**Singapore Management University**

[**Institutional Knowledge at Singapore Management University**](https://ink.library.smu.edu.sg/)

|  |  |  |
| --- | --- | --- |
| [Research Collection School Of Information](https://ink.library.smu.edu.sg/sis_research) | [School of Information Systems](https://ink.library.smu.edu.sg/sis) |  |
| [Systems](https://ink.library.smu.edu.sg/sis_research) |  |
|  |  |

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

Follow this and additional works at: [https://ink.library.smu.edu.sg/sis\_research](https://ink.library.smu.edu.sg/sis_research?utm_source=ink.library.smu.edu.sg%2Fsis_research%2F3774&utm_medium=PDF&utm_campaign=PDFCoverPages)

 Part of the [Health Information Technology Commons](http://network.bepress.com/hgg/discipline/1239?utm_source=ink.library.smu.edu.sg%2Fsis_research%2F3774&utm_medium=PDF&utm_campaign=PDFCoverPages), and the [Information Security Commons](http://network.bepress.com/hgg/discipline/1247?utm_source=ink.library.smu.edu.sg%2Fsis_research%2F3774&utm_medium=PDF&utm_campaign=PDFCoverPages)

**Citation**

YANG, Yang; LIU, Ximeng; DENG, Robert H.; and LI, Yingjiu. Lightweight sharable and traceable secure mobile health system. (2020). *IEEE Transactions on Dependable and Secure Computing*. 17, (1), 78-91. Research Collection School Of Information Systems.

**Available at:** https://ink.library.smu.edu.sg/sis\_research/3774

This Journal Article is brought to you for free and open access by the School of Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email [library@smu.edu.sg](mailto:library@smu.edu.sg).

|  |  |  |
| --- | --- | --- |
| This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI |  |  |
| **Published in IEEE Transactions on Dependable and Secure Computing, 2017 July, Volume PP, Issue 99, Pages 1-14** |  |  |
| 10.1109/TDSC.2017.2729556, IEEE Transactions on Dependable and Secure Computing | 1 |  |
| IEEE TRANSACTIONS**http://doi.org/10**ON**.**DEPENDABLE**1109/TDSC.2017**ANDSECURE**.2729556**COMPUTING |  |

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 viawearable 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.

F

* **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 im-proved data access. ”Improved patient care” means that mHealth could realize telemedicine since the patient’s con-ditions 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, stor-age 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.

Y. Yang and X. Liu are with College of Mathematics and Computer Sci-ence, Fuzhou University, Fuzhou, China, 350116; and School of Informa-tion 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.

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 solu-tion 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 en-crypted 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 re-vocation, which incur too much operational overhead for

1545-5971 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications\_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2729556, IEEE Transactions on Dependable and Secure Computing

|  |  |
| --- | --- |
| IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING | 2 |

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 over-riding design objective. We achieve this by elaborately out-sourcing 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 of-floaded to the public cloud and only a few exponen-tiation 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 ob-tain encrypted EHRs containing a certain keyword from the public cloud, a data user generates a key-word 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 key-word trapdoor from a data user, the cloud runs a test algorithm to retrieve encrypted EHRs containing the underlying keyword. Only three bilinear computa-tions 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 de-cryption algorithm, most of the ABE decryption op-erations 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 expensiveperiodic 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-manyencryption characteristic of ABE, decryption privi-leges 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 ex-isting 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. Ostro-vsky 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 decen-tralized 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 de-cryption 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.

1. 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 com-plexity 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 compu-tation 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

1545-5971 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications\_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2729556, IEEE Transactions on Dependable and Secure Computing

|  |  |
| --- | --- |
| IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING | 3 |

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] intro-duced 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 key-word 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

1. 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],
2. 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.

* **PRELIMINARIES**

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

**2.1 Access Policy**

**Definition 1** (Access Structure [34]). LetfP1; ; Pngbea set of parties. A collection A 2fP1; ;Png is monotone if 8B and C: if B 2 A and B C then C 2 A. An access structure (respectively, monotone access structure)

is a collection (resp. monotone collection) A of non-empty subsets of fP1; ; Png, i.e., A 2fP1; ;Pngnf g. The sets in A are called the authorized sets, and the sets not in A are called the unauthorized sets.

The role of parties is taken by attributes in ABE scheme. Thus, an access structure 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 over a set of parties P is called linear (over Zp) if

The shares for each party form a vector over Zp.

There exists a matrix M with l rows and n columns called the share-generating matrix for . For all i = 1; ; l, the ith row of M is labeled by a party (i) ( is a function from f1; ; lg to P). When we consider the column vector v = (s; r2; ; rn), where s 2 Zp is the secret to be shared and r2; ; rn 2 Zp are randomly chosen, then M v is the vector of l shares of the secret s according to . The share (M v)i belongs to party (i).

Every LSSS according to the definition achieves the linear reconstruction property [34]. Suppose that is an LSSS for the access structure A. Let S 2 A be any authorized set and I f1; ; lg be defined as I = fi : (i) 2 Sg. Then, there exists constants f!i 2 Zpgi2I such that, if f igi2I are valid shares of any secret s according to ,

P

then i2I !i i = s. Furthermore, it is shown in [34] that these constants f!igi2I 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; ) will be used to express an access policy associated to a ciphertext.

**2.2** **Bilinear Groups**

Let Gp be an algorithm that on input the security parameter , outputs the parameters of a prime order bilinear map as (p; g; G; GT ; e), where G and GT are multiplicative cyclic groups of prime order p and g is a random generator of G. The mapping e : G G ! GT is a bilinear map. The bilinear map e has three properties: (1) bilinearity: 8u; v 2 G and a; b 2 Zp, we have e(ua; vb) = e(uv)ab. (2) non-degeneracy: e(g; g) 6= 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])**. LetGbe abilinear group of prime order p and g be a generator of G, the q-Strong Diffie-Hellman (q-SDH) problem in G is defined as follows: given a (q +1)-tuple (g; gx; gx2 ; ; gxq ) as inputs, output a pair (c; g1=(c+x)) 2 Zp G. An al-gorithm A has advantage in solving l-SDH in G if P r[A(g; gx; gx2 ; ; gxq ) = (c; g1=(c+x))] , where the probability is over the random choice of x in Zp and the

random bits consumed by A.

**Definition.** We say that the(q; t;)-SDH assumptionholds in G if no t-time algorithm has advantage at least in solving the q-SDH problem in G.

**Assumption 2 (decisional bilinear Diffie-Hellman as-sumption)**. LetGbe a bilinear group of prime orderpandgbe a generator of G. Let a; b; s 2 Zp be chosen at random. If an adversary A is given ~y = (g; ga; gb; gs), it is hard for the attacker A to distinguish e(g; g)abs 2 GT from an element R that is randomly chosen from GT .

The adversary A has advantage in solving the above

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| assumption if | | P r[A(~y; T = R) = 0] |  |  |
|  | P r[A(~y; T = e(g; g)abs) = 0] | : |  |
|  |  |  |  |  |
|  |  |  |  |  |

1545-5971 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications\_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2729556, IEEE Transactions on Dependable and Secure Computing

|  |  |
| --- | --- |
| IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING | 4 |

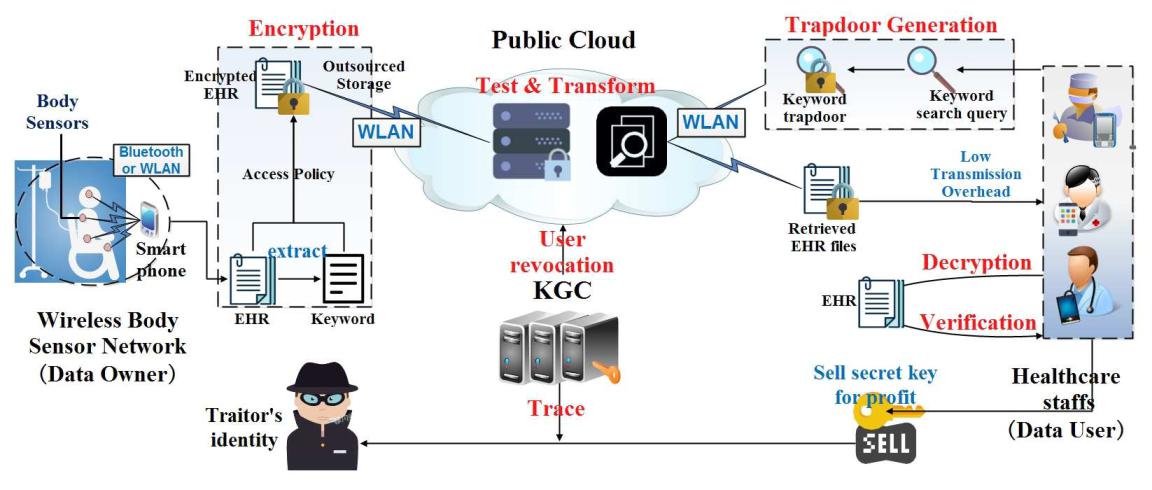


Fig. 1: LiST System Architecture

* **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 wirelesssensors 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 thedata 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 unlimitedstorage and computing power to undertake the EHR re-mote storage task and respond on data retrieval requests.

Lightweight test algorithm is designed in our proposed sys-tem 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 entiresystem 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 Supplemen-tal 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 u-nauthorized 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 cipher-text of an EHR, a secure searchable data sharing system should satisfy the following requirement: indistinguishable

1545-5971 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications\_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2729556, IEEE Transactions on Dependable and Secure Computing

|  |  |
| --- | --- |
| IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING | 5 |

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],

1. 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.

* **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 fig-ure 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 mecha-nism 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; ) | attribute set/access structure |
| PK=SK | public key/ secret key pair of user |
| KW=TKW | keyword/keyword trapdoor |
| CT =Cm | index/message ciphertext |
| CTout | outsourced ciphertext |
| SEnc=SDec | symmetric encryption/decryption pair |

* key space of symmetric encryption

4.2.1 System Setup

The KGC takes the security parameter 1 as input. It outputs the public parameter P P of the whole system and keeps secret the generated master secret key M SK.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Setup(1 ) | | ! (PP; MSK): | | | | | | Run | Gp(1 ) | | ! | |  |
| (p; G; GT ; e). Let g | | | | | 2 G be the generator of group | | | | | | | |  |
| G | . Select random ; ; | | | | | 2R |  | and k |  | ; k | K | . |  |
|  |  |  | ; Y |  | Zp | 1 | | 2 2R | . |  |
| Compute f = g | | |  | = e(g; g) | | ; Y0 = e(g; f); h = g | | | | |  |
| The public parameter is P P | | | | | | | = (g; h; f; Y; Y0) and | | | | | |  |
| the master secret key is M SK | | | | | | | | = ( ; ; ; k1; k2). | | | | |  |
| The KGC | | also | | | defines | | two | hash |  | functions: | | |  |

* : f0; 1g ! Zp and H1 : f0; 1g ! K. We will omit P P 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 SKid;S.

KeyGen(M SK; id; S) ! (P Kid;S; SKid;S): The key generation algorithm takes the master secret key M SK, the user’s identity id and an attribute

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| set S = f 1; 2 | | | ; ; kg | |  |  | Zp as input. Choose | | |  |
| a; r; ; %; s0; s00; u0 | | | ; u00; u000 | | 2R |  | Zp. Compute | | = |  |
| SEnck1 (id); = | | | | SEnck2 ( jj ). The public key | | | | | |  |
| P Kid;S and secret key SKid;S are constructed as: | | | | | | | | | |  |
|  | ar | |  |  |  |  |  | 1 |  |  |
| D1 = g |  | ; D2 = ; D3;i = gar( i+ ) | | | | | | ;D4 = %; | |  |
| + |  |
| 1 = (D1%)u0 | | | | ; 2 = Y0u000 | | ; | 3;i = (D3%;i)u00 | | ; |  |
|  |  |  |  | 4 = gs0; 5 = fs00; | | | |  |  |  |
| P Kid;S = ( | | | | 1; 2; f 3;igi2[k]; | | | | 4; 5); |  |  |

SKid;S = (D1; D2; fD3;igi2[k]; D4; s0; s00; u0; u00; u000):

Note that 1=( + ) is computed modulo p. If gcd( + ; p) 6= 1, theKeyGen algorithm chooses another 2R Zp and repeat the computation

= SEnck2 ( jj ).

1545-5971 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications\_standards/publications/rights/index.html for more information.

; C4; Cm) are out-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2729556, IEEE Transactions on Dependable and Secure Computing

|  |  |
| --- | --- |
| IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING | 6 |

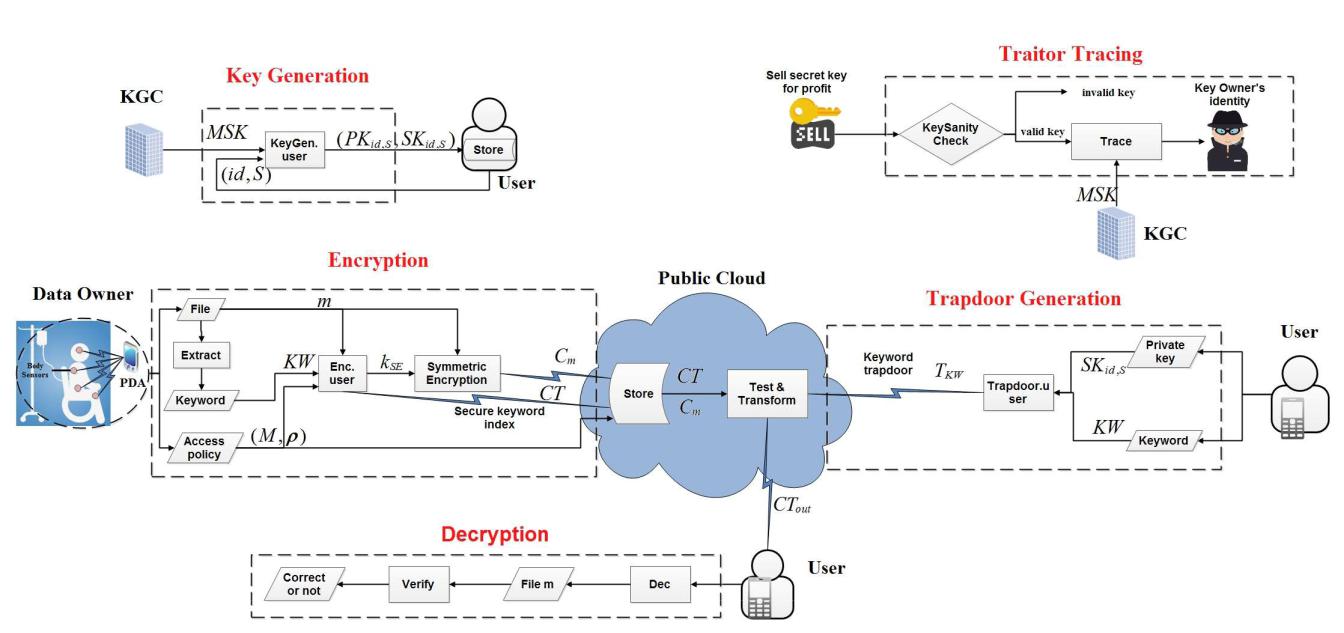


Fig. 2: High level interactions in the concrete construction

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; ); KW ) ! CT: Let M be an l n matrix and be the function that associates rows of M to attributes. Select random 2R GT . Set kSE = H1( ). The data owner selects a positive integer $ and concatenates $-bit 0 string after the message m, which is used for outsourced decryption verification. Then, compute Cm = SEnckSE (mjj0$), where jj denotes concatenation of a string. The data owner picks a random s 2R Zp and then choose a random vector ~v = (s; y2; ; yn)> 2 Znp which is used to share s. For i 2 [l], compute si = Mi ~v, where Mi is the vector corresponding to the ith row of M. Compute the ciphertext CT as:

C0 = Y s; C1 = gs; C2 = hs;

C3;i = (i)si=[s0H(KW )]; C30;i = si=[s00H(KW )];

C4 = Y0H(KW )Y s=H(KW ):

Then, the access policy (M; ) and the ciphertext

CT = (C0; C1; C2; fC3;i; C30;igi2[l] sourced to cloud.

**Remark**: In the LiST system, the encryption algorithm issuitable 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 cipher-

text is denoted as Cm = SEnckSE (mjj0$), where m repre-sents 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 TKW using the following T rapdoor algorithm. The attribute set S is also implicitly included in the generated trapdoor TKW , which is transmitted to public cloud server via wireless channel.

T rapdoor(SKid;S; KW ) ! TKW : The data user chooses u; u0 2 Zp and computes the keyword trap-door TKW = (T0; T1; T2; T3; T30; T4; T5) as:

T0 = u (u0)1 ; T1 = u0=[u0H(KW )]; T2 = D2;

T3 = u0 (u00)1 ; T30 = uH(KW ) (u00)1 ;

T4 = u0D4; T5 = u0D4 H(KW ) (u000)1 :

4.2.5 Data Retrieval

Receiving the data retrieval request from data user, the public cloud server responds on the request and search-es on the stored encrypted EHRs to look for match-ing files. The cloud server provider leverages on the test and transform phases to complete the process:

T est&T ransf orm(CT; TKW ; P Kid;S) ! CTout=?:

In the T est algorithm, the public cloud server searches for the matching encrypted EHRs if the ciphertext satisfies the following two requirements: the attribute set S (im-plicitly included in keyword trapdoor) satisfies the access

1545-5971 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications\_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2729556, IEEE Transactions on Dependable and Secure Computing

|  |  |
| --- | --- |
| IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING | 7 |

structure defined in the encrypted EHR; the keyword con-tained in the keyword trapdoor is in accordance with that in ciphertext.

T est(CT; TKW ; P Kid;S) ! 1=0: Suppose CT as-sociate with keyword KW 0 and TKW with KW . The algorithm verifies whether S associated with

TKW satisfies (M; ) associated with CT . If not, it outputs 0. Otherwise, let I [l] be defined as

* = fi : (i) 2 Sg. There exists a set of constants

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| fwi 2 Zpgi2I so that | i2I wiMi = (1; 0; ; 0). The | | | | | |  |
| algorithm computes | P |  |  |  |  |  |  |
| Y | C3;i wi | ] e[ 5; | Y | C30 | ;i wi |  |  |
|  |  |  |  |  |
| = e[ 4; | 3;i | i2I | 3;i |  | ]; |  |
| i2I |  |  |  |  |  |  |
| = e( 1; C1T2 C2); | | 2= T1; 2= T3: | | | | |  |

Then, the algorithm verifies whether the following equation holds

T25 ( 2= 2) = C4T4:

If the equation does not hold, it outputs 0 indicating that KW 0 6=KW . Otherwise, it outputs 1.

In the T ransf orm algorithm, the public cloud server transforms the matched ciphertext CT into CTout such that the data user could use lightweight decryption algorithm to recover the plaintext.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| T ransf orm(CT; TKW ; P Kid;S) ! | | | | CTout=?: If |  |
| the output of ”T est” algorithm is 0, it outputs ?. | | | | |  |
| Otherwise, it computes CTout = (C0; | | | | 1; 1; Cm), in |  |
| which 1 = | T | T | 0 |  |  |
|  | 0; 1= | 3 . |  |  |

4.2.6 Data Decryption and Verification

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

Dec(CTout; SKid;S) ! m=?: Compute

C0

= :

( 1= 1)1=(uD4)

Then, the user computes kSE = H1( ) and m0 = SDeckSE (Cm). The user checks whether a redundancy 0$ is appended after the recovered message. If so (m0 = mjj0$), the message m can be obtained by truncating $-bit 0 string. Otherwise, the cloud server is dishonest to return an incorrect transformed ciphertext and the algorithm outputs ?.

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(SKid;S) ! 1=0: Suppose S = f 1; 2; ; kg. The key sanity check of secret

key SKid;S consists of two steps. Firstly, the

KGC checks whether SKid;S is in the form of (D1; D2; fD3;igi2[k]; D4; s0; s00; u0; u00; u000) and D2; D4; s0; s00; u0; u00; u000 2 Zp; D1; D3;i 2 G. Then, the KGC verifies whether the equation holds

e(D1; h gD2 )k e(g; Y D3i;i) e(f; Y D3;i) = Y k:

i2[k] i2[k]

If SKid;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 com-putations on the component D1 of the secret key SKid;S. The T race algorithm is implemented by KGC using the master secret key M SK.

T race(M SK; SKid;S) ! id=?: If the output of KeySanityCheck algorithm is 0, it means that

SKid;S is not a well-formed secret key and not worth to be traced. The T race algorithm outputs

?. Otherwise, SKid;S is a well-formed secret key. The algorithm extracts ( jj ) = SDeck2 (D1). The malicious user’s identity id can be recovered by computing id = SDeck1 ( ).

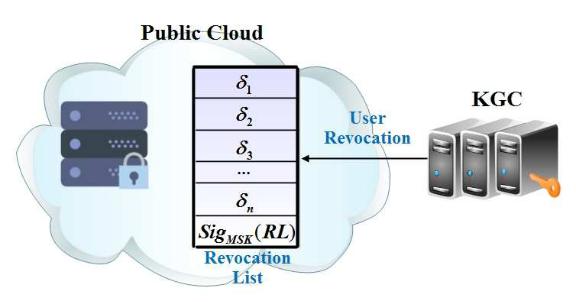


Fig. 3: User Revocation

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 D2 = of secret key contains the identity information of user. Moreover, it must be submitted to public cloud server as a component T2 = D2 of keyword trapdoor to issue a data retrieval re-quest. As shown in Fig. 3, the KGC could simply put D2 = into the revocation list to realize the user revocation. The revocation list should be stored (together with a signature

1545-5971 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications\_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2729556, IEEE Transactions on Dependable and Secure Computing

|  |  |
| --- | --- |
| IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING | 8 |

signed by KGC) in the public cloud server. When the public cloud receives a keyword trapdoor TKW , it should firstly check whether T2 = D2 = 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 D2 = is generated from user’s identity. It is also implicitly embedded into another element

ar

D1 = g + of secret key. When a malicious user wants to sell his secret key without being traced, he may intend to re-randomize the element D2 to remove his identifier information from the secret key. Although he can easily generate a new D20 = 0, it is impossible for him to generate

ar

a valid D10 = g + 0 without the master secret keys and . The construction of element D1 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 orig-inal 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 in-dices 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 com-bined 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 applica-tion. 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 del-icate algorithm design, which can not be realized by the simple combination.

* **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 D2 can be utilized to identify the traitor.

**Theorem 1**. If the decisional bilinear Diffie-Hellman (DB-DH) 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 the-orem 1, the challenger C and polynomial time adversary A interact with each other. If A could break the IND-CKCCA security of the LiST system, C then utilizes the interactive game with A to solve the DBDH problem. The game is briefly described as below.

(1) In the setup phase, C is given an instance of the DBDH assumption (~y; T ), where ~y = (g; ga; gb; gs), and T = e(g; g)abs or T is a random element in GT . Then, C utilizes the received elements to construct the public parameter of the system.

1. In the query phase 1, A adaptively issues the secret key queries and trapdoor queries. C responds on these queries and returns the corresponding secret keys or trap-doors to A.
2. In the challenge phase, A sends the challenge access policy (M ; ), two keywords (KW0 ; KW1 ) and two mes-sages (m0; m1) to C. The restriction is that the secret key of attribute set S that satisfies (M ; ) has not been queried in query phase 1. Then, C flips random coins 1; 2 2 f0; 1g,

and encrypts the message m and the keyword KW . A

1 2

key point is that the element T should be embedded in the generated challenge ciphertext CT , which is sent to A.

1. Receiving the challenge ciphertext CT , the adver-sary A continues to issue queries as in phase 1. The restric-

tion is that KW 2= fKW0 ; KW1 g and S does not satisfy

(M; ).

(5) In the guess phase, A outputs a guess 01; 02 2 f0; 1g. If 01 = 1; 02 = 2, then C outputs 1 meaning T = e(g; g)abs. Otherwise, it outputs 0 meaning T is a random element in GT .

If C wins the game with non-negligible probability, then A could utilize the guess of C to solve the DBDH prob-lem 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(t;)traceable | |
| under the (q; t~0; 0)-SDH assumption with 0 = ; t~0 | t~ + |

te [ O(jSj)qsk ]; where qsk denotes the total numbers of user secret key queries, te denotes the running time of an exponentiation, jSj 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 C and polynomial time adversary A interact with each other. If A could break the traceability of the LiST system, C then utilizes A to solve the q-SDH problem. The game is briefly described as below.

1. In the setup phase, C is given an instance of the q-SDH assumption (^g; g^ ; g^ 2 ; ; g^ q ). Then, C utilizes the received elements to construct the public parameter of the system.
2. In the query phase 1, A adaptively issues the secret key queries. C responds on these queries and returns the corresponding secret keys to A.

1545-5971 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications\_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2729556, IEEE Transactions on Dependable and Secure Computing

|  |  |
| --- | --- |
| IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING | 9 |

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

If C wins the game with non-negligible probability, then A could utilize the the forged secret key of 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 animportant type of attack in multi-user system. The autho-rized 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.

* **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 | Search | Out. | Ver. | W.B. | User |  |
|  | Control |  | Dec. | Dec. | Trace | Revoke |  |
| [6] | p |  |  |  |  |  |  |
| p |  |
| [7] |  |  |  |  |  |  |
| [9] | p |  | p |  |  |  |  |
| [10] | p |  | p | p |  |  |  |
| p | p | p |  |
| [11] |  |  |  |  |
| p | p | p |  |
| [12] |  |  |  |  |
| [13] | p |  |  |  | p |  |  |
| p | p |  |
| [14] |  |  |  |  |  |
| [27] | p | p |  |  |  |  |  |
| [28] | p | p |  |  |  | p |  |
| **LiST** | p | p | p | p | p | p |  |

**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 jP P j; jSKj; jCT j; jTKW j be the sizes of the public pa-rameter, secret key of user, the ciphertext and the keyword trapdoor, respectively. Let l be the number of rows in matrix

* of access structure, jSj be the size of attribute set S and jUj be the size of the universe attribute set U. jGj, jGT j and jZpj represent the bit length of an element in group G, GT and Zp, respectively. Denote te1, te2 and tp as the times consumed for a modular exponentiation on group G, a modular exponentiation on group GT and a bilinear pairing operation, respectively.

The storage overhead comparison is shown in Table 3. The comparison shows that LiST has smaller public param-eter 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 jUj, 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 LiSTconsists of jSj + 1 elements in group G and seven elements in Zp. Generally speaking, jZpj is always smaller than jGj, 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 LiSThas 2 elements in group G, 2 elements in group GT and 2l elements in Zp. Since the jZpj is typically at least one sixth of jGj, the jCT j 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 Zp, 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

1545-5971 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications\_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2729556, IEEE Transactions on Dependable and Secure Computing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING | | | |  | 10 | |
|  |  | TABLE 3: Storage Overhead Comparison | | |  |  |
|  |  |  |  |  |  |  |
|  | **Scheme** | jP P j | jSKj | jCT j | jTKW j | |
|  | [6] | (2jUj + 3)jGj | (5jSj)jGj | (2l + 1)jGj + jGT j | ? |  |
| [7] | | (2jUj + 4)jGj + jGT j | (jSj + 6)jGj | (5l)jGj + jGT j | ? |  |
| [9] | | 2jGj + jGT j | (jSj + 2)jGj | (2l + 1)jGj + jGT j | ? |  |
| [10] | | (jUj + 5)jGj + jGT j | (jSj + 2)jGj | (4l + 3)jGj + 2jGT j | ? |  |
| [11] | | (jUj + 2)jGj + jGT j | (jSj + 2)jGj | (2l + 1)jGj + jGT j | ? |  |
| [12] | | (jUj + 4)jGj + jGT j | (jSj + 2)jGj | (2l + 1)Gj + jGT j | ? |  |
| [13] | | (jUj + 3)jGj + jGT j | (jSj + 3)jGj + jZpj | (2l + 2)jGj + jGT j | ? |  |
| [14] | | 6jGj + jGT j | (2jSj + 3)jGj + jZpj | (3l + 2)jGj + jGT j | ? |  |
| [27] | | (2jUj + 10)jGj + 3jGT j | (3jSj)jGj | (l + 3)jGj + 2jGT j | (3jSj)jGj | |
| [28] | | (3jUj + 1)jGj + jGT j | (2jSj + 1)jGj + jZpj | (l + 2)jGj | (2jSj + 2)jGj | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  | **LiST** | | | |  | 3jGj + 2jGT j | | | | (jSj + 1)jGj + 7jZpj | 2jGj + 2jGT j + 2ljZpj | | | | | | 7jZpj | |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | TABLE 4: Computation Overhead Comparison | | | | | | | | |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |
| **Scheme** |  | KeyGen | | | | | | | | |  |  |  |  | Enc | |  | Dec | T rapdoor | | | |  | T est | KeySanityCheck | |  |
|  |  |  |  |  |  |  |  | &T race |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| [6] | (6 | | | | j | S | | | )te |  |  |  |  | tp + te2 | | |  | (3jSj)tp |  |  |  | ? |  | ? | ? | |  |
|  |  |  |  |  |  |  | j |  | 1 |  |  | +(2l + 1)te1 | | | | | +jSjte1 + 2jSjte2 |  |  |  |  |  |
| [7] | ( |  | S | |  | + | | | 9)te | |  |  | tp + (6l)te1 | | | | | (6jSj)tp |  |  |  | ? |  | ? | ? | |  |
|  | j | | j | | |  |  |  |  |  | 1 | |  |  | +l te2 | |  | +(2jSj)te2 |  |  |  |  |  |
| [9] | (jSj + | | | | | | | | 2)te1 | | | | tp + (3l + 1)te1 + te2 | | | | | te1 |  |  |  | ? |  | ? | ? | |  |
| [10] | (jSj + | | | | | | | | 3)te1 | | | | 2tp + (6l + 4)te1 + 2te2 | | | | | te1 |  |  |  | ? |  | ? | ? | |  |
| [11] | (jSj + | | | | | | | | 3)te1 | | | | tp + (3l + 1)te1 + te2 | | | | | te1 |  |  |  | ? |  | ? | ? | |  |
| [12] | (jSj + | | | | | | | | 3)te1 | | | | tp + (3l + 3)te1 + te2 | | | | | te1 |  |  |  | ? |  | ? | ? | |  |
| [13] | ( |  | S | |  | + | | | 4)te | |  |  | tp + te2 + | | | |  | (2jSj + 1)tp + jSjte2 |  |  |  | ? |  | ? | (2jSj + 5)tp + te2 | |  |
|  | j | | j | | |  |  |  |  |  | 1 | | (3jlj + 2)te1 | | | | | +(jSj + 1)te1 |  |  |  |  | +(jSj + 3)te1 | |  |
| [14] | (4 | j | | S | |  | + 4)te | | | | |  | tp + te2 + | | | |  | (3jSj + 1)tp + jSjte2 |  |  |  | ? |  | ? | (4jSj + 5)tp + (jSj + 1)te2 | |  |
|  |  |  | j | |  |  |  |  |  | 1 | (5jlj + 2)te1 | | | | | +(jSj + 1)te1 |  |  |  |  | +(jSj + 4)te1 | |  |
| [27] |  |  |  | 4jSjte1 | | | | | |  |  |  | 2tp + 2te2 + | | | | | 4tp + te2 + |  |  | 8jSjte1 | |  | 2tp+ | ? | |  |
|  |  |  |  |  |  | (l + 5)te1 | | | |  | (3l + 4)te1 |  |  |  | (2l)te1 |  |
| [28] | (2 | j | | S | |  | + 2)te | | | | |  | (2 | j | S | + 2)te |  | ? | (2 | j | S | + 2)te |  | (jSj + 1)tp | ? | |  |
|  |  |  | j | |  |  |  |  |  | 1 |  | j |  | 1 |  | j |  | 1 | +te2 |  |
| **LiST** | (jSj + | | | | | | | | 1)te1 | | | | 2te1 + 3te2 | | | | | te1 | 0tp + 0te1 | | | |  | 3tp + te2 + | 3tp + 2te2 + | |  |
|  |  | +0te2 | |  | (2l + 4)te1 | (jSj + 1)te1 | |  |

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 over-head in each algorithms compared with other schemes.

KeyGen: In the KeyGen algorithm, the KGC could utilize jSj + 1 exponentiation operations on group 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 6jSj, 4jSj + 4 and 4jSj exponentiation calculations on group G in KeyGen computation, respectively.

Enc: In LiST, the EHR encryption is done by user’s energy limited device. In real-time monitoring med-ical care system, the health information will be con-tinuously 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 G and three expo-nentiations on group GT 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 mechanis-m, the schemes in [9], [10], [11], [12] and LiST can recover the EHR with only one exponentiation com-

putation on group G. However, they [9], [10], [11],

* 1. 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.

T rapdoor: In Table 4, it is shown that the com-

putation overhead of T rapdoor algorithm in LiST is 0tp + 0te1 + 0te3 . It means that no pairing or exponentiation computations is required in LiST. In fact, only a few lightweight multiplication, division and inversion computations on Zp 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.

T est: In the T est 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 pro-cesses stronger computation power and continuous energy supply.

KeySanityCheck&T race: The schemes in [13] and

1545-5971 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications\_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2729556, IEEE Transactions on Dependable and Secure Computing

|  |  |
| --- | --- |
| IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING | 11 |

1. take the traitor tracing problem into considera-tion. However, a large amount of pairing computa-tions 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 con-ducting experiment is running Windows 7 64-bit operation system with the following configurations: Intel CoreT M i3-2120 CPU @ 3.30 GHz, 4.00 GB RAM.

We use Java Pairing Based Cryptography (JPBC) Li-brary [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 con-ducting experiment, which is equivalent to 80-bit security level [39]. It has the expression form E : y2 = x3 + x over Fq finite field. Both group G and group GT have order p and are subgroups of E(Fq). The parameters q and p are equivalent to 512 bits and 160 bits numbers in binary system, respectively. Then, we have jZpj = 160 bits, jGj = 1024 bits and jGT j = 1024 bits.

TABLE 5: Storage Overhead (bits) (jSj = 100)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Scheme | [13] | [14] | [27] | [28] | **LiST** |
| jP P j | 106,496 | 8,192 | 218,112 | 309,248 | **7,168** |
| jSKj | 105,632 | 208,032 | 307,200 | 207,008 | **104,544** |
| jCT j | 207,008 | 309,408 | 105,792 | 104,448 | **36,096** |
| jTKW j | ? | ? | 307,200 | 206,848 | **1,120** |

6.2.2 Experiment Results

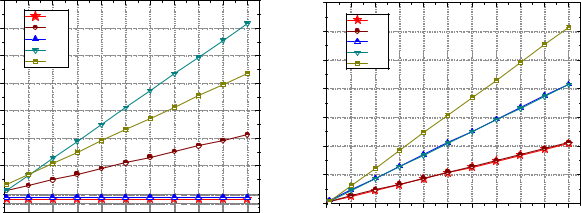
1. **Storage and Transmission Efficiency.**

To evaluate the storage and transmission overhead, we

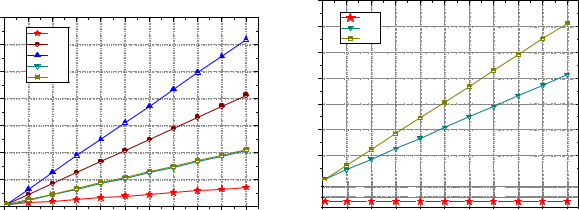
compare LiST with the schemes [13], [14], [27], [28] in terms of the public parameter size, secret key size and trapdoor size (bits) in Figure 4 and Table 5. In order to describe the performance, a non-uniform axis is used in Fig. 4(a) to make the different values much clearer.

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 350000 |  |  |  |  |  |  |  |  |  |  |  | 350000 |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | LiST |  |  |  |  |  |  |  |  |  |  |  |  | LiST |  |  |  |  |  |  |  |  |  |
|  | 300000 |  | [13] |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  | 300000 |  |  | [13] |  |  |  |  |  |  |  |  |  |
|  |  |  | [14] |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | [14] |  |  |  |  |  |  |  |  |  |
|  | 250000 |  | [28] |  |  |  |  |  |  |  |  |  | 250000 |  |  | [28] |  |  |  |  |  |  |  |  |  |
| Size(bits) |  |  | [27] |  |  |  |  |  |  |  |  | Size(bits) |  |  | [27] |  |  |  |  |  |  |  |  |  |
| 200000 |  |  |  |  |  |  |  |  |  |  | 200000 |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 150000 |  |  |  |  |  |  |  |  |  |  |  | 150000 |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 100000 |  |  |  |  |  |  |  |  |  |  |  | 100000 |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 50000 |  |  |  |  |  |  |  |  |  |  |  | 50000 |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 10000 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 5000 |  |  |  |  |  |  |  |  |  |  |  | 0 |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 0 0 | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |  |  | 0 | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |  |
|  |  |  |  | Number of attributes | | | | |  |  |  |  |  |  |  |  | Number of attributes | | | | | |  |  |  |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | (a) Public Parameter Size | |  |  | (b) Secret Key Size |  |
|  |  |  |  | 350000 |  |  |
|  | 350000 |  |  | 300000 | LiST |  |
|  |  | LiST |  | [28] |  |
|  |  |  |  | [27] |  |
|  | 300000 | [13] |  |  |  |
|  |  | 250000 |  |  |
|  |  | [14] |  |  |  |
|  | 250000 | [28] | Size(bits) |  |  |  |
| Size(bits) | [27] | 200000 |  |  |
|  |  |  |
|  | 200000 |  |  | 150000 |  |  |
|  |  |  |  |  |  |
|  | 150000 |  |  | 100000 |  |  |
|  |  |  |  |  |  |



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 100000 |  |  |  |  |  |  |  |  |  |  | 50000 |  |  |  |  |  |  |  |  |  |  |  |
| 50000 |  |  |  |  |  |  |  |  |  |  | 4000 |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0 0 |  |  |  |  |  |  |  |  |  |  | 2 000 |  |  |  |  |  |  |  |  |  |  |  |
| 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 | 00 | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |  |

Number of attributes Number of attributes

(c) Ciphertext Size (d) Trapdoor Size

Fig. 4: Storage and transmission overhead

total number of attributes in the system grows to 100, which is 43 times of ours parameter size. Since the P P 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].

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 jSK j and the lines overlaps with each other. When jSj = 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 trans-mit 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 num-ber of the attribute set. It is much smaller than that in [28] (206,848 bits) and [27] (307,200 bits) when jSj = 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 G | 9.175 | 90.118 |
| Exponentiation on group GT | 2.784 | 33.4 |
| Multiplication on Zp | 0.001 | 0.026 |
| Division on Zp | 0.005 | 0.093 |
| Inversion on Zp | 0.004 | 0.057 |

1545-5971 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications\_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2729556, IEEE Transactions on Dependable and Secure Computing

|  |  |
| --- | --- |
| IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING | 12 |

TABLE 7: Computation Time (ms) (jSj = 100)

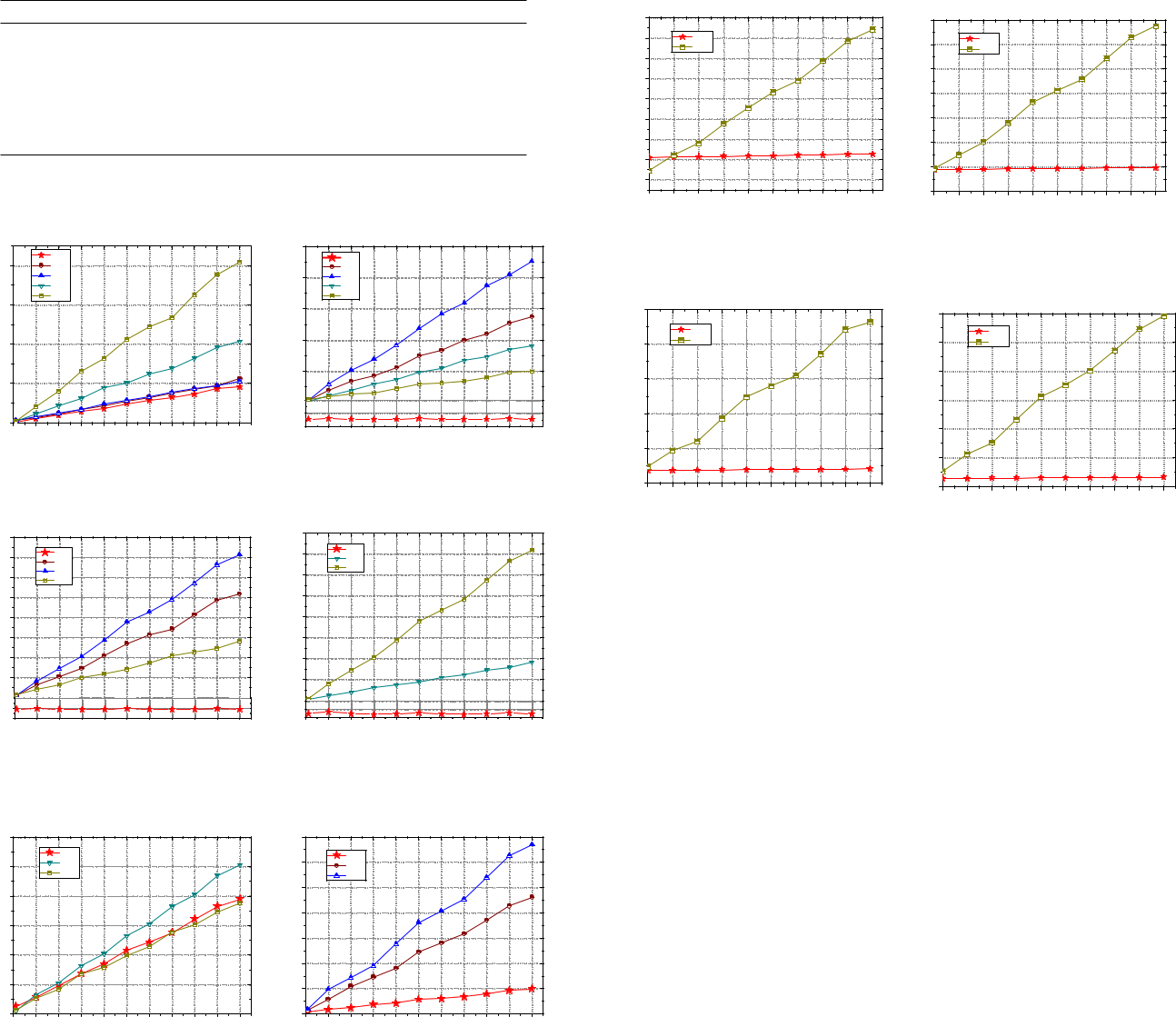
|  |
| --- |
| Time (ms) |

|  |
| --- |
| Time (ms) |

|  |
| --- |
| Time (ms) |

(2)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Scheme | | |  |  | [13] | |  | [14] | | [27] | | |  |  | [28] | |  | **LiST** | |  |  |  |  | Data Retrieval and Recovery Time | | | | | | | | |  |  |  | Data Retrieval and Recovery Time | | | | | | | | | |  |  |
|  |  |  |  |  |  |  |  |  |  | 650 | |  |  |  |  | |S|=10 |  |  |  |  | 1100 | |  |  |  |  |  |  | |S|=20 | |  |  |  |  |  |
|  | KeyGen | | | |  | 1,118 | | |  | 1,055 | | 4,093 | | |  | 2,072 | | |  | **906.67** | | |  | 600 | |  | LiST | |  |  |  |  |  |  | 1000 | |  |  | LiST | |  |  |  |  |  |  |  |  |  |
|  |  | Enc | |  |  | 27,444 | | |  | 45,377 | | 9,919 | | |  | 18,203 | | |  | **280.43** | | |  |  |  |  | [27] |  |  |  |  |  |  |  |  |  | [27] | |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | 550 | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 900 |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | Dec | |  |  | 51,657 | | |  | 71,167 | | 28,209 | | | |  | ? |  |  | **90.11** | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  | 450 | |  |  |  |  |  |  |  |  |  |  | 800 |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | ? |  |  |  | ? |  |  |  |  |  |  |  |  |  |  |  | Time (s) | 500 | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | T rapdoor | | | |  |  |  |  |  | 72,094 | | | | 18,203 | | |  | **0.501** | |  | 400 | |  |  |  |  |  |  |  |  |  | Time (s) | 700 |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | T est | |  |  |  | ? |  |  |  | ? | 1,885 | | |  | 2,530 | | |  | **1,949** | |  | 350 | |  |  |  |  |  |  |  |  |  | 600 |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | T race | | |  | 4,624 | | |  | 6,712 | |  |  | ? |  |  | ? |  |  | **985.91** | | |  | 300 | |  |  |  |  |  |  |  |  |  |  | 500 |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 250 | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |  | 400 10 | | 20 | 30 |  | 40 |  | 50 | 60 | 70 | 80 | 90 | 100 |  |
|  |  |  | Key Generation Time | | | | | | |  |  |  |  |  |  |  | Encryption Time | | | |  |  |  |  |  |  |  | Number of returned files | | | | |  |  |  |  |  |  |  | Number of returned files | | | | | |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | (a) jSj = 10 | | | | |  |  |  |  |  |  |  | (b) jSj = 20 | | | | | | |  |  |  |  |
|  |  |  |  |  | (PKG: PC) | | |  |  |  |  | 50000 |  |  |  | (User: Mobile Device) | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | LiST |  |  |  |  |  |  |  |  |  |  | LiST |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4000 |  |  | [13] |  |  |  |  |  |  |  |  |  |  |  | [13] |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | [14] |  |  |  |  |  |  |  |  | 40000 |  |  | [14] |  |  |  |  |  |  |  |  |  |  | Data Retrieval and Recovery Time | | | | | | | | |  |  |  |  | Data Retrieval and Recovery Time | | | | | | | | | |  |
|  |  |  | [28] |  |  |  |  |  |  |  |  |  |  |  | [28] |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3000 |  |  | [27] |  |  |  |  |  |  |  | (ms) |  |  |  | [27] |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |S|=30 |  |  |  |  |  |  |  |  |  |  |  |  |  | |S|=40 | |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  | 30000 |  |  |  |  |  |  |  |  |  |  |  | 1600 |  |  |  |  |  |  |  |  |  |  | 2000 |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2000 |  |  |  |  |  |  |  |  |  |  | Time |  |  |  |  |  |  |  |  |  |  |  |  | 1400 |  |  | LiST |  |  |  |  |  |  |  |  | 1800 |  |  |  | LiST | |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  | 20000 |  |  |  |  |  |  |  |  |  |  |  |  |  | [27] |  |  |  |  |  |  |  |  |  |  |  | [27] | |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  | 10000 |  |  |  |  |  |  |  |  |  |  |  | 1200 |  |  |  |  |  |  |  |  |  |  |  | 1600 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1000 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | (s) |  |  |  |  |  |  |  |  |  |  | (s) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  | 1000 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1400 |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | Time |  |  |  |  |  |  |  |  |  |  |  | Time |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0 |  |  |  |  |  |  |  |  |  |  |  | 500 |  |  |  |  |  |  |  |  |  |  | 1000 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0 | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 | 0 | 0 | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |  |  |  |  |  |  |  |  |  |  |  |  | 1200 |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | Number of attributes | | | | |  |  |  |  |  |  |  | Number of attributes | | | | | |  |  | 800 |  |  |  |  |  |  |  |  |  |  |  | 1000 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| (a) Key Generation Time | | | | | | | | | | |  |  |  | (b) Encryption Time | | | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | 600 |  |  |  |  |  |  |  |  |  |  |  | 800 |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 10 |  | 20 | 30 |  | 40 | 50 | 60 | 70 | 80 | 90 | 100 |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | Number of returned files | | | | |  |  |  |  |  |  |  |
|  |  |  |  | Decryption Time | | | | |  |  |  |  |  |  | Trapdoor Generation Time | | | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  | Number of returned files | | | | | |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  | (User: Mobile Device) | | | | |  |  |  |  |  |  | (c) jSj = 30 | | | | |  |  |  |  |  |  |  | (d) jSj = 40 | | | | | | |  |  |  |  |
| 80000 |  |  | LiST | ( User: Mobile Device) | | | | |  |  |  | 80000 |  |  | LiST |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 70000 |  |  |  |  |  |  |  |  |  |  | 70000 |  |  | [28] |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | [13] |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | Fig. 6: Data Retrieval and Recovery Time Comparison | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | [27] |  |  |  |  |  |  |  |  |  |
| 60000 |  |  | [14] |  |  |  |  |  |  |  |  | 60000 |  |  |  |  |  |  |  |  |  |  |  |
|  |  | [27] |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 50000 |  |  |  |  |  |  |  |  |  |  | (ms) | 50000 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 40000 |  |  |  |  |  |  |  |  |  |  | 40000 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  | Time |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 30000 |  |  |  |  |  |  |  |  |  |  | 30000 |  |  |  |  |  |  |  |  |  |  |  |  |  | about 280.43 ms and not varies with the number | | | | | | | | | | | | | | | | | | | | | | |  |
| 20000 |  |  |  |  |  |  |  |  |  |  |  | 20000 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 10000 |  |  |  |  |  |  |  |  |  |  |  | 10000 |  |  |  |  |  |  |  |  |  |  |  |  |  | of | attributes. However, | | | | | | | | | the | | | EHR | | | | encryption | | | | | time |  |
| 200 | |  |  |  |  |  |  |  |  |  |  | 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 | 0 | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |  |  | consumes 27,444 ms, 45,377 ms, 9,919 ms and 18,203 | | | | | | | | | | | | | | | | | | | | | | |  |
|  | 00 | 0 |  |  |  |
| 100 | |  |  |  |  |  |  |  |  |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | Number of attributes | | | | |  |  |  |  |  |  |  | Number of attributes | | | | |  |  |  |  |  | ms in [13], [14], [27], [28], respectively. | | | | | | | | | | | | | | | | | | |  |  |  |  |  |
|  |  | (c) Decryption Time | | | | | | | |  | (d) Trapdoor Generation Time | | | | | | | | | | | | | In the Dec algorithm, LiST has a constant computa- | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | Test Time | | | |  |  |  |  |  |  |  |  | Trace Time | | | |  |  |  |  |  | tion cost 90.11ms to complete a decryption operation. | | | | | | | | | | | | | | | | | | | | | | |  |
| 3000 |  |  |  | (Public Cloud: PC) | | | | |  |  |  | 7000 |  |  |  |  | (PKG: PC) | | |  |  |  |  |  |  | However, the schemes in [13], [14], [27] require 51,657 | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2500 |  |  | LiST | |  |  |  |  |  |  |  | 6000 |  |  | LiST |  |  |  |  |  |  |  |  |  |  | ms, 71,167 ms and 28,209 ms to recover a message, | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  | [28] | |  |  |  |  |  |  |  |  |  | [13] |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | [27] | |  |  |  |  |  |  |  |  |  |  | [14] |  |  |  |  |  |  |  |  |  |  | respectively. The large computation overhead will | | | | | | | | | | | | | | | | | | | | | | |  |
| 2000 |  |  |  |  |  |  |  |  |  |  | (ms) | 5000 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  | 4000 |  |  |  |  |  |  |  |  |  |  |  |  |  | quickly consumes the battery of user’s resource lim- | | | | | | | | | | | | | | | | | | | | | | |  |
| 1500 |  |  |  |  |  |  |  |  |  |  | Time |  |  |  |  |  |  |  |  |  |  |  |  |  |  | ited mobile device. | | | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  | 3000 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1000 |  |  |  |  |  |  |  |  |  |  |  | 2000 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | The T rapdoor algorithm in LiST requires no bilinear | | | | | | | | | | | | | | | | | | | | | | |  |
| 500 |  |  |  |  |  |  |  |  |  |  |  | 1000 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0 0 | | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 | 0 0 | | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |  |  | paring or exponentiation computations. Only mul- | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  |  |  | Number of attributes | | | | |  |  |  |  |  |  |  | Number of attributes | | | | | |  |  |  |  | tiplication, division and inversion operations on Zp | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  |  | (e) Test Time | | | | |  |  |  |  |  |  |  | (f) Trace Time | | | | | |  |  |  |  |  | are calculated in smart phone, which consume only | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  |  |  |  | Fig. 5: Computation overhead | | | | | | | | | | | | |  |  |  |  |  |  |  |  | 0.026 ms, 0.093 ms and 0.057 ms, respectively. When | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  | jSj = 100, LiST only requires 0.501 ms to complete | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | a keyword trapdoor generation algorithm, while the | | | | | | | | | | | | | | | | | | | | | | |  |
| **Computation Efficiency.** | | | | | | | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  | schemes in [27], [28] needs 72,094 ms and 18,203 ms. | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |



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 in-comparable efficiency advantage. A non-uniform axis is also used in Fig. 5 for clear description.

As indicated in Table 7, our encryption time is

The above analysis shows that LiST has efficiency sig-nificantly better than the other schemes. The only exception is that the computation cost of T est 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 jSj are considered in Figure 6.a-6.d.

1545-5971 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications\_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2729556, IEEE Transactions on Dependable and Secure Computing

|  |  |
| --- | --- |
| IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING | 13 |

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 jSj = 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].

* **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 functional-ities, 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 Foun-dation of China under Grant No. 61402112, 61472307, 61472309, 61502086; the Singapore National Research Foun-dation under the NCR Award Number NRF2014NCR-NCR001-012; AXA Research Fund; Fujian Provincial Key Laboratory of Information Processing and Intelligent Con-trol (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 pre-serving 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 encryp-tion 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 Francis-co, 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 out-sourced 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 struc-tures,” 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 at-tributes,” 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 decryp-tion 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 Appli-cations, 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 Pro-ceedings 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 Designat-ed 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. Theo-ry 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 construc-tions,” 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 Y-our Right: Verifiable Attribute-based Keyword Search with Fine-grainedOwner-enforced Search Authorization in the Cloud,” IEEE

1545-5971 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications\_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2729556, IEEE Transactions on Dependable and Secure Computing

|  |  |
| --- | --- |
| IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING | 14 |

Transactions on Parallel and Distributed Systems, 2016, vol. 27, no.

4, pp. 1187 - 1198.

1. 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.
2. Y. Yang, S. Yang, F. Wang, J. Sun. “Post-Quantum Secure Public Key Broadcast Encryption with Keyword Search.” Journal of Infor-mation Science and Engineering 33.2 (2017).
3. Q Huang, H Li. “An Efficient Public-Key Searchable Encryption Scheme Secure against Inside Keyword Guessing Attacks.” Infor-mation Sciences (2017).
4. A. Ruhul. “Design of a Certificateless Designated Server Based Searchable Public Key Encryption Scheme.” Mathematics and Com-puting: Third International Conference, ICMC 2017, Haldia, India, January 17-21, 2017, Proceedings. Vol. 655. Springer, 2017.
5. 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).
6. A. Beimel, “Secure Schemes for Secret Sharing and Key Distribu-tion,” PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel, 1996.
7. D. Boneh and X. Boyen, “Short signatures without random or-acles,” 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.
8. J. Baek, R. Safavi-Naini, W. Susilo, “Public key encryption with keyword search revisited,” in International conference on Compu-tational Science and Its Applications. Springer Berlin Heidelberg, 2008: 1249-1259.
9. L. Guo, W.C. Yau, “Efficient secure-channel free public key en-cryption with keyword search for EMRs in cloud storage,” Journal of medical systems, 2015, 39(2): 1-11.
10. B. Lynn. The Stanford Pairing Based Crypto Library. [Online]. Available: http://crypto.stanford.edu/pbc, accessed May 7, 2014.
11. http://csrc.nist.gov/groups/ST/toolkit/keymanagement.html.
12. 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.

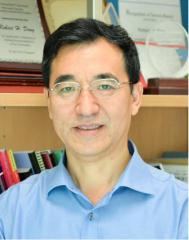
**Yang Yang** received the B.Sc. degree from Xi-dian University, Xi’an, China, in 2006 and Ph.D. degrees from Xidian University, China, in 2012. She is a research fellow (postdoctor) under su-pervisor Robert H. Deng in School of Information System, Singapore Management University. She is also an associate 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 Xi-dian 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 in-clude cloud security and big data security.



**Robert H. Deng** is AXA Chair Professor of Cy-bersecurity in the School of Information Sys-tems, 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 Se-curity, IEEE Transactions on Dependable and Secure Computing, the International Journal of Information Security, and IEEE Security and Pri-



vacy Magazine. He is Fellow of IEEE.

**Yingjiu Li** is an Associate Professor at theSchool of Information Systems, Singapore Man-agement University. His research interests in-clude 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 se-curity international journals (and conferences).



1545-5971 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications\_standards/publications/rights/index.html for more information.