CLOUD-ASSISTED MOBILE-ACCESS OF PRIVACY PRESERVING AUDITED HEALTH DATA

[1]TIRUPATI DIVYA, [2]THATIMAKULA SUDHA, [3]JALADANKI BABU

M.Tech, Dept. of Computer Science and Engineering, Sri Padmavati Mahila Visva Vidyalayam, Tirupati
Professor, Dept. of C.S. and Head(I/C) of C.S.E. and I.T., Sri Padmavati Mahila Visva Vidyalayam, Tirupati
Assistant Professor, Dept. of Computer Science and Engineering, Sri Padmavati Mahila Visva Vidyalayam-Tirupati
Email: tpt.divi@gmail.com, thatimakula_sudha@yahoo.com, jaladankibabu@gmail.com

Abstract— Motivated by the privacy issues, curbing the adoption of electronic healthcare systems and the wild success of cloud service models, we propose to build privacy into mobile healthcare systems with the help of the private cloud. Our system offers salient features including efficient key management, privacy-preserving data storage, and retrieval, especially for retrieval at emergencies, and auditability for misusing health data. Specifically, we propose to integrate key management from pseudorandom number generator for unlinkability, a secure indexing method for privacy preserving keyword search which hides both search and access patterns based on redundancy, and integrate the concept of attribute based encryption with threshold signing for providing role-based access control with auditability to prevent potential misbehavior, in both normal and emergency cases.

Keywords—Access control, Auditability, eHealth, Privacy.

I. INTRODUCTION

Fast access to health data enables better healthcare service provisioning, improves quality of life, and helps saving life by assisting timely treatment in medical emergencies. Anywhere-anytime-accessible electronic healthcare systems play a vital role in our daily life. Services supported by mobile devices, such as home care and remote monitoring, enable patients to retain their living style and cause minimal interruption to their daily activities. In addition, it significantly reduces the hospital occupancy, allowing patients with higher need of in-hospital treatment to be admitted. While these e-healthcare systems are increasingly popular, a large amount of personal data for medical purpose are involved, and people start to realize that they would completely lose control over their personal information once it enters the cyberspace.

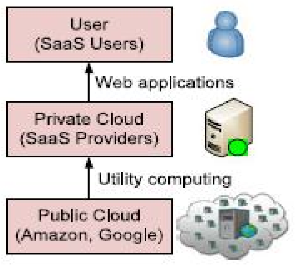


Fig. 1. SaaS service model.

According to the government website, around 8 million patients' health information was leaked in the past two years. There are good reasons for keeping medical data private and limiting the access. An employer may decide not to hire someone with certain diseases. An insurance company may refuse to provide life insurance knowing the disease history of a patient. Despite the paramount importance, privacy issues are not addressed adequately at the technical level and efforts to keep health data secure have often fallen short. This is because protecting privacy in the cyberspace is significantly more challenging.

Thus, there is an urgent need for the development of viable protocols, architectures, and systems assuring privacy and security to safeguard sensitive and personal digital information. Outsourcing data storage and computational tasks becomes a popular trend as we enter the cloud computing era. A wildly successful story is that the company's total claims capture and control (TC3) which provides claim management solutions for healthcare payers such as medicare payers, insurance companies, municipalities and self-insured employer health plans. TC3 has been using Amazon's EC2 cloud to process the data their clients send in (tens of millions of claims daily) which contain sensitive health information. Outsourcing the computation to the cloud saves TC3 from buying and maintaining servers, and allows TC3 to take advantage of Amazon's expertise to process and analyze data faster and more efficiently. The proposed cloud-assisted mobile health networking is inspired by the power, flexibility, convenience, and cost efficiency of the cloud-based data/computation outsourcing paradigm. We introduce the private cloud which can be considered as a service offered to mobile users. The proposed solutions are built on the service model shown in Fig. 1.

A software as a service (SaaS) provider provides private cloud services by using the infrastructure of the public cloud providers (e.g., Amazon, Google). Mobile users outsource data processing tasks to the private cloud which stores the processed results on the public cloud. The cloud-assisted service model supports the implementation of practical privacy mechanisms since intensive computation and storage can be shifted to the cloud, leaving mobile users with lightweight tasks.

A. Related Work

Some early works on privacy protection for e-health data concentrate on the framework design [2]–[6], including the demonstration of the significance of privacy for e-health systems, the authentication based on existing wireless infrastructure, the role-based approach for access restrictions, etc. In particular, identity-based encryption (IBE) has been used for enforcing simple role-based cryptographic access control. Among the earliest efforts on e-health privacy, Medical Information Privacy Assurance (MIPA) pointed out the importance and unique challenges of medical information privacy, and the devastating privacy breach facts that resulted from insufficient supporting technology. MIPA was one of the first few projects that sought to develop privacy technology and privacy-protecting infrastructures to facilitate the development of a health information system, in which individuals can actively protect their personal information. We followed our line of research with other collaborators and summarized the security requirements for e-health systems in. Privacy-preserving health data storage is studied by Sun *et al.*, where patients encrypt their own health data and store it on a third-party server. This work and Searchable Symmetric Encryption (SSE) schemes are most relevant to this paper. Another line of research closely related to this study focuses on cloud-based secure storage and keyword search. The detailed differences will be described later. The proposed cloud-assisted health data storage addresses the challenges that have not been tackled in the previously stated papers. There is also a large body of research works on privacy preserving authentication, data access, and delegation of access rights in e-health systems, while are most related to our proposed research. Lee and Lee proposed a cryptographic key management solution for health data privacy and security. In their solution, the trusted server is able to access the health data at any time, which could be a privacy threat. The work of Tan *et al.* is a technical realization of the role-based approach proposed in. The scheme that failed to achieve privacy protection in the storage server learns which records are from which patient in order to return the results to a querying doctor. Benaloh *et al.* Proposed the concept of patient-controlled encryption (PCE) such that health-related data are decomposed into a hierarchy of smaller piece of information which will be encrypted using the key which is under the patients' control. They provided a symmetric-key PCE for fixed hierarchy, a public-key PCE for fixed hierarchy, and a symmetric-key PCE for flexible hierarchy from RSA. The first public-key PCE for flexible hierarchy from pairings is proposed by Chu *et al.* [30]. The system of Li *et al.* [29] utilizes multi authority attribute-based encryption (ABE) proposed by Chase and Chow for fine-grained access control. Their system allows break-glass access via the use of "emergency" attributes. However, it is not clear who will take on the role of issuing such a powerful decryption key corresponding to this attribute in practice. The backup mechanisms in for emergency access rely on someone or something the patient trusts whose availability cannot be guaranteed at all times .More over, the storage privacy proposed in is a weaker form of privacy because it does not hide search and access patterns. The previously stated research works failed to address the challenges in data privacy, we aim to tackle in this paper. Finally, we also remark that there are other cryptographic mechanisms for privacy-preserving access of general data stored in a cloud environment.

II. PRELIMINARIES

A. Searchable Symmetric Encryption SSE allows data owners to store encrypted documents on remote server, which is modeled as honest-but-curious party, and simultaneously provides away to search over the encrypted documents. More importantly, neither the operation of outsourcing nor keyword searching would result in any information leakage to any party other than the data owner, thus achieving a sound guarantee of privacy. SSE was first put forward by Goh, and later improved by Curtmola *et al.* We base this study on Curtmola *et al.*'s construction. At a high level, SSE consists of the following algorithms.

KeyGen(*s*): This function is used by the users to generate keys to initialize the scheme. It takes the security parameter *s* and outputs a secret key *K*.

BuildIdx (*D*,*K*): The user runs this function to build the indexes, denoted by *I*, for a collection of document *D*. It takes the secret key *K* and *D* and outputs *I*, through which document can be searchable while remaining encrypted.

Trapdoor(*K*,*w*): The user runs this function to compute a trapdoor for a keyword *w*, enabling searching for this keyword. A trapdoor *Tw* can also be interpreted as a proxy for *w* in order to hide the real meaning of *w*. Therefore, *Tw* should leak the information about *w* as little as possible. The function takes the secret key *K* and the keyword *w* and outputs the respective trapdoor *Tw*. *Search*(*I*, *Tw*): This function is executed by the remote server to search for documents containing the user defined keyword *w*. Due to the use of the trapdoor, the server is able to carry out the specific query without knowing the real keyword. The function takes the built secure index *I* and the trapdoor *Tw* , and outputs the identifiers of files which contains keyword *w*. Concretely, in Curtmola *et al.*'s construction, each document is represented by an identifier and corresponds to a node. All documents in *D* are encrypted and stored in the remote servers. The index *I* is made up of two data structures, namely an array A, for storing the nodes, and a look-up table T, for keeping information that enables the remote server to locate the elements in A. All nodes are encrypted with random generated keys (different from the keys for encrypting the document) and stored as entries in A "scrambled" in a random order. However, to effectively organize the nodes, two measures are taken. 1) All the nodes whose respective files containing the same keyword *wi* are linked together in the linked list L*i* , and 2) each node contains the index in A as well as the random generated encryption key of next node in L*i* . Obviously, with the information contained in the first node, one will be able to decrypt all the nodes in the same linked list L*i* , and, thus, access all the respective file identifiers of files containing keyword *wi* . However, because the first node in the linked list does not have a previous node, the first node's index in A and its decryption key are stored in the field value of an entry in T, which is defined as a map _address, value. The field value is encrypted as it will be XOR-ed with an output of a pseudorandom permutation (PRP) function. The other field address is given by the output of a pseudorandom number generator to locate the first node. In other word, address serves as part of the trapdoor *Tw* to access the documents containing the respective keyword *w*. In fact, *Tw* consists of an output of a random number generator, for the purpose of locating entries in T, and an output of a PRP function, for the purpose of encrypting the entries, given the input *w* of pseudorandom algorithms. To set up SSE, the user runs BuildIdx, which constructs A and T based on the documents *D* in clear texts in ways said above. The user then stores A, T, and encrypted *D* in the remote server (clouds), none of which leaks information about the actual contents of the documents. To search document containing keyword *w*, the user run Search. Specifically, it uses Trapdoor to compute the respective trapdoor *Tw* and send the first part of *Tw* to the remote server. Upon receiving this information, the remote server uses it to locate and returns the respective encrypted entry in T. Then, the user uses the second part of *Tw* to decrypt the entry and get the information of the first node of the respective linked list. With that, the user can get all identifiers of wanted files, and, thus, retrieve and decrypt with the respective keys the encrypted files containing keyword *w*. *B. Threshold Secret Sharing*

Secret sharing is a mechanism for sharing secret information among multiple entities so that the cryptographic power is distributed which at the same time avoid single point of failure. For (k, n) threshold secret sharing, a piece of information I is divided into n pieces $I1, \ldots, In$, such that knowledge of any k or more of these Ii ($i \in [1, n]$) pieces can recover I, while knowledge of ($k - 1$) or fewer pieces keeps I completely undetermined [35]. Shamir [35] proposed such a scheme based on polynomial interpolation. Specifically, for the secret $I = a0$ is in a group G, randomly pick a ($k - 1$) degree polynomial

$y(x) = a0 +\_k-1\ i=1\ aixi$ (1)

with $a0 = I \in$G, and $a1, \ldots, ak-1 \in$G. Let $Ii = y(i)$, $i \in[1, n]$ and $\Phi \subseteq\{I1, \ldots, In\}$ with $|\Phi| \geq k$, where $|\cdot|$ denotes the cardinality of the given set. The Ii values in Φ and the indices i can be used to reconstruct the original information $I = y(0) = a0$ by computing

$y(x) = \_j \in\Psi\ \rho\Psi\ xj\ Ij$, (2)

where $\rho\Psi\ xj = \_l \in\Psi, l\_=j\ x-l\ j-l \in \mathbb{Z}q$ is the Lagrange coefficient for

a set $\Psi \subseteq\{1, \ldots, n\}$ with $|\Psi| \geq k$.

C. Identity-Based Encryption

A practical IBE scheme in the random oracle model was proposed by Boneh and Franklin. Identity-based systems allow any party to generate a public key from a known identity value, for example, the string "alice@xyz.com" for Alice. IBE makes it possible for any party to encrypt message with no prior distribution of keys between individuals. It is an important application of the pairing-based cryptography. Next, we review some technical details of Boneh-Franklin IBE. To set up IBE, we need to define the public parameters for the pairing groups. Let G1 be a group with prime order q, e : G1 × G1 → G2 be a bilinear map, and g be a generator ofG1. Let $\hat{}g = e(g, g) \in$G2. Let $H : \{0, 1\}* \rightarrow$ G1 and $h2 : \{0, 1\}* \rightarrow$ G2 be hash functions to be modeled as random oracles. The private key generator (PKG) in the IBE cryptosystems picks $s\ R\leftarrow \mathbb{Z}q$ as the private master key and gs as the master public key. When anyone wants to send a message m to Alice, she picks $r\ R\leftarrow \mathbb{Z}q$ and computes Encrypt((g, gs), "Alice",m) by (u, v) = (gr,$m \oplus h2$ (e(H("Alice"), gs)r)) which in turn equals to (gr,$m \oplus h2$ (e(H("Alice"), g)rs)) by bilinearity of e. Before decrypting the message, Alice needs to get her private key from PKG, who computes and send to Alice through a secure channel KeyExt(s, "*Alice*") = H("*Alice*")s With this private key, denoted by $w = H$("*Alice*")s , and a ciphertext (u, v), Alice now can decrypt it as Decrypt ((u, v), w)=$v \oplus h2$ (e(w, u))=$m \oplus h2$ (e(H("*Alice*"), g)rs) $\oplus h2$ (e(H("*Alice*")s, gr)) = $m \oplus h2$ (e(H("*Alice*"), g)rs) $\oplus h2$ (e(H("*Alice*"), g)rs) = m. The work of Boneh and Franklin has also described how to secret share the master secret key s [7]. Moreover, the private key corresponding to an identity string can also be viewed as a signature on a message by viewing the identity string as the message to be signed.

D. Attribute-Based Encryption

ABE has shown its promising future in fine-grained access control for outsourced sensitive data. Typically, data are encrypted by the owner under a set of attributes. The parties accessing the data are assigned access structures by the owner and can decrypt the data only if the access structures match the data attributes.

III. SYSTEM AND THREAT MODELS

A. System Model

The main entities involved in our system are depicted in Fig. 2. Users collect their health data through the monitoring devices worn or carried, e.g., electrocardiogram sensors and health tracking patches. Emergency medical technician (EMT) is a physician who performs emergency treatment. By user and EMT, we refer to the person and the associated computing facilities. The computing facilities are mainly mobile devices carried around such as smartphone, tablet, or personal digital assistant. Each user is associated with one private cloud. Multiple private clouds are supported on the same physical server. Private clouds are always online and available to handle health data on behalf of the users.

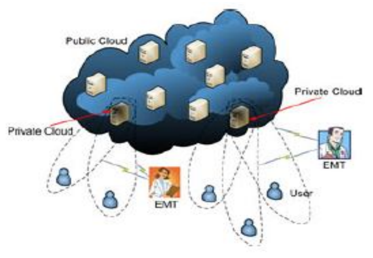


Fig. 2. Cloud-assisted mobile health network.

This can be very desirable in situations like medical emergencies. The private cloud will process the data to add security protection before it is stored on the public cloud. Public cloud is the cloud infrastructure owned by the cloud providers such as Amazon and Google which offers massive storage and rich computational resource. We assume that at the bootstrap phase, there is a secure channel between the user and his/her private cloud, e.g., secure home Wi-Fi network, to negotiate a long-term shared-key. After the bootstrap phase, the user will send health data over insecure network to the private cloud residing via the Internet backbone. Note that, we do not focus on the location privacy of mobile users which can be leaked when sending health data to the private cloud. There is a large body of location privacy schemes in the literature.

B. Threat Model

The private cloud is fully trusted by the user to carry out health data-related computations. Public cloud is assumed to be honest-but-curious, in that they will not delete or modify users' health data, but will attempt to compromise their privacy. Public cloud is not authorized to access any of the health data. The EMT is granted access rights to the data only pertinent to the treatment, and only when emergencies take place. The EMT will also attempt to compromise data privacy by accessing the data he/she is not authorized to. The EMT is assumed to be rational in the sense that he/she will not access the data beyond authorization if doing so is doomed to be caught. Finally, outside attackers will maliciously drop users' packets, and access users' data though they are unauthorized to.

C. Security Requirements

In this paper, we strive to meet the following main security requirements for practical privacy-preserving mobile healthcare systems.

1) *Storage Privacy:* Storage on the public cloud is subject to five privacy requirements.

a) *Data confidentiality:* unauthorized parties (e.g., public cloud and outside attackers) should not learn the content of the stored data.

b) *Anonymity:* no particular user can be associated with the storage and retrieval process, i.e., these processes should be anonymous.

c) *Unlinkability:* unauthorized parties should not be able to link multiple data files to profile a user. It indicates that the file identifiers should appear random and leak no useful information.

d) *Keyword privacy:* the keyword used for search should remain confidential because it may contain sensitive information, which will prevent the public cloud from searching for the desired data files.

e) *Search pattern privacy:* whether the searches were for the same keyword or not, and the access pattern, i.e., the set of documents that contain a keyword [15], should not be revealed. This requirement is the most challenging and none of the existing efficient SSE [14]–[17] can satisfy it. It represents stronger privacy which is particularly needed for highly sensitive applications like health data networks.

2) *Auditability:* In emergency data access, the users may be physically unable to grant data access or without the perfect knowledge to decide if the data requester is a legitimate EMT. We require authorization to be fine-grained and authorized parties' access activities to leave a cryptographic evidence.

IV. CLOUD-ASSISTED PRIVACY-PRESERVING EHEALTH

Our cloud-assisted privacy-preserving mobile healthcare system consists of two components: searchable encryption and auditable access control. Upon receiving the health data from users, the private cloud processes and stores it on public cloud such that storage privacy and efficient retrieval can be guaranteed. Next, the private cloud engages the bootstrapping of data access and auditability scheme with users so that it can later act on the users' behalf to exercise access control and auditing on authorized parties.

A. Storage Privacy and Efficient Retrieval

The first component is storage privacy for the health data. Our storage mechanism relies on secure index or SSE, so that the user can encrypt the data with additional data structures to allow for efficient search. It has been shown that the secure index-based approach is promising among different approaches for storage privacy. In our environment, the private cloud takes the role of user, and the public cloud is the storage server in SSE. Sun *et al.* Shows the feasibility of the secure index for health data storage privacy. Their approach followed the SSE of Curtmola *et al.* Which uses a linked-list data structure. However, there are practical issues that were unsolved which we will address in this paper.

1) The unlinkability requirement was not well addressed. None of the above works mentioned how to construct the file identifiers. If the identifiers bear certain pattern, it will be easy for the attackers to infer that multiple files are

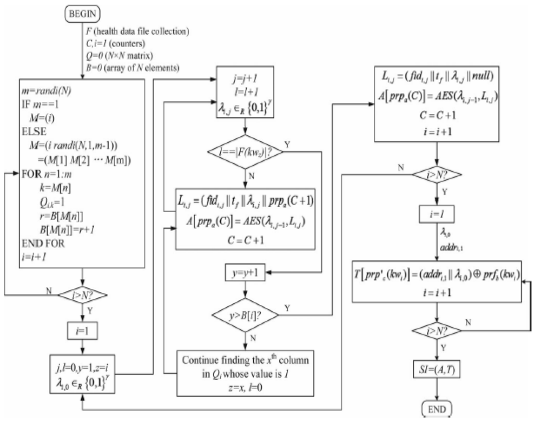

BEGIN
F (health data file collection)
C,i=1 (counters)
Q=0 (N×N matrix)
B=0 (array of N elements)
m=randi(N)
IF m==1
M=(i)
ELSE
M=(i randi(N,1,m-1))
=(M[1] M[2] ··· M[m])
FOR n=1:m
k=M[n]
Q_{i,k}=1
r=B[M[n]]
B[M[n]]=r+1
END FOR
i=i+1
i>N?
N
Y
i=1
j,l=0,y=1,z=i
λ_{i,0} ∈_R {0,1}^γ
j=j+1
l=l+1
λ_{i,j} ∈_R {0,1}^γ
l==|F(kw_z)|?
Y
N
L_{i,j} =(fid_{i,j} ‖ t_f ‖ λ_{i,j} ‖ prp_a(C+1))
A[prp_a(C)] = AES(λ_{i,j-1}, L_{i,j})
C=C+1
y=y+1
y>B[i]?
Continue finding the x^th column in Q_i whose value is 1
z=x, l=0
L_{i,j} =(fid_{i,j} ‖ t_f ‖ λ_{i,j} ‖ null)
A[prp_a(C)] = AES(λ_{i,j-1}, L_{i,j})
C=C+1
i=i+1
i>N?
i=1
λ_{i,0}
addr_{i,1}
T[prp'_c(kw_i)] = (addr_{i,1} ‖ λ_{i,0}) ⊕ prf_b(kw_i)
i=i+1
SI=(A,T)
END

Fig. 3. Pattern hiding secure index.

from a same user. Clearly, we need identifiers that appear random yet can be easily managed.

2) In traditional SSE, all stored data files are encrypted using the same key. This is not a sound security design since the more we use a key, the more information the attackers can obtain to break the key. We therefore need to update the key frequently enough to avoid the key wear-out.

3) To facilitate fast and efficient retrieval, it is desirable to construct the data files such that they could be searched by the date/time of creation, besides the keywords. This is particularly useful in emergencies where the search can be narrowed down to the most helpful data. Searching based on date/time should be treated differently from keywords since date/time is not strictly sensitive information and the privacy requirement can be relaxed for efficiency.

4) None of the existing relevant works could hide the search or access pattern as discussed before. The only SSE schemes that hide both patterns are proposed by Goldreich and Ostrovsky. These constructions are based on oblivious RAMs and are highly inefficient due the round complexity. We take a heuristic approach instead of hiding the search and access patterns instead of relying on relatively heavy cryptographic techniques. Our proposed pattern hiding scheme just slightly increases the computation and storage costs at the public cloud compared to the most efficient construction.

1) Constructing the Secure Index: The private cloud prepares data received from the user for privacy-preserving storage as follows. The private cloud constructs a secure index, SI, as shown in Fig. 3, for keyword search. SI consists of an array *A* and a lookup table *T*. *A*[□] = *d* (and similarly *T*[□]) denotes the value *d* stored in *A*[□]. The collection of linked lists *L = {Li* |*i* = 1, . . . , |*w*|*}* (where |*w*| denotes the size of the keyword space) is encrypted and stored in *A*. Furthermore, each linked list *Li* is a collection of nodes *Li,j* such that *Li = {Li,j* |*j* = 1, . . . , |*F*(*w*)|*}* (where |*F*(*w*)| denotes the number of data files containing *w*). Each linked list node contains three fields in Curtmola *et al.*'s construction [15], i.e., *Li,j* = (*fidi,j* _ *λi,j* _ *ptr*), where *fidi,j* is a unique file identifier, *λi,j* of length *γ* is the secret key used to encrypt the next node *Li,j*+1 in the linked list *Li* , and *ptr* contains the address of the encrypted *Li,j*+1 (i.e., Enc*λi , j* (*Li,j*+1), where Enc can be a symmetric-key encryption algorithm such as AES). Finally, *λi,*0 for each *i* will be stored in the lookup table *T* in an encrypted form. Different from Curtmola *et al.*'s construction, we use file identifiers that appear random so that the attackers cannot link multiple stored files to a same user. The private cloud will pick (*a*, *b*, *c, η*), each of them serves as a key for either a pseudorandom function (PRF) or a PRP. The private cloud inputs a secret seed *η* into the PRF and obtains two outputs _ = PRF(*η,* 1) and *v* = PRF(*η,* 2). The outputs _ and *v* will be used as the seeds for generating the update keys *sf* and the file identifiers *fid*, respectively. Specifically, *fid* = PRF(*v, k*), 1 ≤ *k* ≤ |*F*|, where |*F*| denotes the number of data files in the collection *F*. The first node *Li,*1 is addressed by *addri*,1 . The pointer *ptr* indicates the index location in *A*[□] and is the output of a pseudorandom permutation *prpa* () computed from the private cloud's secret *a*. Similarly, *prp_c* () is another PRP computed from the secret *c* for index location of *T*[□]. The keyword is encrypted by a pseudorandom function *prfb* () computed from the secret *b*.

2) Encrypting the Data Files: We added a time tag *tf* to a linked list node. The time tag infers which update key was used to encrypt the corresponding file and facilitates the search by the date/time of creation of the data. The time tag *tf* is in the form of month/day/year, e.g., 06/23/1997. The seed key *w* is first used to generate the year key *K*year = PRF(w, year), which is then used to generate the month key *K*month =
PRF(*K*year , month), which is finally used to generate the day
key *K*day = PRF(*K*month, day). In our design, data files created on the same day are encrypted using the same update key, i.e., *sf* = *K*day. However, using the above evolving key method, finer time scale can be used to generate the update keys. This is a design issue depending on how many files can be encrypted with the same key before considering the key "worn-out", i.e., not secure any more. Using the time tag, the private cloud can not only determine if a particular file is of interest but also efficiently derives the update key *sf* from the root seed *η*. The private cloud appends the identifier *fid* to each encrypted file and stores the result on the public cloud. *3) Hiding the Patterns:* The idea is to extend a linked list to contain other keywords in addition to the intended one. For

example, linked list Li is supposed to be for the files containing the ith keyword wi in the keyword space, i.e., Li contains only nodes Li,j , $\forall j \in [1, |F(wi)|]$. In the proposed pattern hiding scheme, each linked list will contain multiple (but not the same number of) keywords and each keyword will appear in multiple (but not the same number of) linked lists, e.g., Li is now constructed to include two other keywords wg and wh . The new Li should contain all nodes for the three keywords, i.e., Li,j , $\forall j \in [1, |F(wi)|]$, Lg,j , $\forall j \in [1, |F(wg)|]$, and Lh,j , $\forall j \in [1, |F(wh)|]$. To search for wh -related files, the private cloud can deliberately submit a trapdoor calculated from wi. As the file identifiers associated with all the three keywords will be returned, the private cloud can select the ones containing the desired keyword wh . Similarly, since wi is contained in other linked lists, say Lo , the private cloud can submit a search based on wo to disguise the actual search for wi-related files. The pattern hiding scheme is described as follows for each keyword i in the keyword space:

1) Randomly select an integer m between 1 and $N = |w|$, Where
N is also the number of linked lists to be constructed. The integer m determines how many different linked lists will contain wi.

2) Then, randomly generate an array of $m - 1$ integers between 1 and N, indicating which linked lists will contain wi besides Li . We can keep running this process until we have m distinct integers. Suppose $i = 1$, $N = 8$, $m = 4$, and the array of integers $M = (1\ 8\ 4\ 5)$. The array M shows the positions of wi , i.e., wi is contained in the ith, 4th, 5th, and 8th linked lists.

3) Record the positions of wi in a matrix Q by setting the corresponding elements to 1 (otherwise 0), e.g., $Qi,k = 1$ represents the ith keyword that is contained in the kth linked list. Summation of the columns of Q indicates how many different keywords are contained in the corresponding linked lists. Using the above example, the 1st row of Q is $Q1 = (1\ 0\ 0\ 1\ 1 0\ 0\ 1)$. Suppose as the process continues, $Q2 = (0\ 1\ 0\ 1\ 0 0\ 0\ 1)$, then $B = Q1 + Q2 = (1\ 1\ 0\ 2\ 1 0\ 0\ 2)$ indicating that the 1st, 2nd, and 5th linked lists contain one keyword, the 4th and 8th linked lists contain two keywords, and so on.

4) The actual construction of linked lists is based on array B and matrix Q. Suppose the 1st linked list $L1$ contains three keywords $w1$, $w7$, and $w8$. We start the construction by linking all nodes for $w1$ first. The last node for $w1$, $L1,|w1|$, will be linked to the first node for $w7$ (or $w8$), $L7,1$ (or $L8,1$), etc. We summarize the construction of the proposed pattern hiding secure index, performed by the private cloud, in Fig. 3. We use $randi(N)$ and $randi(N, 1, m - 1)$ to denote randomly generating an integer between 1 and N, and randomly generating a $1 \times (m - 1)$ matrix with elements between 1 and N, respectively.

4) Retrieving the Data Files: The private cloud retrieves the data files upon request on behalf of the user. Suppose files containing "diabetes" are desired, wi = "*diabetes*." In the original retrieval without pattern hiding, the private cloud computes a trapdoor for "diabetes", TD("*diabetes*") = ($prp\_ c$("*diabetes*"), $prfb$("*diabetes*")) and sends it to the public cloud. The public cloud uses $T[prp\_ c$("*diabetes*")] $\oplus$ $prfb$("*diabetes*") to obtain ($addri,1\ \_\ \lambda i,0$) which is used to locate and decrypt linked list Li for "*diabetes*" The public cloud will then be able to obtain the addresses and secret keys for all the following nodes in this linked list. After the whole linked list is decrypted, the time tag is used by the public cloud to determine if a particular file is within the time range of the request submitted by the private cloud. The associated *fid*'s are then used to find the corresponding encrypted files. The files and their time tags are finally returned to the private cloud. In the retrieval with pattern hiding, the private cloud first looks up Q to find the columns whose ith row is 1. The private cloud then selects any one of these columns, say, the jth, and submits $TD(wj)$ instead of TD("*diabetes*") to the public cloud. The next time the private cloud searches for "diabetes," it can select a different column whose ith row is 1. After a further selection based on the time range, the public cloud returns the encrypted files which also contain "diabetes"-related files. The private cloud regenerates the update keys based on the time tags to decrypt the files. Since the decrypted results may include files of other keywords, e.g., $F(wj)$, we let the private cloud append descriptive file identifiers, e.g., "Diabetes_10" and "Diabetes_18" to the data files before encryption. We call the descriptive identifiers inner identifiers which are encrypted with the data, and the *fid*'s outer identifiers which are left outside of the encryption. The process of constructing the secure index and using it for retrieval is shown in Fig. 4. This figure does not include the construction of encrypted data files.

B. Data Access Privacy and Auditability

The second component is the data access during emergencies where the EMT requests data through the private cloud. The proposed approach is for the general data access, although we focus on the emergency access since it is more challenging. The emergency access supported by Sun *et al.* is based on a personal device which is subject to theft, loss, or dead battery, and cannot meet the requirement of anytime anywhere accessibility. Existing papers, most relevant to our data access component have followed the approach to define a set of attributes for each single data file. Each file is then directly encrypted under the associated attributes by ABE or encrypted by a different key which is in turn encrypted under the attributes by ABE. There are some significant drawbacks of this approach. First of all, users (or data owners) are not in a good position to determine who needs access to which data files. This is one of the most prominent features of health data access which requires flexibility and professional judgment. Second, the authenticity of the attributes cannot be verified which is a very practical problem and highly challenging in the proposed mobile health networks, where a set of attributes is defined for each general role (e.g., primary physician, EMT, and insurance provider) that will access the data

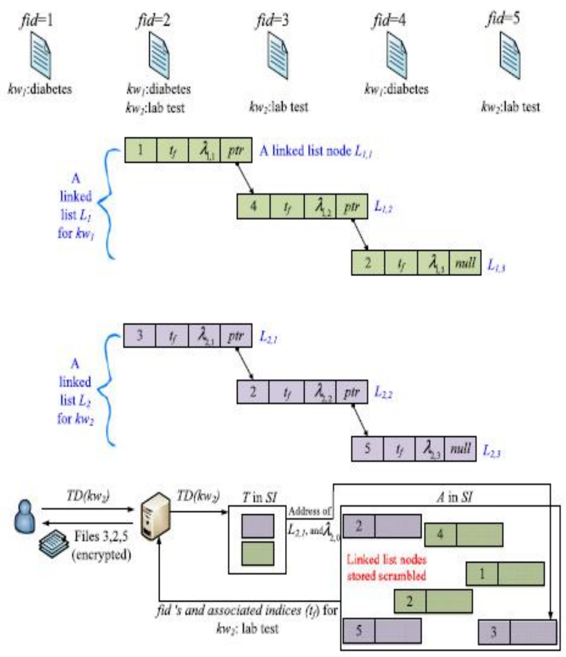


Fig. 4. Example of the construction process of secure index for five files sorted by two keywords, two linked lists each created for a keyword, and a search based on the keyword "lab test." (Legend: *T* is SI is used to find the address of the first linked list node *L*2,1 stored in *A*. The symmetric key *λ*2,0 is used to decrypt this node. Shaded nodes are encrypted.)

For example, a userwould like to grant data access to someone who is a *pediatrician*, has more than *ten years* experience, works in *the Bay Area*, and accepts the *Blue Cross and Blue Shield* or *IGNACIO* insurance plan. How does the private cloud verify, at the time of data access, that the person indeed has the attributes he/she claims? Third, using the ABE-based access control alone cannot audit who has accessed which data. ABE serves as a gatekeeper to prevent unauthorized parties from decrypting the data. However, it does not provide any mechanism for auditability, i.e., to record and prove that an authorized party has accessed certain data. Without auditability, it is not possible to identify the source of breach if authorized parties illegally distribute the health data which will be discussed in our future research issues. Furthermore, in our use of ABE, the user (and his/her primary physician) will have no clue about whether an authorized party has properly accessed the data without auditability. To overcome these difficulties, we propose to combine threshold signature with ABE-based access control. A(*k*, *n*) threshold signature (e.g., guarantees that a valid signature on a message can be generated as long as there are *k* valid signature shares. For instance, we can set *n* = 5 representing the private cloud, the primary physician, the EMT, the specialists (e.g., paediatrician and urologist), and the insurance provider. The private cloud and primary physician are fully trusted by the user. Let *k* = 2 such that any not fully trusted party must perform the threshold signing with either fully trusted party. In reality, for example, the EMT better performs the signing with the private cloud because the Primary physician may not be available online at all times. On the other hand, a pediatrician better performs the signing with the primary physician since users normally rely on their primary physicians for referral to a specialist. We do not further elaborate on this issue but use the emergency access case to describe the detailed design. The user serves as the trust dealer in the threshold signature to assign each participating party a secret share that is essential for generating the valid signature share. In our design, users do not encrypt their health data using ABE. The health data is encrypted using the very efficient method described in our storage privacy component. Instead, users use ABE to encrypt the secret shares so that only authorized parties can decrypt them and generate valid signatures. The private cloud and EMT will threshold-sign the data access request submitted by the EMT which contains the keywords and time range the EMT wishes to search. The user can check the request and the validity of the threshold signature to audit the following at a later time: 1) the request was due to a true medical emergency, 2) the EMT has requested data only pertinent to the treatment, 3) the EMT cannot deny the data request and access if either 1) or 2) is violated, and 4) the private cloud cannot falsely accuse the EMT if neither 1) nor 2) is violated. In doing so, users avoid the daunting task of determining who can access which data file(s). Instead, they only need to determine who can access their data and assign a secret share correspondingly. Whether an authorized party has properly accessed the data is left to the auditability in our design. We also propose to leverage the existing healthcare system architecture to verify the authenticity of the attributes.

1) ABE-Controlled Threshold Signing: The user secretshares a key to *n* participating parties.

1) User defines some parameters for ABE-controlled threshold signing. Let *H*: {0, 1}* → G be a hash function. Let G1 be a bilinear group of prime order *p*1 , *g* and *g*1 be generators of G1 , and *e* : G1 × G1 → G2 be a bilinear map.

2) User (*k*, *n*)-shares *x* such that any subset *S* of *k* or more can reconstruct *x* using the Lagrange interpolation:

x = _*i* ∈*S Lixi* , where *Li* are the appropriate Lagrange coefficients for the set *S*, and *xi* are the secret shares.

3) User ABE-encrypts the secret share *xd* for EMT, denoted by ABE(*xd*), as: Define the universe of attributes $U = \{1, 2, . . . , u\}$ and a hash function $h : \{0, 1\}* \rightarrow$ G2 .
Randomly choose a number $vj \in R$ Z*p*1 for each attribute $j \in U$ and a number $z \in R$ Z*p*1. The public parameters are *V*1 = *gv*1 1 , . . . , *Vu* = *gvu* 1 , *Y* = *e*(*g*1, *g*1)*z* , and the master secret key is (*v*1, . . . , *vu*, *z*). Obtain the encrypted share for EMT as ABE(*xd*) = (_, *xdY τ* , *{V τ} j*∈_*q*), where *q* is a set of attributes and *τ* ∈*R* Z*p*1 *j* is a randomly chosen secret value.
4) User generates the decryption key *D* for EMT using the ABE key generation algorithm [36] and sends (ABE(*xd*), IBERole(*D*)) to the private cloud, where IBERole is the IBE [7] using the general role Role = EMT as the public key.
5) When EMT requests medical data from the private cloud, EMT sends the attributes q, the attribute certificate (q)*SIG*, and *REQ* which contains the keyword for search and the time range of interest. The private cloud verifies *q* using (*q*)*SIG* and returns (ABE(*xd*), IBERole(*D*)) to EMT.
EMTfirst decrypts for*D*using the private key corresponding to the role "EMT," and then decrypts for *xd* using *D*.
6) Private cloud and EMT each generates partial threshold signatures *σi* = (*H*(*REQ*))*xi* , and exchange *σi* and *yi* = *gxi* . They verify the partial signature from each other by checking if (*g, yi*,*H*(*REQ*)*, σi*) is a valid Diffie–Hellman tuple [7].
7) Private cloud and EMT generate the threshold signature *σ* = _*i*∈*S* (*σLi i*) which can be verified by anyone by checking if (*g, y*,*H*(*REQ*)*, σ*) is a valid Diffie–Hellman tuple. The private cloud stores *σi* from EMT, *σ*, *REQ*, and the date/time request is made.
8) Private cloud submits a trapdoor *TD*(*w*) for keyword *w* in *REQ* to public cloud. The private cloud also extracts the time range of interest specified in *REQ*, submits the time tags falling in the time range to public cloud, and regenerates the update keys *sf* 's based on the time tags.
9) Upon receiving the encrypted files from public cloud, the private cloud decrypts the files using the appropriate update keys, re-encrypts the files using the shared-key with EMT established after verifying the attributes, and returns the results to EMT for decryption. The computational load on the mobile user is light since secret sharing needs to be performed once and for all, and the ABE encryption of the shares needs to be performed only for a limited number of general roles.
2) Attribute Verification and Role-Based Encryption: Since the user has no way of knowing which specific person will request data access, it is impossible for the user to authenticate the attributes claimed by the person before ABE-encrypting the secret share. The authentication of the attributes, i.e., verifying (q)SIG, is left to the private cloud when data access is requested. However, in reality, there is likely no trust authority shared by the private cloud and EMT, rendering the authentication of the attributes mission-impossible. Similarly, it is impossible for the user to encrypt the ABE decryption key *D* before knowing who the EMT will be. We take a first step in addressing these challenging issues by leveraging role-based encryption and the healthcare system architecture as proposed by Sun *et al.* [8]. With such an architecture, the

TABLE I
NOTATIONS FOR EFFICIENCY ANALYSIS

S_g	Bit size of an output of pseudorandom generators
S_s	Bit size of a (partial) signature
S_F	Average bit size of a data file
S_{ABE}	Bit size of the ABE a secret share
S_{IBE}	Bit size of the IBE encryption of the ABE encryption key
S_{Att}	Bit size of an attribute certificate
S_l	Bit size of a linked list node
S_{LT}	Bit size of an entry in the lookup table T
S_{Arr}	Bit size of an entry in the array A
N_p	Minimum number of parties required to generate a signature
N_A	Number of attributes used in ABE
N_f	Number of files
N_k	Number of keywords
$N_{k/l}$	Average number of keywords per linked list
$N_{k/f}$	Average number of keywords per file
$N_{f/k}$	Average number of files per keywords

TABLE II
STORAGE OVERHEAD TO OUTSOURCE A COLLECTION OF *N* FILES

Items	Size
T	$N_k S_{LT}$
A	$N_f S_{Arr}$
Linked lists	$N_k N_{kl} N_{fk} S_l$

attributes associated with a particular EMT can be certified (i.e., signed) by the trust authority of its domain, e.g., veterans health administration (VHA) in [8]. Since the domain public parameters are available online, the private cloud can download the parameters which are necessary for verifying the signature on the attributes. Any provably secure digital signature scheme (e.g., [7]) can serve the certification purpose. The role-based technique allows the user to encrypt the decryption key *D* without the knowledge of the specific EMT. However, the user needs to know which trust domain the EMT belongs to in order to compute IBERole(*D*). Since the location of an emergency is unpredictable, the EMT and his/her associated trust domain cannot be predicted. This problem can also be solved by the healthcare architecture by letting the entities in Level 1 (e.g., VHA and regional health information organizations) serve as the role certification authority for their responsible domains. Since these Level 1 authorities are limited in number, it is possible for the user to download the domain parameters necessary for computing IBERole(*D*) from each of the Level 1 authorities.

V. SECURITY ANALYSIS

A. Storage privacy

The proposed approach guarantees the five storage privacy requirements. First, since the data are encrypted, unauthorized parties cannot learn the content of the stored data. Second, our file identifiers are numeric values that do not divulge any information about the file content or the ownership. So multiple data files cannot be linked by their identifiers. Third, by adding redundancy to the linked lists, the adversaries can hardly tell if the searches were for the same keyword, or if a set of data files contain a same keyword. The fourth requirement, i.e., the storage/retrieval anonymity can be easily satisfied because the private cloud performs the storage/retrieval for all the users it supports and no particular user can be associated with any storage/retrieval processes.

TABLE III
COMMUNICATION OVERHEAD FOR A SUCCESSFUL DATA ACCESS REQUEST

Communicating parties	**Overhead**	**Note**
EMT and private cloud	$N_A S_{Att} + S_s + S_{ABE} + S_{IBE} + N_p S_s$	ϱ and $SIG(\varrho)$ $ABE(x_d)$ $IBE(D)$ N_p partial signatures
Private cloud and public cloud	$2S_g + (N_{kl} - 1)N_{fk}S_F$	A trapdoor Redundant files for pattern hiding

Finally, the keyword used for search is encrypted in the trapdoor, and thus, no sensitive information is revealed.

B. Data Access Privacy and Auditability

Fine-grained access control is achieved by our ABE-control threshold signing scheme, where the expensive ABE operations are only used for encrypting small secret values and the majority of data encryption is fulfilled by efficient symmetric key scheme. The threshold signature exchange used in our scheme enables the private cloud to record evidence that is signed by the authorized parties which can be used as audit logs. By having the private cloud and EMT both signing the EMT's data access requests, users can later check whether the request is legitimate and appropriate, and simultaneously, be assured that the EMT cannot deny a request and the private cloud cannot falsely accuse an EMT. Since the mobile users outsource most of their computations to the private cloud and most storage to the public cloud, the computation and storage costs at the mobile side are expected to be highly practical. Note that a downside of being cost-efficient is the potential security breach if the private cloud acts maliciously. With our current schemes, as long as the private cloud is honest, our privacy guarantees cannot be broken even if all entities collude. We argue that a private cloud, by definition, should be highly trustworthy. Otherwise, it is difficult to attract users to pay for the service. As part of our future work, we will investigate the impact of relaxing trust on the private cloud and consequently, the tradeoff between security and efficiency.

VI. PERFORMANCE EVALUATION

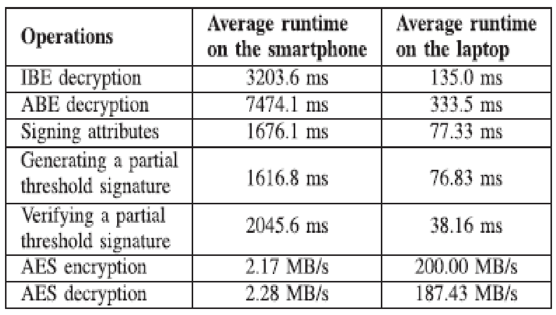
A. Storage and Communication Efficiency

We analyze the storage and communication efficiency by looking at the storage and communication overheads during data outsourcing and retrieval. The overhead is defined to be any information that serves the purposes of management, security, bookkeeping, etc., but the essential healthcare data or its encryption. For ease of presentation, we list in Table I notations of parameters that we will use in the analysis. The storage overhead is mainly due to the use of *Secure Index*, which employs linked lists, the lookup table *T*, and an array *A*.
We summarize the storage overhead in Table II. The overall storage overhead for outsourcing *N* files with our scheme is trivially obtained by summing up all the overheads, which is given by *Nf SArr + NkSLT + NkNklNf kSl*. As *Nk · Nf k = Nf* and*Nk < Nf* , the overhead becomes *Nf SArr + NkSLT + NfNklSl = O*(*Nf*). We also investigate the communication overhead during an EMT's data request with a successful retrieval. For clarity, we decompose the communication into two parts, i.e., communication between data requesters, such as EMT, and the private cloud and that between the private cloud and the public cloud. The respective communication overheads are illustrated in Table. III. It is worth mentioning that although, as we can see from the table, the pattern hiding requires retrieving redundant files during data retrieval, which seems to significantly contribute to the overhead, it takes place only between the private and public cloud where the wired intercloud connection is stable and fast, making the increased data transferring time negligible. On the other hand, the private cloud sends only the requested file to EMT (possibly through wireless channels, which are relatively less predictable and of lower capacity). Therefore, it does not affect the overall performance very much. From the analysis above, we know that the storage overhead is linear with the number of outsourced healthcare data files, while the communication overhead can be considered as constant per data request. The result indicates that the proposed scheme is efficient as well as scalable.

B. Computation Efficiency

In this section, we analyze the computational efficiency of proposed schemes. Specifically, we are interested in whether our schemes are efficient when mobile devices are involved, i.e., patients preparing the privacy-preserving storage and EMTs accessing the medical data in emergencies. We implemented our schemes using Samsung Nexus S smart phones (1-GHz Cortex- A8, 512-MB RAM) and measured the runtime. For implementations of IBE and ABE, we used the Java Paring-Based Cryptography Library [42] and used a pairing-friendly type-A 160-bit elliptic curve group. In privacy-preserving storage leveraging patient mobile devices, efficient secret key operations are mainly involved which we will not focus on in the evaluation. In emergency medical data access leveraging EMT mobile devices, the most costly real-time computation includes IBE decryption and ABE decryption, generating a regular signature on attributes and a partial threshold signature on the access request, and verifying the partial threshold signature from the private cloud. However, IBE decryption, ABE decryption, and regular signature can be performed once and for all access for the same patient, which is beneficial if the EMT will issue multiple access requests. We still take this cost into account since an EMT is likely to access a patient's medical data only once in many cases.

TABLE IV
RUNTIME OF CRYPTOGRAPHIC OPERATIONS ON EMT'S MOBILE DEVICES



Operations	Average runtime on the smartphone	Average runtime on the laptop
IBE decryption	3203.6 ms	135.0 ms
ABE decryption	7474.1 ms	333.5 ms
Signing attributes	1676.1 ms	77.33 ms
Generating a partial threshold signature	1616.8 ms	76.83 ms
Verifying a partial threshold signature	2045.6 ms	38.16 ms
AES encryption	2.17 MB/s	200.00 MB/s
AES decryption	2.28 MB/s	187.43 MB/s

Following parameters are assumed: the number of attributes $n_{attr} = 5$, the minimal number of secret shared needed to recover the secret $k = 2$, key size of the AES is 128 bits, key size of RSA (for signatures) is 512 bits.

We summarize the most costly real-time computation on EMT mobile devices in Table IV. The Smartphone we used is not the latest model. The runtime is expected to improve with newer and more powerful models. For comparison, we also provide in the table the runtime of the same implementation on a laptop (Intel Core i5, 4-GB RAM), which can also be regarded as a mobile device. Roughly, for each access, it takes around 16 s to perform the required cryptographic computation using the chosen Smartphone and around 0.6 s on the laptop, both of which are acceptable for an efficient retrieval of electronic healthcare records.

CONCLUSION

In this paper, we proposed to build privacy into mobile health systems with the help of the private cloud. We provided a solution for privacy-preserving data storage by integrating a PRF based key management for unlink ability, a search and access pattern hiding scheme based on redundancy, and a secure indexing method for privacy-preserving keyword search. We also investigated techniques that provide access control (in both normal and emergency cases) and audit ability of the authorized parties to prevent misbehaviour, by combining ABE-controlled threshold signing with role-based encryption. As future work, we plan to devise mechanisms that can detect whether users' health data have been illegally distributed, and identify possible source(s) of leakage (i.e., the authorized party that did it).

REFERENCES

[1] U.S. Department of Health & Human Service, "Breaches Affecting 500 or More Individuals," (2001). [Online]. Available: http://www.hhs.gov/ocr/ privacy/hipaa/administrative/breachnotificationr ule/breachtool.html

[2] P. Ray and J.Wimalasiri, "The need for technical solutions for maintaining the privacy of EHR," in *Proc. IEEE 28th Annu. Int. Conf.*, New York City, NY, USA, Sep. 2006, pp. 4686–4689.

[3] M. C. Mont, P. Bramhall, and K. Harrison, "A flexible role-based secure messaging service: Exploiting IBE technology for privacy in health care," presented at the 14th Int. Workshop Database Expert Syst. Appl., Prague, Czech Republic, 2003.

[4] G. Ateniese, R. Curtmola, B. de Medeiros, and D. Davis, "Medical information privacy assurance: Cryptographic and system aspects," presented at the 3rd Conf. Security Commun. Netw., Amalfi, Italy, Sep. 2002.

[5] L. Zhang, G. J. Ahn, and B. T. Chu, "A role-based delegation framework for healthcare information systems," in *7th ACM Symp. Access Control Models Technol.*, Monterey, CA, USA, 2002, pp. 125–134.

[6] L. Zhang, G. J. Ahn, and B. T. Chu, "A rule-based framework for rolebased delegation and revocation," *ACM Trans. Inf. Syst. Security*, vol. 6, no. 3, pp. 404–441, 2003.

[7] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing. Extended abstract in CRYPTO 2001," *SIAM J. Comput.*, vol. 32, no. 3, pp. 586–615, 2003.

[8] J. Sun, C. Zhang, Y. Zhang, and Y. Fang, "An identity-based security system for user privacy in vehicular ad hoc networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 9, pp. 1227–1239, Sep. 2010.

[9] J. Sun, X. Zhu, and Y. Fang, "Preserving privacy in emergency response based on wireless body sensor networks," in *Proc. IEEE Global Telecommun. Conf.*, Dec. 2010, pp. 1–6.

[10] J. Sun, X. Zhu, and Y. Fang, "Privacy and emergency response in ehealthcare leveraging wireless body sensor networks," *IEEE Wireless Commun.*, vol. 17, no. 1, pp. 66–73, Feb. 2010.

[11] J. Sun, X. Zhu, C. Zhang, andY. Fang, "HCPP: Cryptography based secure EHR system for patient privacy and emergency healthcare," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst.*, Jun. 2011, pp. 373–382.

[12] L. Guo, C. Zhang, J. Sun, and Y. Fang, "PAAS: Privacy-preserving attribute-based authentication system for eHealth networks," in *Proc. IEEE Intl. Conf. Distrib. Comput. Syst.*, Jun. 2012, pp. 224–233.

[13] J. Sun, X. Zhu, C. Zhang, and Y. Fang, Security and Privacy for Mobile Healthcare (m-Health) Systems, in *Handbook on Securing Cyber-Physical Infrastructure*, S. Das, K. Kant, and N. Zhang, Eds. Amsterdam, The Netherlands: Elsevier, 2011.

[14] E.-J. Goh, "Secure indexes," *IACR Cryptology ePrint Archive*, vol. 2003, p. 216, 2003.

[15] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions

and efficient constructions," presented at the ACM Conf. Comput. Commun. Security, Alexandria, VA, USA, 2006.

[16] Y. C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *Proc. 3rd Int. Conf. Appl. Cryptogr. Netw. Security*, 2005, pp. 442–455.

[17] D. Song, D.Wagner, and A. Perrig, "Practical techniques for searching on encrypted data," in *Proc. IEEE Symp. Security Privacy*, 2000, pp. 44–55.

[18] O. Goldreich and R. Ostrovsky, "Software protection and simulation on oblivious RAMs," *J. ACM*, vol. 43, pp. 431–473, 1996.

[19] R. Ostrovsky, "Efficient computation on oblivious RAMs," in *Proc. ACM Symp. Theory Comput.*, 1990, pp. 514–523.

[20] C. Wang, K. Ren, S. Yu, and K. Urs, "Achieving usable and privacyassured similarity search over outsourced cloud data," in *Proc. IEEE Conf. Comput. Commun.*, Mar. 2012, pp. 451–459.

[21] N. Cao, Z. Yang, C.Wang, K. Ren, andW. Lou, "Privacy-preserving query over encrypted graph-structured data in cloud computing," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst.*, Jun. 2011, pp. 393–402.

[22] X. Liang, R. Lu, X. Lin, and X. S. Shen, "Patient self-controllable access policy on PHI in ehealthcare systems," *Adv. Health Inform. Conf.*, pp. 1–5, Apr. 2010.

[23] M. Katzarova and A. Simpson, "Delegation in a distributed healthcare context: A survey of current approaches," in *Proc. 9th Int. Conf. Inform. Security*, 2006, pp. 517–529.

[24] I. Foster, C.Kesselman,G. Tsudik, and S. Tuecke, "A security architecture for computational grids," in *Proc. ACMConf. Comput. Commun. Security*, San Francisco, CA, USA, 1998, pp. 83–92.

[25] X. Liang, R. Lu, L.Chen, X. Lin, andX. Shen, "PEC:Aprivacy-preserving emergency call scheme formobile healthcare social networks" *J. Commun. Netw.*, vol. 13, no. 2, pp. 102–112, 2011. L. Guo, C. Zhang, J. Sun, and Y. Fang, "A privacy-preserving attribute-based authentication system for mobile health networks," *IEEE Trans. Mobile Comput.*, vol. PP, no. 99, pp. 1–1, 2013.

[26] W.-B. Lee and C.-D. Lee, "A cryptographic key management solution for HIPAA privacy/security regulations," *IEEE Trans. Inf. Technol. Biomed.*, vol. 12, no. 1, pp. 34–41, Jan. 2008.

[27] C. C. Tan, H.Wang, S. Zhong, and Q. Li, "Body sensor network security: An identity-based cryptography approach," in *Proc. ACM Conf. Wireless Netw. Security*, Apr. 2008, pp. 148–153.

[28] J. Benaloh, M. Chase, E. Horvitz, and K. Lauter, "Patient controlled encryption: Ensuring privacy of electronicmedical records," in *Proc. ACM Workshop Cloud Comput. Security*, 2009, pp. 103–114.

[29] M. Li, S. Yu, Y. Zheng, K. Ren, andW. Lou, "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 1, pp. 131–143, Jan. 2013.

[30] C.-K. Chu, S. S. M. Chow, W.-G. Tzeng, J. Zhou, and R. H. Deng, "Keyaggregate cryptosystem for scalable data sharing in cloud storage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 99, no. PrePrints, p. 1, 2013. Available: http://doi.ieeecomputersociety.org/10.1109/TPDS.2013.112

[31] M. Chase and S. S. M. Chow, "Improving privacy and security in multiauthority attribute-based encryption," in *Proc. ACM Conf. Comput. Commun. Security*, 2009, pp. 121–130.

[32] S. S. M. Chow, "New privacy-preserving architectures for identity-/attribute-based encryption" Ph.D. dissertation, Courant Inst. Math. Sci., New York University, New York, NY, USA, 2010.

[33] S. S. M. Chow, Y. J. He, L. C. K. Hui, and S.-M. Yiu, "SPICE—Simple privacy-preserving identity-management for cloud environment," in *Proc. 10th Int. Conf. Appl. Cryptography Netw. Security*, 2012, pp. 526–543.

[34] S. S. M. Chow, C.-K. Chu, X. Huang, J. Zhou, and R. H. Deng, "Dynamic secure cloud storage with provenance," in *Cryptography and Security*, Berlin, Germany, Springer-Verlag, 2012, pp. 442–464.

[35] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, pp. 612–613, 1979.

[36] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attributed-based encryption for fine-grained access control of encrypted data," in *Proc. ACMConf. Comput. Commun. Security*, 2006, pp. 89–98.

[37] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," presented at the IEEE Conf. Comput. Commun., San Diego, CA, USA, Mar. 2010.

[38] A. Pingley,W. Yu, N. Zhang, X. Fu, andW. Zhao, "CAP: A context-aware privacy protection system for location-based services," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst.*, 2009, pp. 49–57.

[39] T. Xu and Y. Cai, "Location cloaking for safety protection of ad hoc networks," in *Proc. IEEE Conf. Comput. Commun.*, 2009, pp. 1944–1952.

[40] Y. Earn, R. Alsaqour, M. Abdelhaq, and T. Abdullah, "Searchable symmetric encryption: Review and evaluation," *J. Theoret. Appl. Inf. Technol.*, vol. 30, no. 1, pp. 48–54, 2011.

[41] A. Boldyreva, "Efficient threshold signatures, multisignatures and blind signatures based on the Gap-Diffie-Hellman-group signature scheme," in *Proc. 6th Int. Workshop Theory Practice Public Key Cryptography*, 2003, pp. 31–46.

[42] The java pairing based cryptography library (jpbc)," (2013). [Online].

[43] Available: http://gas.dia.unisa.it/projects/jpbc/.

★★★