundation under the NCR Award Number NRF2014NCR-NCR001-012; AXA Research Fund; Fujian Provincial Key Laboratory of Information Processing and Intelligent Control (Minjiang University) MJUKF201734; Fujian Major Project of Regional Industry 2014H4015; and Major Science and Technology Project of Fujian Province under Grant No. 2015H6013.

References

[1] L. Guo, C. Zhang, J. Sun, Y. Fang. "A privacy-preserving attribute based authentication System for Mobile Health Networks," IEEE Transactions on Mobile Computing, 2014, vol. 13, no. 9, pp. 1927-1941.

[2] A. Abbas, S. Khan, "A review on the state-of-the-art privacy preserving approaches in e-health clouds," IEEE Journal of Biomedical Health Informatics, 2014, vol. 18, pp. 1431-1441.

[3] J. Yang, J. Li, Y. Niu, "A hybrid solution for privacy preserving medical data sharing in the cloud environment," Future Generation Computer Systems, 2015, vol. 43-44, pp. 74-86.

[4] http://www.pbs.org/newshour/updates/has-health-care-hacking-become-an-epidemic/.

[5] V. Goyal, O. Pandey, A. Sahai, B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," Proc. 13th ACM Conf. Computer and Comm. Security (CCS'06), pp. 89-98, 2006.

[6] R. Ostrovsky, A. Sahai, B. Waters, "Attribute-based encryption with nonmonotonic access structures," in: Proceedings of the 14th ACM Conference on Computer and Communications Security, ACM, 2007, pp. 195-203.

[7] J. Han, W. Susilo, Y. Mu. "Improving privacy and security in decentralized ciphertext-policy attribute-based encryption," IEEE Transactions on on Information Forensics and Security, 2015, vol. 10, no. 3, 665-678.

[8] M. Li, S. Yu, Y. Zheng, K. Ren, W. Lou. "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption," IEEE transactions on parallel and distributed systems, 2013, 24(1): 131-143.

[9] M. Green, S. Hohenberger, B. Waters, "Outsourcing the decryption of ABE ciphertexts," in Proc. USENIX Security Symp., San Francisco, CA, USA, 2011.

[10] J. Lai, R. H. Deng, C. Guan, J. Weng, "Attribute-based encryption with verifiable outsourced decryption," IEEE Trans. Inf. Forensics Security, vol. 8, no. 8, pp. 1343-1354, Aug. 2013.

[11] B. Qin, R. H. Deng, S. Liu, S. Ma, "Attribute-based encryption with efficient verifiable outsourced decryption," IEEE Trans. Inf. Forensics Security, vol. 10, no. 7, pp. 1384-1394, JULY. 2015.

[12] X. Mao, J. Lai, Q. Mei, K. Chen, J. Weng, "Generic and efficient constructions of attribute-based encryption with verifiable outsourced decryption," IEEE Transactions on Dependable and Secure Computing, publish online, DOI: 10.1109/TDSC.2015.2423669.

[13] Z. Liu, Z. Cao, D. Wong, "White-box traceable ciphertext-policy attribute-based encryption supporting any monotone access structures," IEEE Transactions on Information Forensics and Security, 2013, 8(1), 76-88.

[14] J. Ning, X. Dong, Z. Cao, L. Wei, X. Lin, "White-box traceable ciphertext-policy attribute-based encryption supporting flexible attributes," IEEE Transactions on Information Forensics and Security, 2015, 10(6), 1274-1288.

[15] Z. Liu, Z. Cao, D. Wong, "Traceable CP-ABE: how to trace decryption devices found in the wild," IEEE Transactions on Information Forensics and Security, 2015, 10(1), 55-68.

[16] D.X. Song, D. Wagner, A. Perrig, "Practical techniques for searches on encrypted data", in: IEEE Symposium on Security and Privacy, 2000, pp. 44-55.

[17] B. Wang, S. Yu, W. Lou, and Y. T. Hou, "Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud," in Proc. IEEE INFOCOM, Apr./May 2014, pp. 2112-2120.

[18] E. Stefanov, C. Papamanthou, and E. Shi, "Practical dynamic searchable encryption with small leakage," in Proc. NDSS, Feb. 2014.

[19] Y. Yang, H. Li, W. Liu, H. Yang, and M. Wen, "Secure dynamic searchable symmetric encryption with constant document update cost," in Proc. GLOBECOM, Anaheim, CA, USA, 2014.

[20] D. J. Park, K. Kim, and P. J. Lee, "Public key encryption with conjunctive field keyword search," in Information Security Applications, 5th International Workshop, WISA 2004, Jeju Island, Korea, August 23- 25, 2004, Revised Selected Papers, ser. Lecture Notes in Computer Science, vol. 3325. Springer, 2004, pp. 73-86.

[21] H. S. Rhee, J. H. Park, W. Susilo, and D. H. Lee, "Improved searchable public key encryption with designated tester," in Proceedings of the 2009 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2009, Sydney, Australia, March 10-12, 2009. ACM, 2009, pp. 376C379.

[22] Z. Lv, C. Hong, M. Zhang, and D. Feng, "Expressive and secure searchable encryption in the public key setting," in Information Security - 17th International Conference, ISC 2014, Hong Kong, China, October 12-14, 2014. Proceedings, ser. Lecture Notes in Computer Science, vol. 8783. Springer, 2014, pp. 364-376.

[23] Y. Yang and M. Ma, "Conjunctive Keyword Search With Designated Tester and Timing Enabled Proxy Re-Encryption Function for E-Health Clouds," IEEE Transactions on on Information Forensics and Security, 2016, vol. 11, no. 4, 746-759.

[24] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in Proc. Int. Conf. Theory Appl. Cryptograph. Techn., Adv. Cryptol. (EUROCRYPT), vol. 3027. Interlaken, Switzerland, May 2004, pp. 506-522.

[25] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," in Proc. 13th ACM Conf. Comput. Commun. Security, 2006, pp. 79-88.

[26] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in Proc. 4th Theory Cryptogr. Conf. (TCC), vol. 4392. Amsterdam, The Netherlands, Feb. 2007, pp. 535-554.

[27] K. Liang, W. Susilo, "Searchable Attribute-Based Mechanism with Efficient Data Sharing for Secure Cloud Storage," IEEE Transactions on Information Forensics and Security, 2015, vol. 10, no. 9, pp. 1981 - 1992.

[28] W. Sun, S. Yu, W. Lou, Y. Hou and H. Li, "Protecting Your Right: Verifiable Attribute-based Keyword Search with Fine-grainedOwner-enforced Search Authorization in the Cloud," IEEE

Transactions on Parallel and Distributed Systems, 2016, vol. 27, no. 4, pp. 1187 - 1198.
[29] S. Qiu, J. Liu, Y. Shi, R. Zhang. "Hidden policy ciphertext-policy attribute-based encryption with keyword search against keyword guessing attack." Science China Information Sciences 60.5 (2017): 052105.
[30] Y. Yang, S. Yang, F. Wang, J. Sun. "Post-Quantum Secure Public Key Broadcast Encryption with Keyword Search." Journal of Information Science and Engineering 33.2 (2017).
[31] Q Huang, H Li. "An Efficient Public-Key Searchable Encryption Scheme Secure against Inside Keyword Guessing Attacks." Information Sciences (2017).
[32] A. Ruhul. "Design of a Certificateless Designated Server Based Searchable Public Key Encryption Scheme." Mathematics and Computing: Third International Conference, ICMC 2017, Haldia, India, January 17-21, 2017, Proceedings. Vol. 655. Springer, 2017.
[33] Y. Yang, X. Zheng, V. Chang, S. Ye, C. Tang. "Lattice assumption based fuzzy information retrieval scheme support multi-user for secure multimedia cloud." Multimedia Tools and Applications: 1-15 (2017).
[34] A. Beimel, "Secure Schemes for Secret Sharing and Key Distribution," PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel, 1996.
[35] D. Boneh and X. Boyen, "Short signatures without random oracles," in Advances in Cryptology (Lecture Notes in Computer Science), vol. 3027, C. Cachin and J. L. Camenisch, Eds. Berlin, Germany: Springer-Verlag, 2004, pp. 56-73.
[36] J. Baek, R. Safavi-Naini, W. Susilo, "Public key encryption with keyword search revisited," in International conference on Computational Science and Its Applications. Springer Berlin Heidelberg, 2008: 1249-1259.
[37] L. Guo, W.C. Yau, "Efficient secure-channel free public key encryption with keyword search for EMRs in cloud storage," Journal of medical systems, 2015, 39(2): 1-11.
[38] B. Lynn. The Stanford Pairing Based Crypto Library. [Online]. Available: http://crypto.stanford.edu/pbc, accessed May 7, 2014.
[39] http://csrc.nist.gov/groups/ST/toolkit/keymanagement.html.
[40] W. Yau, S. Heng, B. Goi. "Off-line keyword guessing attacks on recent public key encryption with keyword search schemes." International Conference on Autonomic and Trusted Computing. Springer Berlin Heidelberg, 2008.

Yang Yang received the B.Sc. degree from Xidian University, Xi'an, China, in 2006 and Ph.D. degrees from Xidian University, China, in 2012. She is a research fellow (postdoctor) under supervisor Robert H. Deng in School of Information System, Singapore Management University. She is also an associate professor in the college of mathematics and computer science, Fuzhou University. Her research interests are in the area of information security and privacy protection.

Ximeng Liu received the B.Sc. degree from Xidian University, Xi'an, China, in 2010 and Ph.D. degrees from Xidian University, China, in 2015. He was the research assistant at School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore from 2013 to 2014. Now, he is a research fellow at School of Information System, Singapore Management University, Singapore. His research interests include cloud security and big data security.

Robert H. Deng is AXA Chair Professor of Cybersecurity in the School of Information Systems, Singapore Management University. His research interests include data security and privacy, network and system security. He has served/is serving on the editorial boards of many international journals in security, such as IEEE Transactions on Information Forensics and Security, IEEE Transactions on Dependable and Secure Computing, the International Journal of Information Security, and IEEE Security and Privacy Magazine. He is Fellow of IEEE.

Yingjiu Li is an Associate Professor at the School of Information Systems, Singapore Management University. His research interests include RFID Security and Privacy, Mobile and System Security, Data Application Security and Privacy. He has served on the editorial boards (and Committee Chair) of many information security international journals (and conferences).