SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks 0000; **00**:1–21
DOI: 10.1002/sec

RESEARCH ARTICLE

Role-Based and Time-Bound Access and Management of EHR Data

Rui Zhang[1]*, Ling Liu[2] and Rui Xue[1]

[1]State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]College of Computing, Georgia Institute of Technology, Atlanta, GA, US

ABSTRACT

Security and privacy are widely recognized as important requirements for access and management of Electronic Health Record (EHR) data. In this paper we argue that EHR data needs to be managed with customizable access control in both spatial and temporal dimensions. We present a role-based and time-bound access control model (RBTBAC) that provides more flexibility in both roles (spatial capability) and time (temporal capability) dimensions to control the access of sensitive data. Through algorithmic combination of role-based access control and time-bound key management, our RBTBAC model has two salient features. First, we have developed a privacy-aware and dynamic key structure for role-based privacy aware access and management of EHR data, focusing on the consistency of access authorization (including data and time interval) with the activated role of user. In addition to role-based access, a path-invisible EHR structure is built for preserving privacy of patients. Second, we have employed a time tree method for generating time granule values, offering fine granularity of time-bound access authorization and control. Our initial experimental results show that tree-like time structure can improve the performance of the key management scheme significantly and RBTBAC model is more suitable than existing solutions for EHR data management since it offers high-efficiency and better security and privacy.

KEYWORDS

EHR system; privacy preserving; role-based access control; time-bound key management; time tree

***Correspondence**

State Key Laboratory of Information Security, Institute of Information Engineering Chinese Academy of Sciences, Beijing, China.
E-mail: zhangrui@iie.ac.cn

Received . . .

1. INTRODUCTION

Electronic Health Record (EHR) is a digital record shared across different healthcare settings, by network-connected enterprise-wide information systems called EHR systems [1]. On one hand, EHR systems hold the promise to provide fast and on-demand access to medical documents and help reduce medical errors and enhance healthcare quality by providing healthcare workers with decision support. On the other hand, this openness, while being an essential feature of EHR, exposes patients to the risks of privacy disclosure by improper authorization, misuse and abuse of EHR data. Therefore, security and privacy are widely recognized as important requirements for access and management of EHR data. Although regulations such as HIPAA [2] and the HITECH Act [3] have been put forward to rule the useage of EHR, but specific mechanisms are not described.

1.1. Security and Privacy Requirements for EHR Access

We identify three main requirements for making access EHR data secure and privacy preserving.

First, when a patient is offered medical treatment, he expects that his medical records can only be accessed by authorized doctors, and other unauthorized doctors should not be able to read any part of his medical records. For example, the EHR data of a patient must be accessed by authorized physicians or healthcare professionals when they have the matching roles or higher levels of roles in a role hierarchy as shown in Fig.1b. Furthermore, in role-based EHR data structure as shown in Fig.1a, the node authorized to physicians should be accessible by $Physician$ or nodes in lower levels of $Physician$. Fig.1a gives an example of role-based hierarchical data structure for EHR data and Fig.1b shows an example of role

Prepared using secauth.cls [Version: 2010/06/28 v2.00]

hierarchy (the detailed explanation of Fig.1 will be given in Section 3.2). However, some special circumstances may need more fine-grained access control. For instance, a hematologist may need to access all blood test results of a patient, which are nested under different role nodes (see Fig.1a). Therefore, in an EHR system, we also need customizability to effectively achieve flexible, fine-grained, role-based access control over large number of EHR data.

Second, in the healthcare setting, a physician is allowed to access the medical data of a specific patient only during the time period of offering healthcare treatment. For example, emergency room doctor should not be able to access any medical document of a patient, once he or she has completed the emergency treatment of the patient and the patient has left the emergency room. In addition, given a patient, different doctors involved should have different time intervals in terms of accessibility to this patient's medical data. Furthermore, a doctor may need to access the medical data of different patients at the same time period. Thus, we need to support time-bound access and to manage the accessibility of EHR data from time dimension in healthcare domain.

Finally, it is to the best interest of some patients that their doctor only know the medical data that are relevant to the diseases currently under treatment by the doctor (such as dental disease) but should not be able to access other medical data (such as psychotherapy data). Therefore, in addition to EHR data, the role-based structure and indices of EHR data should not be released to users, since the contents of internal nodes in such structure may contain sensitive information that can be used by healthcare professionals to infer other diseases of the patient. Moreover, the structure of EHR data may change frequently due to the involvement of different sets of healthcare providers for different medical conditions and treatments. Thus, the third challenge in securing EHR access is to make the structure of EHR data and access path invisible to users who are unauthorized or only authorized to access a partial component of the EHR data. Furthermore, for healthcare settings with untrusted DB, the indices of EHR data should be invisible too.

In the common setting of access control model, encryptor decrypts the data retrieved from database before they are sent to the requester. The secrecy of the transmitted data is guaranteed by secure transport-level protocols (such as SSL/TLS) over network. This approach requires the data need to be decrypted then re-encrypted before it is transported in the public network. Moreover, recipients of EHRs obtain the cleartext records and usually cache them unprotected on the end device, leading to leakage and insecurity of the data. Thus we consider a more secure way that encryptor sends the encrypted data directly. Then data can be decrypted through a client installed software if the user is legally authorized.

A basic and straightforward approach to deal with above problems is to encrypt the medical records and update encryption and decryption keys periodically. Obviously, this approach is known to increase the cost of the EHR system, and cannot achieve fine-grained access control. Furthermore, periodic distribution of decryption keys to every user of EHR system is unpractical and insecure.

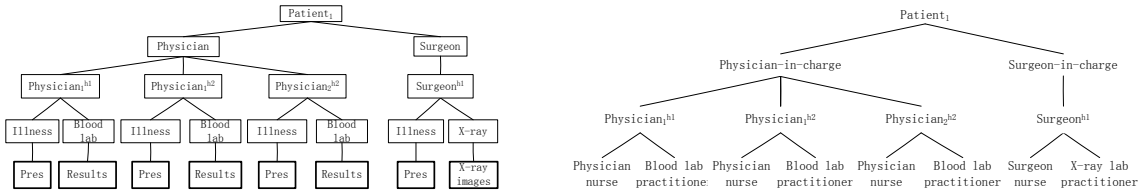
1.2. Our Contributions

In this paper, we propose a general purpose *role-based and time-bound access control (RBTBAC)* model for EHR systems. The development of RBTBAC model is based on algorithmic combination of *role-based* access control and *time-bound* hierarchical key management such that a legitimate user of EHR system is authorized a time interval to access EHR data based on his/her role. The RBTBAC model offers more flexibility of both roles (spatial capability) and temporal capability to control the access of sensitive data. Concretely, we have developed a privacy-aware and dynamic key structure for role-based privacy aware access and management of EHR data, focusing on the consistency of access authorization (including data and time interval) with the activated role of user. In addition we have employed a time tree method for generating time granule values, offering fine granularity of time-bound access authorization and control. Through extensive experimentation, we demonstrate that our RBTBAC model not only offers better security and privacy for access EHR data, but also provides high efficiency and customizability.

RBTBAC model is capable of accessing and managing EHR data, because it provides both structure and time constraint when a user accesses EHR data with an activated role. Compared with most existing access control models, RBTBAC model has three characteristics: *role-based*, *time-bound* and *structure-invisible access control hierarchy*.

Roles and role based access are the first characteristics of the RBTBAC model. Given a care delivery organization, roles are created for various healthcare functions. The permission to perform certain healthcare operations is assigned to specific roles. Similarly, users of EHR system are assigned permissions through their particular roles. However, a user can have multiple roles at the same time. For instance, a doctor is a physician-in-charge for one patient, and in the mean time he is also a temporary physician for another patient. In this paper, the concept of role-based EHR management is composed of *role-based hierarchical data structure*, *role hierarchy* and *role-based access control*. These three role-based concepts make up a *role-based access control hierarchy*. Specifically, if a user is granted the access to a node in higher level of a role-based hierarchical data structure, then he can access all leaf nodes of the authorized node. For example, if a user has a role as $Physician_1^{h_1}$ in Fig.1b, then he is authorized to access data under the role $Physician_1^{h_1}$ and the first two leaf nodes $Pres$ (here pres is the abbreviation for

Security Comm. Networks 0000; **00**:1–21 © 0000 John Wiley & Sons, Ltd.
DOI: 10.1002/sec
Prepared using secauth.cls



(a) Role-based hierarchical structure of medical data

(b) Role hierarchy

Figure 1. EHR data hierarchy and role hierarchy

prescription) and $Results$ in Fig.1a, where $Physician_1^{h_1}$ refers to the physician$_1$ from hospital h_1. If a user has a role as *Physician-in-charge* as shown in Fig.1b, then he can be authorized to access data in node *Physician* in the second level in Fig.1a, which means that he can access all EMR (Electronic Medical Record) data from all three physicians of $Patient_1$. Note EMR is usually a component of EHR [4].

The second characteristic of RBTBAC is time-bound access. The roles in healthcare system have temporal constraint by time bound such that each healthcare delivery is only valid for a period of time. Therefore, in EHR systems, time-bound access means that each user will be granted a valid time period based on his/her role, and the user is only allowed to read EHR data of authorized nodes during the given time period. Once the valid time period has expired, the user should no longer be able to access the data.

The third characteristic of RBTBAC is to support a structure-invisible access hierarchy in the RBTBAC model for EHR data. The basic idea of structure-invisibility is that the structure and indices of EHR data should not be known to the users and the untrusted DB. In addition, the paths from the granted node to all the leaf nodes should also be invisible to the user if this user is only authorized the access right to read the data of leaf nodes. We will have separate subsections to discuss this issue (see Section 4.1.3 and Section 6.1.5).

Technically, we achieve RBTBAC from following three dimensions respectively: role-based, time-bound and path-invisible access structure.

In role-based dimension, we adopt a role-based access control hierarchy to assign access nodes for users (doctors). Typically, we use an access credential to bind one healthcare treatment. That is, for a doctor, one access credential binds one specific role and a set of EHR data of a patient. Doctors can access the set of EHR data by providing an access credential with a specific role. Furthermore, the process of authorization is under the control of predetermined access control policy.

In time-bound dimension, we make use of hierarchical time-bound key management scheme as basic key generation and distribution scheme. With this approach, each user is granted a consecutive time interval based on his role, in which he can generate decryption keys by himself. On the contrary, user cannot create any decryption key beyond the granted time interval even he is authorized to access the data before. Then, we use hierarchical key structure to manage large number of keys, so that users can produce multiple temporary decryption keys to access different parts of EHR data with only one long-term key. Furthermore, we develop a practical time-bond key management scheme that maps time granules into a *time tree* to provide more efficient computation on time parameters than most existing key management schemes.

In path-invisible access structure dimension, we encode all index nodes of EHR data and build an invisible path for users to access authorized data, so that all users of EHR system even DB are unaware of EHR data structure. Therefore, it provides higher level of privacy protection for patient, and supports various EHR structures and dynamical updates of EHR data structure.

The rest of this article is organized as follows. In Section 2, we describe current access control models and time-bound key management approaches, and their limitations. In Section 3, we give an overview of the RBTBAC model. In Section 4, we present the enforcement of RBTBAC model including a role-based key structure and an efficient tree-like time structure. We present the detailed RBTBAC protocol in Section 5. Then we discuss security and privacy of our RBTBAC model in Section 6. In Section 7, we analyze and compare time and space complexity of our RBTBAC scheme with existing schemes. Next, we discuss the related works in Section 8. Finally, we conclude with notes on future work and other applications of our proposed technique in Section 9.

2. BACKGROUND AND LIMITATIONS OF EXISTING APPROACHES

President Barack Obama gave a speech at annual conference of the American Medical Association in 2009. The key words are: All that (medical) information should be stored securely in a private medical record so that your information can be tracked from one doctor to another-even if you change jobs, even if you move, even if you have to see a number of different specialists. Although the words seem so simple, the ability for clinical institutions to actually accomplish this is incredibly difficult. The term

stored securely involves several secure issues to be solved: data encryption, secure storage, strong authentication, access control and key management. Also, it contains issues of actual implementation-searchability, feasibility and efficiency.

In the healthcare scenario, when multiple users sign in an EHR system to query records for a specific patient, the access control component verifies the authorization of each user first. Then the records are retrieved through corresponding index information and sent to the user based on different privilege.

In this section, we provide a brief overview of access control and time-bound key management and discuss the limitations of existing works.

2.1. Access Control

In the EHR system, the database storing the composite EHR is a new paradigm of database outsourcing, called database-as-a-service (DAS) [5], where an organization's database is stored at an external service provider. In such systems, access control is a very important issue, especially if the data owner wishes to publish her data for external use, e.g. patient's EHR data are used for group consultation. Access control is in charge of guaranteeing that each user obtains the correct information and protecting the sensitive data from revealing to unauthorized users by identification and authentication. In the healthcare scenario, it requires more stringent security and privacy, that is, enforcing rigid mandatory access control on encrypted EHR data.

Though much work on access control (see 8.1) has achieved some aims, we find the work on access of encrypted EHR data is very little. There are several reasons for this situation. First, the types of medical records are more than file system (such as X-ray image, pharmacy, prescription, etc.), and the properties of medical records are more complex than file system (such as sensitivity), so that it needs more fine-grained access control policy. Second, since different patient's EHR data are collected from different EHR providers and accessed by any authorized user (under normal circumstances, the accessor is not the creator), the selection of encryption scheme and key generation algorithm are thorny problems. Third, key management and distribution for different users is also a hot potato. For instance, how to distribute keys to every user so that each of them can decrypt the data authorized by patient, while cannot decrypt any other data containing sensitive information or not allowed by patient is very hard.

In our scheme, based on the role of users, each of them is granted different permission to access encrypted EHR data within a specific time interval. The user cannot access those data that the patient does not allow to reveal to him in the inconvenient time. Compared with existing diverse access control schemes, our scheme combines role-based access control with time-bound key management technique, so that it is capable to enhance the privacy for patients.

2.2. Time-Bound Key Management

Key management is another important issue which is closely related to encryption and access control. Key management schemes are used to provide access control to data streams for legitimate users. Recall that, in an EHR system, each piece of record should be encrypted stored in database and accessed by access control policies. Every time the use of encryption algorithm, there will be a corresponding encryption and decryption key pair. Therefore, how to manage these multiple keys is a big issue in EHR systems.

For our motivated scenario, we employ time-bound hierarchical key management scheme to restrict user's access on EHR data. Literally, this scheme has two properties: hierarchy and time-bound. The hierarchy is a widely used structure for key management [6, 7, 8, 9, 10, 11, 12], which is used to assign cryptographic keys to a set of partially ordered classes so that the user in a higher class can derive the cryptographic key for users in a lower class. Time-bound means each cryptographic key is bound to current time period by adding time parameters in the process of key generation. In the healthcare scenario, users are assigned to access EHR data for only a certain period of time. Once the time period has expired, user should not be able to access any record with key if he is not authorized to do so. Thus, encrypting EHR data when send it to each user with time-bound key is a good way to tackle above problems.

In the recent decade for development of time-bound key management technique, a typical work is proposed by Elisa Bertino et al. [11]. Their scheme is based on elliptic curve cryptography and claimed secure against X. Yi's attack [13]. In Bertino'scheme, the encryption key $K_{i,t}$ for class of node C_i at time granule $t \in [0, z]$ is computed by the formula below:

$$K_{i,t} = H_K\left((K_i)_Y \oplus H^t(a) \oplus H^{z-t}(b) \oplus CID_i\right) \tag{1}$$

Where K_i is the class key of class of node C_i, $(K_i)_Y$ is the Y-coordinate of K_i, $H^t(c)$ is the t-fold iteration of hash function $H(\cdot)$ applied to c, CID_i is the identity of C_i and $\oplus$ is the bitwise XOR.

When a user is given a tamper-resistant device storing H_K, $H^{t_b}(a)$, $H^{z-t_e}(b)$, CID_i and other parameters, he can derive decryption key $K_{i,t}$ in time granule $t \in [t_b, t_e]$ with Equation (1) in tamper-resistant device, where $H^t(a) = H^{t-t_b}(H^{t_b}(a))$, $H^{z-t}(b) = H^{t_e-t}(H^{z-t_e}(b))$.

However, the security of their scheme is questioned by Sui [14]. Moreover, the use of tamper-resistant device is infeasible in EHR system.

In summary, as applied to the EHR system, time-bound hierarchical key management scheme should be rebuilt to

Security Comm. Networks 0000; **00**:1–21 © 0000 John Wiley & Sons, Ltd.
DOI: 10.1002/sec
Prepared using secauth.cls

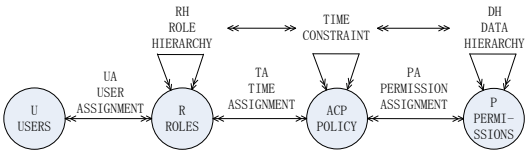


Figure 2. RBTBAC model

provide better security, privacy and practicality, so that it has capability of satisfying the special requirements of EHR systems.

3. THE RBTBAC MODEL FOR EHR SYSTEMS

This section discusses related access control model for EHR infrastructures. We begin with presenting an RBTBAC model used in role and data hierarchy systems. Then we give the reference security model for an EHR system.

3.1. RBTBAC Model

Role-based access control (RBAC) [15] [16] is designed to simplify security administration by introducing the 'role' abstraction between principles (subjects) and privileges (objects). This splits management of the principal to privilege mapping by splitting it into two parts: a mapping from user to roles, and a mapping from roles to privileges. In [17], four RBAC schemes are described. Among them, flat RBAC is the basic model which simply dictates two relationships: user-role and role-privilege. Hierarchical RBAC extends the basic flat RBAC model by adding role hierarchy. Role hierarchy has an associated constraint that must be satisfied if a user is to activate the target role based on their already being active in the source role (the original role used for creating medical data hierarchy).

Hierarchical RBAC can be used to build our RBTBAC model since it facilitates the development of powerful policy schemes in EHR system. We extend the basic hierarchical RBAC model to RBTBAC model as shown in Fig.2. An additional mapping from privilege to access control policy (ACP) with time constraint is added between roles and permissions, that is, each access permit for a set of data is bounded by a time interval. The double-headed arrow in Fig.2 indicates the many-to-many relationship between the two objects. One user can activate several different roles, where each role is corresponding to a valid access time interval of specific data sets. Similarly, one permit is bounded by number of rules, and one rule can affect more than one permit.

Compared with hierarchical RBAC, RBTBAC has time parameters in each access control policy. When TA (trusted authority) concludes a result by searching access control policies, it actually not only gives the access permission to a specific role but also grants a time period for the authorized access.

3.2. EHR Data Structure and Role Hierarchy

Usually, EHR data are hierarchically clustered. In the EHR environment, hierarchical structure of EHR data is required to be flexible and dynamic. Theoretically, there are two types of hierarchical structure of medical records: attribute-based structure and role-based structure. Fig.1a is an example of role-based hierarchical structure of medical records [4].

As a general rule, patient visits a special doctor for a specific illness. Therefore, multiple records can be clustered by the different roles of doctors. We construct an EHR tree for each patient using $Patient_i$ as the root of EHR tree. For the same patient, say Alice, the same token is assumed. We can set the initial role based hierarchical structure of an EHR in terms of hierarchical template as shown in Fig.1a. The root of the tree is at the top level, say Level 0. Level 1 is the role nodes of doctors, and their children nodes are labeled with unique identity of doctors within each corresponding CDO (Care Delivery Organization), such as $hospital_1$ and $hospital_2$ in Fig.1a (Level 2). The nodes in and below Level 3 are medical diagnosis nodes and other correlative inspection nodes. Finally, only the leaf nodes contain EHR records, such as prescriptions and diagnosis and so on. Thus the tree has two types of nodes: leaf nodes that contain real EHR data and other nodes of upper levels that are actually indices for EHR data. Thereby the former are called *data nodes* and the latter are called *index nodes*. For easy retrieval, we want to sort all child nodes by alphabetical ordering of their tokens or node IDs from left to right except the diagnosis nodes in Level 3, where we place diagnosis nodes as the leftmost child node of their parent node, since they are more important than other sibling nodes. Obviously in this structure, all data nodes are nested according to a role node, so that they can be expediently retrieved by different roles of doctors.

Role hierarchy is a natural mean for structuring roles to reflect an organization's lines of authority and responsibility, as well as in medical organizations. An example of role hierarchy corresponding to role-based hierarchical EHR data is illustrated in Fig.1b. Mathematically, the role hierarchy is partial order, which is reflexive, transitive and anti-symmetric relation. By convention more powerful (or senior) roles are shown toward the top of role-hierarchy diagram, and less powerful (or junior) roles toward the bottom. Fig.1b shows that senior roles aggregate the access permission of junior roles. Thus $Patient_1$ can get access permission of his/her own medical records in Fig.1a. Similarly, role *Physician-in-charge* acquires the permissions of $Physician_1^{h_1}$, $Physician_1^{h_2}$ and $Physician_2^{h_2}$, and may have additional permission of its own to access physician records of $Patient_1$. This tree-structure hierarchy is good for aggregation but does not support sharing. In Fig.1b, there

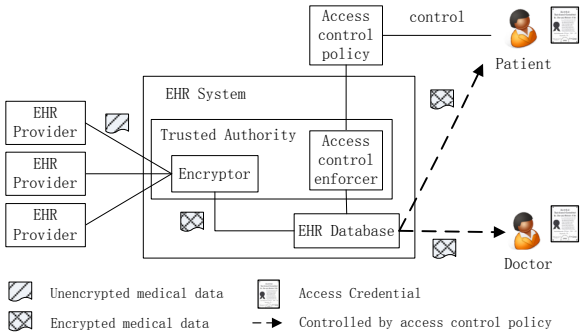


Figure 3. The reference security model for EHR system

can be no sharing of medical data between the *Physician-in-charge* roles on the left and *Surgeon-in-charge* roles on the right.

3.3. Reference Security Model for EHR System

The reference security model mainly has three entities cooperatively to manage and control the usage of EHR data in security as shown in Fig.3. The most important one is the TA (trusted authority) who retains two roles: encryptor, who is responsible for encrypting various EHR data from different healthcare providers into ciphertexts and access control enforcer, who enforces predefined access control policies. The remote EHR database is a necessary component to store the encoded composite EHR data. Additionally, an access control policy engine is the collector and developer for access control policies under patient's control.

The original EHR data of a patient are collected periodically from repositories of different healthcare providers. Then encryptor encrypts the various medical data and uploads the encrypted data to the remote EHR database. To request access to EHR data of a patient, user is required two types of credentials: *system identity credential* and *access credential*. When a new user signs up for EHR system, TA returns a system identity credential to the user. The system identity credential includes a unique identity, the original role of the user (which is used to sign into the EHR system), a user's master key (which is used to generate short-term access keys for subsequent accesses) and some system parameters. With system identity credential, user can only log into the EHR system, but cannot access any EHR data even his own EHR data. To access EHR data of a specific patient, user sends an access request to TA with his system identity credential. The request should mainly contain the patient's ID, a target role for accessing the requested data (which is lower or equal to his/her original role in role hierarchy), the keywords for requested data and an expected access time period. TA replies permission result based on the predefined access control policies. Once the request is consistent to access control policies, TA authorizes an access credential to the user. An access credential should include access time period, parameters used to compute decryption keys of authorized data and verification information. User can legally query EHR data from remote database with corresponding access credential and decrypt the encrypted data in the granted time period.

Here the user is granted access privilege for a set of data in a specific time period by an access credential. When there is an access request, database only needs to verify the system identity credential and access credential rather than interacting with TA and access control policy repository to verify the authorization.

3.4. Patient-Controlled RBTBAC Policy

For protecting patient's privacy, patient should be involved in the development process of access control policy to decide which parts of his/her EHR data can be shared with whom. Here we first give a formal definition of RBTBAC policy, and then illustrate how the policy works under patient's control through some examples.

Definition 3.1 (RBTBAC Policy)
An *RBTBAC policy* is a tuple $acp = (ro, \langle Patient_j, \{C\}\rangle, \langle t_b, t_e, ti\rangle, pu, re)$, where $ro \in Ro$ is the target role of the access requester and Ro is the set of roles in role hierarchy; $\langle Patient_j, \{C\}\rangle$ is a set of EHR data nodes of patient j, which can be selected as target objects; $\langle t_b, t_e, ti\rangle$ is a set of time parameters for the user accessing requested data: t_b is the start time of the requested time interval, t_e is the end time of the requested time interval, and $ti \in TI$ is the longest time interval for valid access on selected data; $pu \in \{treatment, research, payment, default\}$ indicates the purposes of user access; $re \in \{permit, deny\}$ is the authorization result of current request.

Example 3.2
Using Definition 3.1, Fig.1b as role hierarchy and Fig.1a as corresponding EHR data structure, the following access control policies can be articulated:
acp1: $(Patient_1, \langle Patient_1, Patient_1\rangle, default, default, permit)$;
acp2: $(surgeon^{h_1}, \langle Patient_1, surgeon^{h_1}\rangle, \langle t_b, t_e, twoweeks\rangle, treatment, permit)$;
acp3: (*physician-in-charge*, $\langle Patient_1, surgeon^{h_1}\rangle, \langle t_b, 12:00am, oneday\rangle, treatment, deny$).

In acp1, the patient is allowed to read all his/her own EHR data at any time. Inferred from acp 2, a user with role $surgeon^{h_1}$ is allowed to read the EHR data nested under the node $Surgeon^{h_1}$ of $Patient_1$ in Fig.1a when he offers healthcare treatment for $Patient_1$, which starts at t_b and ends at t_e. The longest valid time interval for access those EHR data is two weeks. While in acp 3, the request that user with role *physician-in-charge* accesses EHR data under the node $Surgeon^{h_1}$ in the day will be denied.

Note that, in most cases, time interval from t_b to t_e (namely, the value of $t_e - t_b + 1$) depends on the time interval of offering healthcare treatment and is not equal to ti. If $(t_e - t_b + 1) < ti$, then the valid access time interval

Security Comm. Networks 0000; **00**:1–21 © 0000 John Wiley & Sons, Ltd.
DOI: 10.1002/sec
Prepared using secauth.cls

should subject to t_e; otherwise, the access time interval should end at $t_b + ti$. In the latter case, the user should re-request access permission for the rest time interval. Using acp 3 as an example, no matter when the access right is authorized to user, after the midnight of the day the user can not access the data anymore. Here we need to point out that once EHR data have been created, they should be read-only.

4. ENFORCEMENT OF RBTBAC MODEL

So far we have discussed our RBTBAC in model terms. In this section, we give technical details for enforcing our RBTBAC model.

In an RBTBAC model, the key technical issue is generation of access key, so that users can access different pieces of EHR data in different time intervals. In this paper, the key generation and distribution scheme is called *RBTB key management scheme*. Since the algorithm used to encrypt and decrypt EHR data is symmetric cryptographic algorithm, decryption key is identical to encryption key, namely access key mentioned in Section 5.2. The algorithm for generating an access key $K_{k,t}$ is parameterized with two numbers: a long-term class (node) key K_k for class node C_k and a time granule value $V_{B(t)}$. This section focuses on computation of the two primary parameters. We first discuss the role-based key structures of RBTB key management scheme, and then put forward an efficient method, time tree, for computing time granule values.

4.1. Role-Based Key Structures

As mentioned before, a user may have several different roles, so he may need to have multiple keys to access different pieces of EHR data. The ideal situation would be that a user can decrypt multiple parts of EHR data with the smallest number of keys. To achieve this goal, we let each user efficiently produce multiple access keys by themselves with only one long-term master key. Technically, we first construct a static hierarchical encryption key structure, and then dynamically establish a hierarchical decryption key structure.

4.1.1. Encryption and Decryption Key Structures

Role-based key structure provides a way of integrating access control in the key distribution phase in a manner that facilitates generating access keys for different data nodes.

Definition 4.1 (Security Class)
A security class is a set of EHR data that share common searchability access privileges. That is, all EHR data contained within a security class can be accessed and searched by the same authorized user.

Suppose that there are totally n doctors and m patients in an EHR system. Consider key tree as Fig.4a, the partially ordered set $\langle C, \preceq \rangle$, where the vertices are $C = \{C_0, C_{P_1}, \cdots, C_{P_m}, C_1, C_2, \cdots, C_N\}$, where each vertex is a security class and diagramed with a node in the hierarchical tree. If $C_i \preceq C_j$, we say security class C_i is subordinate to security class C_j (or security class C_j is superordinate to security class C_i). $C_i \prec C_j$ means that $C_i \preceq C_j$ and $C_i \neq C_j$. If $C_i \prec C_j$ and there is no C_x such that $C_i \prec C_x \prec C_j$, we say C_i is immediate subordinate to C_j (or C_j is immediate superordinate to C_i), which denoted by $C_i \prec_d C_j$ and labeled by a directed edge between the two security classes in the tree.

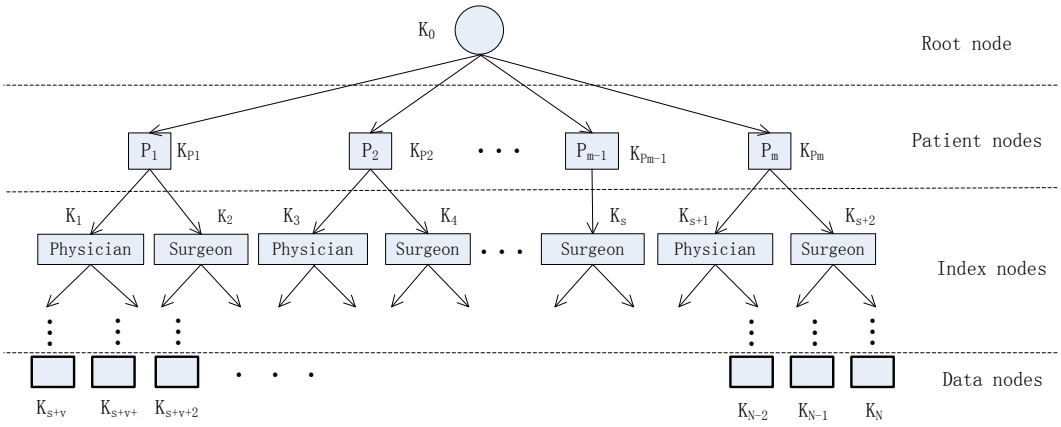
To encrypt EHR data in each security class, encryptor first establishes encryption key tree as shown in Fig.4a. Without loss of generality, let C_0 be the root of the tree. Correspondingly, K_0 is the *system root key* used to encrypt the data in class C_0. The nodes in the second level are called *patient nodes* which is a token of the patient. All the EHR data of the patient are nested under the matching patient node. Therefore the corresponding key is called *patient master key* $K_{P_j} (j = 1, 2, \cdots, m)$, which can be distributed to patient j, so that the patient can derive keys of lower level when given access right. For example, the key K_{P_1} is related to node P_1, which is also the root node in Fig.1a. That means the subtree rooted with $K_{P_j} (1 \leq j \leq m)$ is isomorphic to the tree of EHR data of patient j. Similarly, the keys in lower levels are encryption keys used to encrypt data in homologous classes in the tree of EHR data (see Fig.1a). Note again, in this key tree, only the keys in leaf nodes are used to encrypt real EHR data. In summary, the keys in lower level classes can be derived from the keys in higher level classes, whenever they are partial ordered, which means there exists a path between these two security classes in the tree.

The decryption key structure is slightly different than the encryption key structure. Consider the situation where Fig.4a is used as the decryption key structure: a doctor is required to have more than one decryption key to access EHR data of different patients. To avoid the inconvenience and added security risks that come with the distribution of many keys, we dynamically add *doctor nodes* between root node and patient nodes as shown in Fig.4b. The key related to a doctor node is called *doctor master key* $K_{D_i} (i = 1, 2, \cdots, n)$. It is bound with a doctor's ID and assigned to the corresponding doctor. Therefore, doctor can use the unique master key to produce multiple keys to access EHR data of different patients. Here the 'dynamically' means the relationship between doctor node and patient node is not static. According to the description of Section 3.3, it is established only when a doctor is issued an access credential. Moreover, it can be deleted once the valid access time interval has expired.

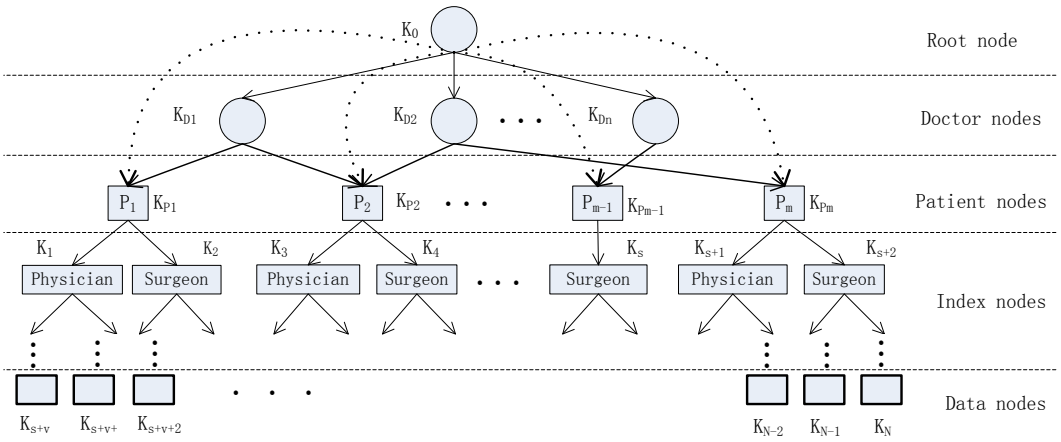
4.1.2. Computation of Class Key

The class keys are relative static and have two roles: encrypting index nodes and computing access (encryption/decryption) keys for data nodes.

Let $\mathbb{G}$ be a cyclic group of prime order q (with $\|q\| = n'$, where n' is the security parameter), and let g be a



(a) Encryption key structure



(b) Decryption key structure

Figure 4. Hierarchical key structure for encrypting and decrypting EHR data

generator of $\mathbb{G}$. $H' : \{0,1\}^* \to \mathbb{G}$ is a secret collision-resistant hash function. IV is a random initial value. P_j, D_i and CID_x represent identities of patient j, doctor i and security class C_x respectively. Relationship value R_{C_x,C_y} between two partial ordered classes $C_y \prec C_x$ can be calculated by Equation (2).

$$R_{C_x,C_y} = g^{[H'(CID_y)-H'(CID_x)] \bmod q} \tag{2}$$

Then class key K_y for security class C_y can be computed with Equation (3).

$$\begin{aligned} K_y &= g^{H'(CID_y)} = K_x \cdot R_{C_x,C_y} \\ &= g^{H'(CID_x)} \cdot g^{H'(CID_y)-H'(CID_x)} \end{aligned} \tag{3}$$

Concretely, for encryption key structure (Fig. 4a), encryptor computes class keys for each type of security classes as follows:

- Root node (C_0): $K_0 = g^{H'(IV)}$
- Patient node (C_{P_j}): $K_{P_j} = g^{H'(P_j)} = K_0 \cdot R_{C_0,C_{P_j}} = g^{H'(IV)} \cdot g^{H'(P_j)-H'(IV)}$;
- Index node (C_x): $K_x = g^{H'(CID_x)} = K_{P_j} \cdot R_{C_{P_j},C_x} = g^{H'(P_j)} \cdot g^{H'(CID_x)-H'(P_j)}$;
- Data node (C_k): $K_k = g^{H'(CID_k)} = K_x \cdot R_{C_x,C_k} = g^{H'(CID_x)} \cdot g^{H'(CID_k)-H'(CID_x)}$.

For decryption key structure (Fig.4b), a doctor node is added between root node and patient node when a docotor is issued an access credential. The doctor master key is computed with $K_{D_i} = g^{H'(D_i)} = K_0 \cdot R_{C_0,C_{D_i}}$ and distributed to user D_i. When user D_i queries EHR data of security class C_k of patient j, the relationship value $R_{C_{D_i},C_k}$ is computed and sent to user D_i as well as other parameters through an access credential. Consequently, user D_i can derive class key K_k with his master key K_{D_i} and given value $R_{C_{D_i},C_k}$ by himself through Equation (4).

$$K_k = K_{D_i} \cdot R_{C_{D_i},C_k} \tag{4}$$

Correctness. Suppose partial order $C_k \prec_d C_x \prec_d C_{P_j} \prec_d C_{D_i} \prec_d C_0$ in the decryption key structure, user D_i can correctly compute class key K_k for security class

C_k by given $R_{C_{D_i},C_k}$ and K_{D_i}, because the following equation holds.

$$K_k = K_x \cdot R_{C_x,C_k} = K_{P_j} \cdot R_{C_{P_j},C_x} \cdot R_{C_x,C_k}$$
$$= K_{D_i} \cdot R_{C_{D_i},C_{P_j}} \cdot R_{C_{P_j},C_x} \cdot R_{C_x,C_k} = K_{D_i} \cdot R_{C_{D_i},C_k}$$

Security. Based on the condition that H' is only known to encryptor, the security of class key generation is based on the *discrete logarithm problem* in a cyclic group $\mathbb{G}$ of prime order q with given generator g [18]. That is, user D_i is unable to compute $\log_g K_{D_i}$ given a random element $K_{D_i} \in \mathbb{G}$. The randomness of K_{D_i} is ensured by the secret collision-resistant hash function H' and the randomness of its inputs. Similarly, for other class keys and relationship values, we have the same security conclusions. In addition, each of user's master keys should be secretly kept by its owner and other class keys for index nodes are securely stored by encryptor. Once a class key is exposed, it should be revoked and recreated with new identity (except the class keys of data nodes: they can be computed by users through authorization).

4.1.3. Path Invisible Access Structure

In the EHR system, both EHR data and the indices to that data can reveal part of the patient's sensitive information. For this reason, we highly suggest the indices of EHR data, namely the index nodes of EHR data structure should be encrypted too. In our RBTBAC scheme, they are encrypted by corresponding class keys. Thus, users can not obtain any other information about the patient except the EHR data authorized to him, that is, an invisible access path is build for users.

Specifically, when a user is authorized to access EHR data of security class C_k, he is only given the relationship vlaue $R_{C_{D_i},C_k}$ through the access credential. Then the user computes K_k with Equation (4) without any intermediate index node involved in the process of calculation. It can be inferred from above steps that user is unaware of any index node or of the structure of EHR data. Therefore, we call the structure with above character *path invisible access structure*.

4.2. Time Tree

In this subsection, we introduce a novel method, *time tree*, to efficiently compute time granule values.

4.2.1. Definition of Time Tree

We divide time into small granules, which are numbered as $0, 1, 2, \cdots, z$, and map these time granules into a time tree. The time tree can be binary tree or multi-tree according to the actual situation.

First, we distinctly state the concepts of different time periods that used in our RBTBAC scheme. The whole time period is called *timeline*, which is the longest time unit involved in this paper. The timeline is divided into uniform small pieces. Each small time slot is called *time granule*, which is the smallest unit of time period. We call several consecutive time granules *time interval* which represents a valid time period for user accessing a set of EHR data. For example, the timeline is one year and the small time granule is one day, that means a whole year is divided into 365 days. A user can be granted any consecutive days in 365 days as time interval such as one week or one month to access EHR data.

Next, we need to define a new concept, *time tree*, which is introduced to effectively compute time granule values in our RBTBAC scheme.

Definition 4.2 (Time Tree)
A *time tree* is a finite set of time interval values, which begins at a value of whole timeline as root node. Child nodes divide time period of parent node into several smaller consecutive disjoint time intervals. Each leaf node denotes the smallest time granule value. The value of node can be computed through the path from root node to itself.

Definition 4.3 (Time Binary Tree)
If the time tree has at most two children for each node, we call it *time binary tree*.

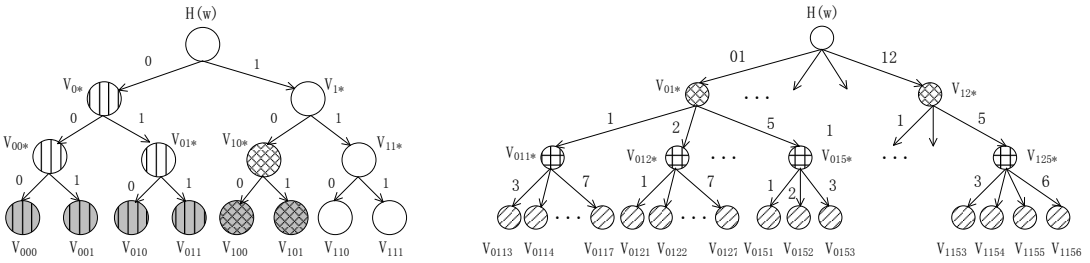
Definition 4.4 (Time Multi-Tree)
If the time tree has more than two children for each node, we call it *time multi-tree*.

4.2.2. Two Structures of Time Tree

Now we explain how to map time granules into a *Complete Binary Tree (CBT)* through an example. Using Fig.5a as an example, the timeline is divided into eight small time granules with the binary representation of $000, 001, \cdots, 111$. For simplicity, let $B(t)$ denote the binary expression of time granule and $V_{B(t)}$ indicate the value of time granule t. A star in subscript indicates intermediate nodes (that is not the root and leaf nodes) of the time tree. Obviously, the values of smallest time granules are labeled by leaf nodes of the CBT. Besides, the value of each node can be calculated through the path from the root node to itself, where the value of root node is $H(w)$, H is a one-way hash function and w is a randomly selected integer. Consequently we have following equations, where $\|$ denotes string concatenation.

$$V_{0*} = H(H(w)\|0), V_{1*} = H(H(w)\|1),$$
$$V_{00*} = H(H(H(w)\|0)\|0) = H(V_{0*}\|0), \cdots$$
$$V_{111} = H(H(H(H(w)\|1)\|1)\|1) = H(H(V_{1*}\|1)\|1)$$
$$= H(V_{11*}\|1)$$

We observe that all time interval $[t_b, t_e] \in [0, z]$ can be composed of a number of *Full Binary Subtrees (FBSs)*, that is $[V_{B(t_b)}, V_{B(t_e)}] = FBS[V^1] \cup FBS[V^2] \cup \cdots \cup FBS[V^u]$, where $FBS[V^u]$ denotes an FBS whose value of root node is V^u and symbol $\cup$ means the values of leaf nodes of FBSs (the right side of the equation) compose the values of time interval from t_b to t_e. For example, time interval $[t_b, t_e] = [0, 5] = [000, 101]$, which is labeled in



(a) Time binary tree **(b)** Time multi-tree

Figure 5. Two samples of time tree

gray in Fig.5a, contains two FBSs, one is rooted by node V_{0*}, the other is rooted by node V_{10*}. Therefore, we have following equation: $[V_{000}, V_{101}] = FBS[V_{0*}] \cup FBS[V_{10*}]$.

With the tree structure, any leaf node of an FBS can be computed through the value of its root node. For example, $V_{000}, V_{001}, V_{010}, V_{011}$ can be computed with the value V_{0*} respectively, V_{100} and V_{101} can be computed with the value V_{10*}. Consequently, time granule values of time interval $[t_b, t_e] = [000, 101]$ can be computed with the given values V_{0*} and V_{10*}.

In healthcare domain, time interval of treatment is usually counted by days, weeks or months. If we define the smallest time granularity as daily, it is not very convenient to find right intermediate nodes to compute each time granule when a doctor is authorized access time interval weekly or monthly. For above reason, we consider another time structure called time multi-tree as shown in Fig.5b. The root node denotes the timeline of one year. The nodes in the second level are the values of months expressed by V_{month*}. For instance, V_{01*} means the value of January and so forth. The third level is divided by weeks, the expression of time interval values of this level is $V_{month\|week*}$, such as V_{011*} expresses the time interval value of the first week of January. The leaf nodes denote the smallest time granule such as days, the value of leaf node is expressed by $V_{month\|week\|day}$. For example, V_{0113} denotes the time granule value of Wednesday, the first week in January. So this multi-tree has 365 leaf nodes to represent a year. The computation expression for value of each leaf node is $V_{month\|week\|day} = H(H(H(H(w)\|month)\|week)\|day)$, i.e. $V_{0113} = H(H(H(H(w)\|01)\|1)\|3) = H(V_{011*}\|3)$. With time multi-tree, doctors can be assigned intermediate nodes easily when they are given time interval weekly or monthly in condition that the timeline is a year and the smallest time granularity is daily or weekly.

However, when a doctor is authorized time interval by days, like from Monday to Friday, he needs to be given five time interval values and computes decryption key in each time granule with time multi-tree. In this situation, time binary tree is more efficient and flexible than time multi-tree.

The method of time tree is faster than most existing time-bound hierarchical key management schemes, since they use linear hash chain to compute time granule value, whose time complexity of hashing operation is $O(t_e - t_b)$. While with binary tree structure, the time complexity of hashing operation is $O(\lceil \log_2(t_e - t_b + 1) \rceil)$. Obviously, the method of time tree is comparatively efficient for computing time granule values. The detail analysis and comparison of time complexity of these two methods will be given in Section 7.2.

4.2.3. Algorithm for Locating the Roots of FBSs

In the previous sections, we have shown how to construct a time tree by given timeline $[0, z]$. However, there is an unsolved issue: that is, when given a time interval $[t_b, t_e]$, how to correctly find root nodes of FBSs in a time tree. Clearly, it is much easier when using time multi-tree than using time binary tree. In this subsection we outline a practical algorithm that is able to effectively locate root nodes of FBSs given any consecutive time granules $[t_b, t_e] \in [0, z]$ of time binary tree.

Algorithm 1 shows the pseudo-code for locating the root nodes of FBSs. Given time interval $[t_b, t_e]$, the procedure $FindRootOfFBS$ exposes a set of positions of FBS roots. Since the given time interval is composed of consecutive leaf nodes, the algorithm analyzes the components of time interval from bottom to top. It uses a structure with two parameters val and pos to represent the position of a node of time binary tree. In addition, a variable $level$ is used to indicate the current number of rounds and initialized by 0. It first allocates two variables $head$ and end to respectively indicate the beginning and end of time interval, that is, the value of $head$ is t_b and the value of end is t_e. If the value of $head$ is odd (or the value of end is even), then this node is one of the nodes we are looking for; Otherwise, the value of $head$ is increased by 1 (or the value of end is reduced by 1). Then remove the last bit of $head$ and end (that is standard right shift of one bit) and go to the next round (the upper level of the time binary tree) until the value of $head$ is less than or equal to end.

Security Comm. Networks 0000; **00**:1–21 © 0000 John Wiley & Sons, Ltd.
DOI: 10.1002/sec
Prepared using secauth.cls

ALGORITHM 1: Pseudo-code for locating the roots of FBSs

```
procedure FindRootOfFBS
level = 0, head = t_b, end = t_e;
while head < end do
    if (head mod 2 == 1) then
        RootNodes → elem.val = head;
        RootNodes → elem.pos = level;
        RootNodes = RootNodes → next;
        head=head + 1;
    end
    if (end mod 2 == 0) then
        RootNodes → elem.val = end;
        RootNodes → elem.pos = level + 1;
        RootNodes = RootNodes → next;
        end = end − 1;
    end
    level + +;
    head = head >> 1;
    end = end >> 1;
end
Print(RootNodes);
```

Take $[t_b, t_e] = [0, 5] = [000, 101]$ as an example (see Fig.5a). Initially, the value of $head$ is 0 (000 in binary) and the value of end is 5 (101 in binary). In the first round (the value of $level$ is 0), no node is singled out, since the value of $head$ is even and the the value of end is odd. Continue to right shift one bit of $head$ and end, the values of $head$ and end become to 0 (000 in binary) and 2 (010 in binary) respectively, and value of pos increases to 1. In the second round, since the value of end is even, the first root node whose $val = 2$ and $pos = 1$ is obtained. Then end reduces to 1. In the third round, after remove the last bit of $head$ and end, the values of them both become to 0. The value of end is even again. So the second root node whose $val = 0$, $pos = 2$ (that is, 0 in binary) is located. The program terminates.

4.3. Rekey

To guarantee the security of EHR system, all keys should be updated periodically. In our scheme, the access key $K_{k,t}$ for security class C_k is changed with current time granule t. The frequency depends on the size of t, such as per day or per week. The class keys K_i for generating access keys and encrypting index nodes are also regularly renewed by updating initial value IV periodically (such as per month or per year). Accordingly, all class keys are changed with IV regularly.

5. IMPLEMENTATION OF RBTBAC PROTOCOL IN EHR SYSTEM

In this section we present the RBTBAC protocol for accessing EHR data based on previous RBTBAC model. The access of EHR data is an interactive process involving user, EHR system and EHR database. As mentioned in Section 3.3, to successfully access the encrypted EHR data, the user should be issued both system identity credential and access credential for the requested data set. An RBTBAC protocol is composed of five sub-protocols: *initialization*, *data encryption*, *user registration*, *user request* and *access and decryption*.

5.1. Initialization

In this phase, system parameters for accessing EHR data are initialized, all class keys are generated and the hierarchical encryption key structure is established as Fig.4a.

1. Encryptor chooses a random integer IV and two secret keyed HMAC $H_K(\cdot)$ and $\overline{H}_{K_x}(\cdot)$, where K is the system access master key and K_x is the class key for security class C_x. Then encryptor runs a polynomial-time group-generating algorithm $\mathcal{G}(1^{n'})$ to generate group $\mathbb{G}$ as described in Section 4.1.2 and selects a collision-resistant hash function $H' : \{0,1\}^* \to \mathbb{G}$.
2. Encryptor computes class keys of encryption key structure using the method in Section 4.1.2. Then encryptor encodes all index nodes with corresponding class keys, that is, computes the values of HMAC $\overline{H}_{K_x}(C_x)$ for node C_x as well as all patient nodes, and updates the secured indices to DB.

5.2. Data Encryption

Encryptor generates the encryption (access) key $K_{k,t}$ for C_k at time granule $t \in [0, z]$ and encrypts the data of security class C_k with key $K_{k,t}$, whenever C_k is a security class of leaf node. $K_{k,t}$ is computed as Equation (5), where $V_{B(t)}$ is computed with the method in Section 4.2.2. It is worth noting that only the data (leaf) nodes of EHR structure are encrypted with the access keys.

$$K_{k,t} = H_K\left(K_k \| V_{B(t)}\right) \tag{5}$$

5.3. User Registration

When a user (doctor) D_i requests registration in the EHR system, TA computes user master key K_{D_i} and issues a system identity credential to user D_i as the identity proof to access the EHR system. The system identity credential should contain $Sig_{TA}\left\{Enc_{PK_{D_i}}(K_{D_i}, H_K(\cdot))\right\}$ and $Enc_{PK_{D_i}}(K_{D_i}, H_K(\cdot))$, where $Sig_{TA}\{M\}$ is a digital signature signed with TA's private key on message M and $Enc_{PK_U}(\cdot)$ is a ciphertext encrypted by public key encryption algorithm with U's public key.

5.4. User Request

Once the user D_i has obtained the system identity credential, he can request access to the EHR data of a specific patient (or patients) with role ro and keyword I within consecutive time granules $[t_b, t_e] \in [0, z]$. TA

first verifies user's system identity credential. Then TA searches the access control policies. If the request matches the access control policies, TA retrieves the set of secured indices $\left\{\overline{H}_{K_{x_r}}\left(\overline{I}_r\right)\right\}$, where $\overline{I}_r$ is the rth retrieved index nodes on keyword I. Next, TA locates the right root nodes of FBSs and calculates the values V^u of these nodes (see Algorithm 1). At last, TA issues the doctor an access credential, which contains $Sig_{TA}\left\{E_{K_{D_i}}\left(t_b, t_e, \{V^u\}, \left\{R_{C_{D_i},C_k}\right\}\right)\right\}$, $E_{K_{D_i}}\left(t_b, t_e, \{V^u\}, \left\{R_{C_{D_i},C_k}\right\}\right)$, $Sig_{TA}\left\{Enc_{PK_{DB}}\left(D_i, t_b, t_e, \left\{\overline{H}_{K_{x_r}}(\overline{I}_r)\right\}\right)\right\}$ and $Enc_{PK_{DB}}\left(D_i, t_b, t_e, \left\{\overline{H}_{K_{x_r}}(\overline{I}_r)\right\}\right)$, where $E_{K_{D_i}}(\cdot)$ is a symmetric encryption algorithm with secret key K_{D_i} of user D_i, $\{V^u\}$ is a set of root nodes' values of FBSs, $\left\{R_{C_{D_i},C_k}\right\}$ is a set of relationship values, $\left\{\overline{H}_{K_{x_r}}(\overline{I}_r)\right\}$ is a set of secured index nodes.

5.5. Access and Decryption

Suppose a user D_i has received an access credential from TA and has granted access time interval $[t_b, t_e]$, then he can decrypt and read EHR data of C_k at any time granule $t \in [t_b, t_e]$, whenever $C_k \prec C_{D_i}$.

1. In the current time granule $t \in [t_b, t_e]$, user D_i requests EHR data with system identity credential and access credential. DB verifies the authenticity of the two credentials by verifying TA's signature and checks the identity of the user and the time granule against the access credential. Then DB retrieves encrypted data of security class C_k by comparing the values of $\left\{\overline{H}_{K_{x_r}}(\overline{I}_r)\right\}$ with stored indices and sends them to user D_i.
2. User D_i retrieves $\{R_{D_i,C_k}\}$ and $V^0, \cdots, V^u$ from the access credential with his secret master key, and computes K_k with Equation (4) and $V_{B(t)}$ with the method in Section 4.2.3. Then user D_i computes decryption key $K_{k,t}$ with Equation (5).
3. Finally, the encrypted data of security class C_k can be decrypted by the authorized user D_i with the derived access key $K_{k,t}$.

Note that, first, in the process of access EHR data, TA is entirely offline except the process of issuing system identity credentials and access credentials, which can be done before the access time interval. In the granted time granules, user communicates with DB directly without involvement of TA. This approach avoids complicated communication among user, TA and DB in each time granule, thereby reducing the communication overload of TA. Second, in order to access and control of EHR data, a special software, which can provide data and key management (such as limiting the saving and copying of ciphertext and plaintext of EHR data, and deleting EHR data and access key when current time granule has expired), must be installed on the client.

6. ENFORCEMENT VERIFICATION

This section focuses on the discussion of security and privacy analyses of our RBTBAC scheme.

6.1. Security Analysis

The discussion of security of RBTBAC protocol contains the security against possible attacks and the secure indices. Basically, successfully accessing and decrypting EHR data must have both valid system identity credential and access credential. In our scenario, we suppose that user never reveals his system identity credential to others, namely his master key K_{D_i} is secretly kept. Once the user is compromised (namely, his master key is exposed), his system identity credential and his master key should be revoked by TA immediately.

6.1.1. Attack from the Outside

The first scenario is an adversary without system identity credential and access credential attempts to access data. He may try to request encrypted EHR data from DB with a forged system identity credential and a forged access credential (or an access credential from other user). Obviously, the adversary will fail to pass the verification, when DB verifies the signature of system identity credential with TA's public key. In addition, the adversary cannot decrypt the EHR data without valid system identity credential and access credential even if he gets the encrypted EHR data.

6.1.2. Attack on Unauthorized Class

The second situation is a malicious user D_i of the EHR system may attempt to access data of an unauthorized class C'_k (which has never accessed by the user) in consecutive time granules $[t_b, t_e]$. Assume the user has legally obtained access credential containing $Sig_{TA}\left\{E_{K_{D_i}}\left(t_b', t_e', \{V^u\}, \left\{R_{C_{D_i},C_k}\right\}\right)\right\}$, $E_{K_{D_i}}\left(t_b', t_e', \{V^u\}, \left\{R_{C_{D_i},C_k}\right\}\right)$, $Sig_{TA}\left\{Enc_{PK_{DB}}\left(D_i, t_b', t_e', \left\{\overline{H}_{K_{x_r}}(\overline{I}_r)\right\}\right)\right\}$ and $Enc_{PK_{DB}}\left(D_i, t_b', t_e', \left\{\overline{H}_{K_{x_r}}(\overline{I}_r)\right\}\right)$, where $t_b' \le t_b \le t_e \le t_e'$, $C_k \prec C_{D_i}$, $C_{k'} \not\prec C_{D_i}$ and $k \ne k'$. To request the EHR data of class C'_k, the user needs to forge values $\left\{R_{C_{D_i},C_{k'}}\right\}$ and $\left\{\overline{H}_{K_{x_r}}(\overline{I'}_r)\right\}$. However, the user cannot pass the verification using a forged access credential without TA's signature. Therefore, the user cannot access any EHR data of unauthorized class because of the unforgeability of the signature.

6.1.3. Attack in Unauthorized Time Interval

The third type of attacks is a malicious user D_i of EHR system may attempt to access data of security class C_k in

Security Comm. Networks 0000; **00**:1–21 © 0000 John Wiley & Sons, Ltd.
DOI: 10.1002/sec
Prepared using secauth.cls

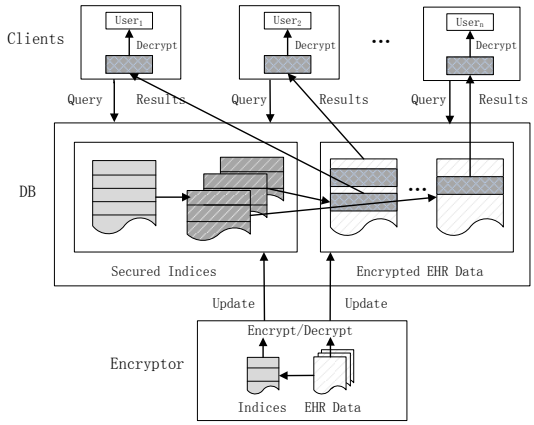


Figure 6. Secure search over EHR data

unauthorized time interval $[t_b{}', t_e{}']$. Assume that in time interval $[t_b, t_e]$, where $t_b \leq t_e < t_b{}' \leq t_e{}'$, user D_i has right to access the data of class C_k, that is, he can get values $\left\{R_{C_{D_i},C_k}\right\}$ from the valid access credential. In addition, in time interval $[t_b{}', t_e{}']$, the user may obtain $t_b{}'$, $t_e{}'$ and $\left\{V^{u\prime}\right\}$ from his other access credentials. But he still can not get $Enc_{PK_{DB}}\left(D_i, t_b{}', t_e{}', \left\{\overline{H}_{K_{x_r}}(\overline{I}_r)\right\}\right)$ without DB's private key. Even if the user successfully forged the values, he will fail to pass the verification using a forged access credential without TA's signature. Thus, the user cannot access EHR data in unauthorized time interval because of the unforgeability of the signature.

6.1.4. Collusion Attack

Some existing time-bound key management schemes are proved insecure against collusion attack, such as Tzeng's scheme [6] is insecure against Yi and Ye's attack [13], Chien's scheme [7] is insecure against Yi's attack [19], and Bertino's scheme [11] is insecure against Sun's attack [20]. These collusion attacks can be described in our scheme like this: one or two users collude with the other user D_i to derive certain access credential and access key of a unauthorized class $C_{k'}$, then try to request encrypted EHR data of $C_{k'}$ with the forged access credential from DB. Suppose D_i is able to derive the access key $K_{k',t}$ of $C_{k'}$ by getting $K_{k'}$ and $V_{B(t)}$ from conspirators. In addition, he needs to request encrypted data with a forged access credential. Similar to above attacks, based on the unforgeability of the signature, user D_i will fail to obtain the encrypted data from DB.

6.1.5. Security of Indices and Search

As mentioned in Section 5.1, each index node is encoded by its class key K_x which is never exposed to users or untrusted DB. Therefore, a vicious user or adversary will not obtain any information of indices even if he accesses the index nodes by malicious means.

In order to successfully decrypt an index node, the adversary should know the class key K_x, which is secretly computed with CID_x and H' by encryptor. Obviously, without knowing CID_x and H', adversary cannot figure out K_x directly. From the hierarchical key structure, there are two ways to compute the class key K_x. The first is calculating from top to bottom. The second is calculating in reverse. With top-down approach, class key K_x can be computed through one of its superordinate nodes K_i and the corresponding relationship value R_{C_i,C_x}, where $C_x \prec C_i$. In our scheme, the adversary cannot compute K_x, since both of these two values are kept secret, even if he has K_{D_i}, where $C_x \prec C_{D_i}$. With bottom-up approach, class key K_x can be computed through one of its subordinate nodes K_j and the corresponding relationship value R_{C_x,C_j}, where $C_j \prec C_x$. Again, the adversary cannot work out K_x without knowing both of two values, even if he get a class key K_k, where $C_k \prec C_x$.

For searchability as shown in Fig. 6, DB is given the encoded indices of requested EHR data through the access credential from user. Then it compares the given values with all entries of secured indices, which are encrypted and updated by encryptor. If any entry of secured indices is identical to the given value, DB goes to the pointed address to next level search until the encrypted EHR data are located. Finally, after the requested EHR data of security class C_k has been retrieved, DB sends them to the user. Throughout the search process, DB does not have any knowledge of indices and EHR data.

6.2. Privacy Analysis

Our RBTBAC model is capable of providing better privacy for patients from four aspects.

First, when a user accesses EHR data, our RBTBAC scheme not only narrows the access scope of EHR data by dividing EHR data into different security classes, but also limits the access time to the period of providing healthcare treatment. The user consequently is only given the right to access EHR data of authorized classes in authorized time interval. Accordingly, the privacy of patient can be protected from both space and time dimension.

Second, all the index nodes are securely stored in remote DB, so that both users and DB are unaware of any intermediate index node and internal structure of EHR data. It greatly reduces the probability a deliberate user deduces additional information about the patient from intermediate nodes or the relationship of partial ordered nodes. For example, a user has accessed all blood test results of the patient under different disease category nodes. Then he may infer other disease information about the patient if he has knowledge of those disease category nodes.

Third, unlike most existing key management schemes, our RBTB key management scheme does not publicly disclose any relationship value between partial ordered classes, even the relationship between patients and doctors. Thus, it avoids the situation where a professional user could infer the disease of patient from the doctor-patient

relationship. For example, we can infer that a patient has cancer if he has a relationship with a cancer specialist.

Finally, we use patient-controlled access policy to better protect privacy of patients. We strongly suggest that patient should be involved throughout the process of developing access control policies to determine the sharing of their own sensitive information with other users and CDOs. This can help patients better understand the EHR system, control their EHR data more tightly, and know who is accessing their EHR data at what time.

6.3. Credential Revocation

Usually, credential revocation involves the certificate authority periodically issuing a signed data structure called a *certificate revocation list (CRL)*. CRL is used for revoking credential in certificate using system which has certificate authority. A CRL is a time stamped list identifying revoked certificates which is signed by a CA or CRL issuer and made freely available in a public repository. Each revoked credential is identified in a CRL by its credential serial number.

In our model, once access credential has expired, it is invalid automatically without being revoked by TA. When a user requests EHR data with an expired access credential, DB will detect this unauthorized access by checking the access credential and reject the request. However, there are still two cases that TA has to take the initiative to revoke the user's access credential even the system identity credential.

The first case is a user terminates healthcare treatment during the authorized time interval. In this case, the EHR data cannot be accessed by the user anymore and the access credential should be revoked by TA immediately. The revocation of access credential can be done by following two steps: First, TA updates *Access Credential Revocation List (ACRL)* by adding an entry of serial number of the revoked access credential. Second, DB checks the ACRL with access credential for each access request. If the serial number of access credential is in ACRL, DB rejects the access request.

The second is a worse case that a user is compromised by an adversary. In this situation, not only all access credentials but also the system identity credential of the user should be revoked. Once the client of user is compromised, the master key of the user is released to public. TA adds the serial number of system identity credential into *System Identity Credential Revocation List (SICRL)* and appends all serial numbers of access credentials held by the compromised user into ACRL. Consequently, each request for access credential from a user should be validated by TA through checking the SICRL.

7. SPACE AND TIME COMPLEXITY

In the RBTBAC model, RBTB key management scheme is the most important part. Therefore, performance analysis of RBTBAC model can be translated to the analysis on RBTB key management scheme. The performance measure of RBTB key management scheme contains two folds: space complexity and time complexity. The discussion is further focused on two aspects: server's perspective and user's perspective.

7.1. Space Complexity

We first define some parameters to facilitate the following analysis. Let N_T be the total number of security classes in the encryption key structure (see Fig.4a), N_{leaf} is the number of data nodes (leaf nodes) of both encryption and decryption key structures in Fig.4 and N_k is the number of security classes authorized to user in the current time granule. Then the analysis on the server and client are described below.

Generally, the space for public information is a main parameter to measure the space cost of a key management scheme. The comparison of space complexity of public values is shown in Table III. Our RBTB key management scheme does not publish any relationship value on the board, as opposed to other existing schemes where all partial ordered relationship values are published. Thus, our scheme is space-saving in term of public information.

For encryptor, it stores two kinds of keys: long-term class keys for index nodes and access keys for data nodes. The space to store all these two types of keys is relative to the total number of nodes in key structures, that is, N_T long-term class keys and N_{leaf} access keys. In addition, to save the storage space of server, our scheme does not require encryptor to store any system identity credential or access credential. On the contrary, encryptor sends all needed information to user through the system identity credential and access credential at the first request. When the access request comes, DB only needs to verify the information contained in the two credentials.

For a user of EHR system, he/she is required to securely store one long-term master key and N_k temporary access keys, plus one system identity credential and several access credentials. However, the access keys and access credentials will be invalid automatically and deleted when the granted time interval has expired. Chien's scheme [7] and Bertino's scheme [11] use a tamper-resistant device to store all the information and securely compute access key for specific time granule, but issuing a tamper-resistant device for each user is impractical in distributed and worldwide systems such as EHR system. Their approach actually sacrifices convenience for security.

7.2. Time Complexity

Table I contains a summary of hardware and software used in our experiments. The experiments of this paper were run in Code Blocks. Each experiment was run 20 times and averaged. It is known that encrypting all EHR data in each time granule is time-consuming in such time-bound key management scheme. The speed of encryption depends on the used encryption algorithm and server

Security Comm. Networks 0000; **00**:1–21 © 0000 John Wiley & Sons, Ltd.
DOI: 10.1002/sec
Prepared using secauth.cls

Table I. Experimental Setup

Hardware/Software	Components
Processor	Intel(R) Core(TM)2 Duo E7500 2.93GHz
Memory	2.0GB
Operating system	Windows 7 Ultimate
Programming language/Library	C++/Crypt++
IDE	Code::Blocks

Table II. The Number of Hashing Operations in Each Time Granule

$[0, z]$	Linear chain	Binary tree	Multi-tree
One week	7	3	
Fortnight	14	4	
One month	30	5	
One year	365	9	3

performance, which are beyond scope of this paper. A recommended approach is to respectively encrypt different parts of the EHR data with multiple processors at same time. In addition to encryption time, another important parameter to measure the time complexity of a time-bound key management scheme is the key generation time. In this section, we discuss the time complexity of generating access keys. The analysis is respectively given in two folds: the time of encryption key generation and the time of decryption key generation.

7.2.1. Computation of Encryption Key

For the server side, the time for generating an encryption key within a time granule mainly depends on the time of computing time granule values, since the class key is unchanged. The main operation in generation of time granule values is hashing operation. Thus the efficiency for producing encryption key is principally determined by the number of hashing operations. From above analysis, for a fixed timeline $[0, z]$, the number of hashing operations in each time granule is less than $\lceil \log_2 (z + 1) \rceil$ with binary tree structure, as opposed to linear hash chain method used in most existing time-bound key management schemes [7, 11, 21, 14] where number of hashing operations is $z + 1$.

Assume the smallest time granule is one day, the number of hashing operations in each time granule is shown in Table II by comparing our time tree method with existing linear chain method. We can infer that, given the same time granularity, the longer the timeline $[0, z]$ is, the more efficient the time tree is. For example, when the time granule is daily and the whole timeline is one year, it needs 365 hashing operations with linear chain method to compute time granule value, while it needs 9 hashing operations with time binary tree method, furthermore it only needs 3 hashing operations with time multi-tree method in Fig.5b.

7.2.2. Computation of Decryption Key

Before issuing an access credential, encryptor has to compute granted time parameters for user. In the linear

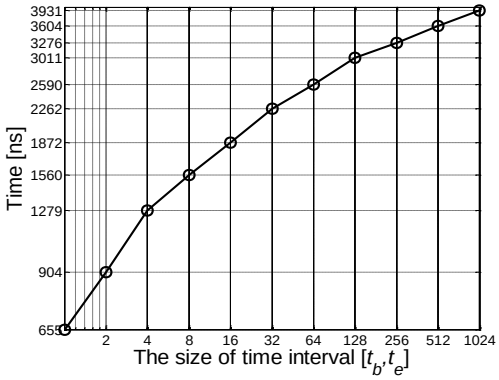


Figure 7. Time for locating the roots of FBSs

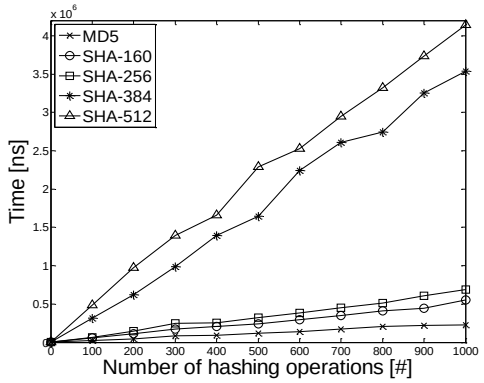


Figure 8. Time for hashing operations

chain method, the given time parameters are $H^{t_b}(a)$ and $H^{z-t_e}(b)$. Therefore, the time complexity for computing time parameters is $O(z - (t_e - t_b))$. In our time tree method, the time for encryptor generating time parameters includes two parts: the time for locating root nodes of FBSs and the time for computing hash values of these root nodes.

According to Algorithm 1, we can infer that the efficiency of locating root nodes only depends on the scale of time interval $[t_b, t_e]$. The performance of Algorithm 1 is shown in Fig.7. For the sake of convenience, we discuss the efficiency based on the condition that the scale of time interval $[t_b, t_e]$ (namely $t_e - t_b + 1$) is power of 2. Here we use binary tree as time structure. The average results are displayed in Fig.7. When the scale of $[t_b, t_e]$ reaches to 2^{10}, it spends almost 4000 ns to locate root nodes of FBSs, which is much faster than hash operations (see Fig.8).

Given fixed scale of whole timeline, say $[0, 1023]$, we illustrate the comparison of the two methods in Fig.9 and Fig.10. Fig.9 shows the comparison of hashing operations between binary tree method and linear chain method. It is seen that the number of hashing operations of binary tree structure drops as the time interval lengthens, while the number of hashing operations of linear chain structure is gradually increases. But when the scale of $[t_b, t_e]$ exceeds

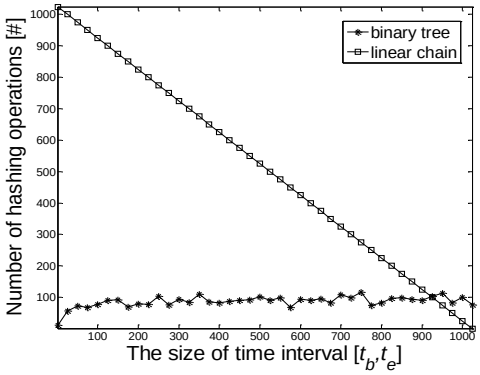


Figure 9. Comparing the number of hashing operations for encryptor generating time parameters

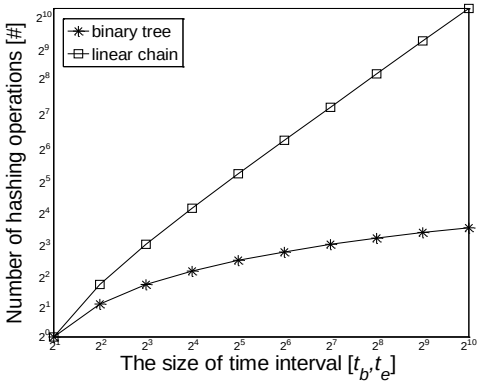


Figure 11. Comparing the number of hashing operations for user computing time granule value

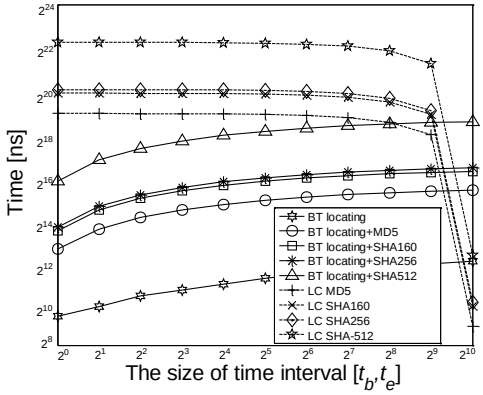


Figure 10. Time comparison for encryptor generating time parameters

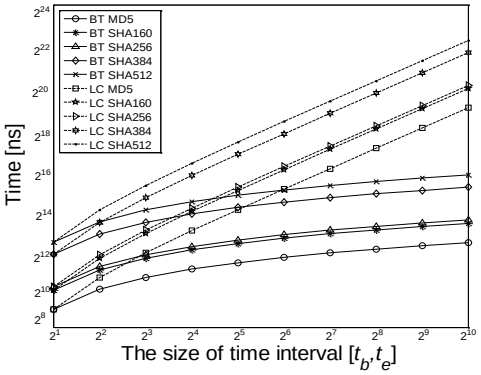


Figure 12. Time comparison for user computing time granule value

around 930, the linear chain method is slightly faster than binary tree method. However in the EHR system, users usually are granted a short time interval to access EHR data. Therefore, in most cases, binary tree structure is extremely efficient compared to the linear chain structure. Fig.10 demonstrates the total time consumption for encryptor generating time granule parameters in both binary tree method and linear chain method. For saving space, binary tree method is abbreviated as BT and linear chain method is abbreviated as LC in the graph. The results are actually the sum of the time for locating root nodes (see Fig.7) and the time for hashing operations (see Fig.8). From Fig.10 we have conclusion that our scheme is much more efficient than existing time-bound key management schemes in the server side.

In user side, the time for producing the decryption key also consists of two parts. The first part is the time for generating long-term class key with user's master key. The second part is the time for computing time granule value. For computing class key, the user only needs to compute one modular exponentiation in our scheme.

For calculating time granule value, in contrast to $O(t_e - t_b)$ with linear chain method, the time complexity of hashing operations is $O(\lfloor \log_2(t_e - t_b + 1) \rfloor)$ with binary tree method. Distinctly, the binary tree structure is more efficient than linear chain structure, and when the granted time interval becomes longer, the gap will be even greater as shown in Fig.11. When the size of given time interval goes to 2^{10}, the number of hashing operations also reaches to 2^{10} in linear chain method. Oppositely, the number of hashing operations is still approximately 10 with binary tree method. The experimental results are shown in Fig.12. The graphic shapes of the set of binary tree methods are much flatter than the set of linear chain methods'. Therefore, our scheme is time-saving in user side.

In summary, our RBTB key management scheme is space-saving, and more important, is more efficient than most existing schemes.

Security Comm. Networks 0000; **00**:1–21 © 0000 John Wiley & Sons, Ltd.
DOI: 10.1002/sec
Prepared using secauth.cls

8. RELATED WORK

The main works related to our scheme are access control and time-bound key management technique. In this section, we briefly describe existing works on these two fields.

8.1. Other Access Control Models

Recently, Israelson et al. [22] suggest restricting access to EHRs to users with a valid token. Access tokens are issued to users for a prescribed amount of time where both the user and the amount of time are authorized by the patient. In our scheme, we employ an extended role-based access control model, role-based and time-bound access control model, to provide privacy aware access control on EHR data that allows users to access authorized data only in a specific time interval. In this section, we describe role-based, attribute-based and cryptographic access control models, which are considered more applicable to EHR setting.

8.1.1. Role-Based Access Control

There are a number of works on RBAC [23] for medical systems [24, 25, 26, 27, 28]. Among them, the authors of [24] proposed a set of authorization policies enforcing role-based access control for the electronic transfer of prescriptions. Then an implementation of EHR prototype system including a basic network and role-based security infrastructure for the United Kingdom National Health Service is demonstrated in [25]. Cassandra [26] is a trust management and role-based policy specification language, which was presented for expressing access control policies in large-scale distributed systems. In this work, a case study discussed how the language can be used to specify security policies for a UK national EHR system. However, none of above approaches took into account of the composition feature of EHR document, and therefore cannot support a more fine-grained access control to selectively share composite EHR data. Recently, the authors of [27] argued an approach for modeling role-based access control scheme for composite EHR. They designed three dimensional properties for each sub-object and property-based authorization to provide a flexible yet efficient means to select and authorize a collection of sub-objects.

Healthcare RBAC Role Engineering Process [28] is another significant work for the Healthcare RBAC Task Force (TF) developed by Science Applications International Corporation (SAIC) in May 2004. The RBAC TF is engaged in a collaborative effort to define common industry-wide roles capable of supporting health information systems. The goal of the RBAC TF is to formalize this shared effort through an ANSI-approved healthcare role standard. The Healthcare RBAC Role Engineering Process document describes the methodology that the TF will follow in pursuing its goal and the mechanisms, processes and products that will be used to create, harmonize, and report TF efforts.

8.1.2. Attribute-Based Access Control

In ABAC, access is granted not based on the rights of the subject associated with a user after authentication, but based on attributes of the user. The user has to prove so called claims about his attributes to access control engine. An attribute-based access control policy specifies which claims need to satisfied in order to grant access to an object.

Some researchers have applied ABAC to access the medical records. Paper [29] presented a patient-centric, attribute-based and source verifiable framework for health record sharing based on MedVault project [30]. It provides patients with substantial control over how their information is shared and with whom and permits fine-grained decisions based on the use of attribute-based techniques for authorization and access control. Another attribute-based health records system currently under development is the ”security infrastructure and national patient summary” as part of the Swedish national eHealth system [31]. Like MedVault, the system is being implemented in XACML (eXtensible Access Control Markup Language) [32], which is able to provide decentralized administration and credentials distribution. Literature [33] presented writing XACML policies and examined performances of access control time using various Sun XACML access control policies in case when attributes are in hierarchical structure in a distributed EHR.

8.1.3. Cryptographic Access Control

Cryptographic access control is a new distributed access control paradigm designed for information systems. It defines an implicit access control mechanism, which relies on cryptography to provide confidentiality and integrity of data managed by the system. It is particularly designed to operate in untrusted environments where the lack of global knowledge and control. Now, it is applied to security-enhanced systems, e.g. EHR sharing systems.

In the existing CAC schemes, the vast majority are based on hierarchical structure. Hierarchical cryptographic access control schemes first emerged in [34, 35], which are more general and capable of providing security in different contexts without requiring extensive changes to the fundamental architecture. For instance, in situations that require data outsourcing, CAC schemes are useful because the data can be double encrypted to prevent a service provider from viewing the information but yet be able to run queries or other operations on the data and return a result to a user who can decrypt the data using the keys in their possession [36]. CAC schemes are typically modeled in the form of a partially ordered set of security classes that each represents a group of users requesting access to a portion of the data on the system.

From the end of last century, CAC has been widely used in file system. A group sharing and random access

model in cryptographic storage file systems was proposed [37]. Paper [38] presented a cryptographic access control in distributed file systems, in which data are encrypted as the applications store them on a server. It separated read and write access: read access to the physical storage device is granted to all principals and write access can be granted to everyone. The authors of [39] first analyzed *nix systems and identify an urgent need for better privacy support in their data sharing mechanisms and gave two solutions for privacy enhancement. Later, they proposed a data sharing platform, named SHAROES, for outsourced storage environment. It provides rich *nix-like data sharing semantics [40].

8.2. Other Time-Bound Key Management Approaches

We have already introduced time-bound key management technique in Section 2.2 and discussed how our approach tackles the problem differently. Table III summarizes the comparison of time-bound key management schemes on various attributes. Specifically, our scheme provides privacy and security enhancement for EHR system by combining role-based access control. In the rest of this section, we discuss other existing time-bound key management schemes.

In 2002, Tzeng [6] first proposed a time-bound hierarchical key management scheme that requires each user to store information whose size does not depend on the number of time periods. However, this scheme is costly because the Lucas function operation incurs heavy computational load. Most importantly, Tzeng's scheme has been proved to be insecure against collusive attacks, whereby two or more users assigned to some classes in distinct time period, collude to compute a key to which they are not entitled [13]. Subsequently, Chien [7] put forward an efficient time-bound hierarchical key management scheme based on tamper-resistant devices. However, Santis and Yi [41] [19] showed that malicious users can collusively misuse their devices to gain unauthorized accesses and also proposed countermeasures. Then Xu [21] improved Chien's scheme without public key cryptography and they proved their scheme is as efficient as Chien's and resistant to Yi's three-party collusion attack. Another time-bound hierarchical key assignment scheme was proposed by Huang and Chang [8] and later shown to be insecure against collusive attacks too [42]. Yeh [9] proposed an RSA-based time-bound hierarchical key assignment scheme, which was proved to be insecure against collusive attacks in [43]. Wang and Laih [10] used a modification of the Akl-Taylor scheme to construct a time-bound hierarchical key management scheme. Then, two provable-secure time-bound hierarchical key assignment schemes the one is based on symmetric encryption schemes, whereas, the other makes use of bilinear maps, were posed by Ateniese and Santis [43], who proved their schemes are simultaneously practical and provably-secure. After that new constructions for provably-secure time-bound hierarchical key assignment schemes were brought forward and tradeoff among storage space and key derivation time were exhibited by Santis [44]. But provably-secure schemes are inefficient when the system has large number of data. Liu and Zhong [12] advanced a practical time-bound hierarchical key scheme without tamper-resistant device, which was claimed to be more secure and need less computational time. However, in their scheme users need to hold a number of keys to decrypt the data on multiple classes. Recently, Bertino [11] proposed another new time-bound scheme using elliptic curve cryptography and claimed that their scheme was efficient and secure against possible attacks. However, Sun [20] found that Bertino's scheme was not as secure as they claimed and some possible improvements were proposed. Moreover, tamper-resistant device makes Bertino's scheme difficult to implement in EHR systems since it is hard to securely issue tamper-resistant device when users are distributed around the world. Unfortunately, above schemes are only applicable to static access hierarchy, since they cannot be used in dynamic and flexible EHR system. Sui [14] presented a key generation and assignment scheme for time-bound hierarchy dynamic access control. But their scheme is not efficient enough when applying it to EHR system which has very large number of medical data.

For applying time-bound key management scheme to EHR system, it should be fully considered and re-developed. Our scheme focuses on integrating role-based access control with time-bound key management, so that it satisfies following four requirements of EHR setting: security (especially against collusion attack), the minimal key storage space in user side (ideally, only one key), privacy preserving (especially for patient's sensitive medical information), and efficiency (mainly measured by the time of generating encryption/decryption keys).

9. CONCLUSIONS AND OTHER APPLICATIONS

We have put forward a practical Role-Based and Time-Bound Access Control (RBTBAC) model for EHR system. Differing from basic RBAC model, this access control model emphasizes more on the flexibility of roles and has the capability to control the access of sensitive data from time dimension. Technically, we have proposed a role based and time bound (RBTB) hierarchical key management scheme. For role-based, we have developed a privacy-aware and dynamic key structure. For time-bound, we have employed a time tree method for generating time granule values. We have analytically and experimentally proved that RBTBAC model is more suitable for EHR system since it offers high-efficiency and better security and privacy for patients.

In our future work, we are interested in experiments on real medical data which are hard to get. As the diverse

Security Comm. Networks 0000; **00**:1–21 © 0000 John Wiley & Sons, Ltd.
DOI: 10.1002/sec
Prepared using secauth.cls

Table III. Comparison of Time-Bound Key Management Schemes

	Tzeng's scheme [6]	Chien's scheme [7]	Bertino's scheme [11]	Our scheme
Implementation requirements	Lucas function	Tamper-resistant device	Tamper-resistant device, ECC	System identity credential, access credential
Number of public values	$N_T + 6$	$N_T - 1$	$N_T(N_T - 1)/2$	0
Number of operations for encryptor deriving an access key	T_e,T_L,T_h	$(z + 2)T_h$	$(z + 2)T_h + T_E$	T_e,$(\lceil \log_2(z + 1) \rceil + 2)T_h, T_e$
Number of operations for user deriving an access key of data node (l-edge-distance child class)	$(t_e - t_b + r)T_e, (t_e - t_b)T_L, T_h$	$(t_e - t_b + 1 + l)T_h$	$(t_e - t_b + 2)T_h, T_E$	$(\lfloor \log_2(t_e - t_b + 1) \rfloor + 1)T_h, T_e$
Collision attack	Insecure (Yi and Ye's attack)	Insecure (Yi's attack)	Insecure (Sun's attack)	Secure

Suppose $C_k \preceq C_x$,$t \in [t_b, t_e]$.
N_T: total number of security classes $|C|$.
r: number of child classes C_x on path from C_x to C_k.
T_h: the time complexity for one hashing operation.
T_e: the time complexity for one modular exponentiation.
T_L: the time complexity for one Lucas function operation.
T_E: the time complexity for one elliptic curve scalar multiplication.

formats of medical data which are different from the format of general file, the preprocessing of medical data that makes the program can directly manipulate these data is a tough problem. The experiments will focus on the efficiency of encryption on a mass of real medical data and possible improvements.

It is important to notice that our RBTBAC model can be applied to many different fields, especially sensitive information system such as government or military systems, banking systems and e-commerce systems. Above description and experiments show that our model can provide more stringent mandatory access control (MAC) from both spatial and temporal dimensions and data confidentiality for this kind of system.

Additionally, the RBTBAC model also can be used in the systems storing a mass of data on the untrusted remote DB or cloud. For such kind of systems, data encrypted stored is necessary. So, how to distribute keys for the legitimate users and how to build an index on the ciphertext are the key issues. Obviously, our RBTB key management scheme can be used to solve these issues. Moreover, our scheme gives a solution for the sharing of data across security domains between different settings. It also can be used to guarantee better security and privacy of data sharing.

ACKNOWLEDGEMENT

This work is supported by the "Strategic Priority Research Program" of the Chinese Academy of Sciences, Grants No. XDA06010701, XDA06030601, XDA06040100, China Postdoctoral Science Foundation (No.2012M510567), National Natural Science Foundation of China (No.61170280). This work is partially supported by grants from NSF CISE NetSE program, Cross-Cutting program, an IBM faculty award and a grant from Intel ISTC.

REFERENCES

1. Electronic health record 2011. Http://en.wikipedia.org/wiki/Electronic-health-record.
2. Health insurance portability and accountability act (hipaa) August 21 1996. (Accessed October 2011).
3. Health information technology for economic and clinical health (hitech) act, title xiii of division a and title iv of division b of the american recovery and reinvestment act of 2009 (arra) 2009. (Accessed November 2011).
4. Zhang R, Liu L. Security models and requirements for healthcare application clouds. *CLOUD '10: Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing*, IEEE Computer Society: Washington, DC, USA, 2010; 268–275.
5. Hacigumus H, Iyer B, Mehrotra S. Providing database as a service. *Proceedings of the 18th International Conference on Data Engineering*, ICDE '02, IEEE Computer Society: Washington, DC, USA, 2002; 29–.
6. Tzeng WG. A time-bound cryptographic key assignment scheme for access control in a hierarchy. *IEEE Trans. on Knowl. and Data Eng.* 2002; **14**(1):182–188.
7. Chien HY. Efficient time-bound hierarchical key assignment scheme. *IEEE Trans. on Knowl. and Data Eng.* 2004; **16**(10):1301–1304.
8. Huang HF, Chang CC. A new cryptographic key assignment scheme with time-constraint access control in a hierarchy. *Computer Standards and Interfaces* 2004; **26**(3):159–166.
9. Yeh Jh. An rsa-based time-bound hierarchical key assignment scheme for electronic article subscription. *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, ACM: New York, NY, USA, 2005;

285–286.
10. Wang SY, Laih CS. Merging: An efficient solution for a time-bound hierarchical key assignment scheme. *IEEE Trans. Dependable Secur. Comput.* 2006; **3**(1):91.
11. Bertino E, Shang N, Wagstaff Jr SS. An efficient time-bound hierarchical key management scheme for secure broadcasting. *IEEE Trans. Dependable Secur. Comput.* 2008; **5**(2):65–70.
12. Liu JQ, Zhong S. A pratical time bound hierarchical key scheme. *International Journal of Innovative Computing, Information and Control* 2009; **5**(2):3241–3247.
13. Yi X, Ye Y. Security of tzeng's time-bound key assignment scheme for access control in a hierarchy. *IEEE Trans. on Knowl. and Data Eng.* 2003; **15**(4):1054–1055.
14. Sui Y, Maino F, Guo Y, Wang K, Zou X. An efficient time-bound access control scheme for dynamic access hierarchy. *Proceedings of the 2009 Fifth International Conference on Mobile Ad-hoc and Sensor Networks*, MSN '09, IEEE Computer Society: Washington, DC, USA, 2009; 279–286.
15. Sandhu RS, Coyne EJ, Feinstein HL, Youman CE. Role-based access control models. *Computer* 1996; **29**(2):38–47.
16. Nyanchama M, Osborn S. The role graph model and conflict of interest. *ACM Trans. Inf. Syst. Secur.* 1999; **2**(1):3–33.
17. Sandhu R, Ferraiolo D, Kuhn R. The nist model for role-based access control: towards a unified standard. *RBAC '00: Proceedings of the fifth ACM workshop on Role-based access control*, ACM: New York, NY, USA, 2000; 47–63.
18. Katz J, Lindell Y. *Introduction to modern cryptography*. Chapman & Hall/CRC cryptography and network security, Chapman & Hall/CRC, 2008.
19. Yi X. Security of chien's efficient time-bound hierarchical key assignment scheme. *IEEE Trans. on Knowl. and Data Eng.* 2005; **17**(9):1298–1299.
20. Sun HM, Wang KH, Chen CM. On the security of an efficient time-bound hierarchical key management scheme. *IEEE Trans. Dependable Secur. Comput.* 2009; **6**(2):159–160.
21. Xu Q, He M, Harn L. An improved time-bound hierarchical key assignment scheme. *Proceedings of the 2008 IEEE Asia-Pacific Services Computing Conference*, IEEE Computer Society: Washington, DC, USA, 2008; 1489–1494.
22. Israelson J, Cankaya EC. A hybrid web based personal health record system shielded with comprehensive security. *Proceedings of the 2012 45th Hawaii International Conference on System Sciences*, HICSS '12, IEEE Computer Society: Washington, DC, USA, 2012; 2958–2968.
23. Sandhu RS, Coyne EJ, Feinstein HL, Youman CE. Role-based access control models. *Computer* 1996; **29**:38–47.
24. Chadwick D, Mundy D. Policy based electronic transmission of prescriptions. *Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on*, 2003; 197 – 206.
25. Eyers D, Bacon J, Moody K. Oasis role-based access control for electronic health records. *Software, IEE Proceedings* - february 2006; **153**(1):16 – 23.
26. Becker M, Sewell P. Cassandra: flexible trust management, applied to electronic health records. *Computer Security Foundations Workshop, 2004. Proceedings. 17th IEEE*, 2004; 139 – 154.
27. Jin J, Ahn GJ, Covington MJ, Zhang X. Toward an access control model for sharing composite electronic health records. *Information Systems Journal* 2008; .
28. (SAIC) SAIC. Role-based access control (rbac) role engineering process version 3.0 may 31 2004.
29. Mohan A, Bauer D, Blough DM, Ahamad M, Krishnan R, Liu L, Mashima D, Palanisamy B. A patient-centric, attribute-based, source-verifiable framework for health record sharing 2009.
30. Medvault: Ensuring security and privacy for medical data. Http://medvault.gtisc.gatech.edu/. (Accessed November 2009).
31. Hagner M. Security infrastructure and national patent summary 2007.
32. extensible access control markup language (xacml) version 2.0 Feburary 1 2005.
33. Sucurovic S. An approach to access control in electronic health record. *Journal of Medical Systems* 2010; **34**:659–666.
34. Akl SG, Taylor PD. Cryptographic solution to a problem of access control in a hierarchy. *ACM Trans. Comput. Syst.* August 1983; **1**:239–248.
35. Sandhu RS. Cryptographic implementation of a tree hierarchy for access control. *Inf. Process. Lett.* February 1988; **27**:95–98.
36. di Vimercati SDC, Foresti S, Jajodia S, Paraboschi S, Samarati P. Over-encryption: management of access control evolution on outsourced data. *Proceedings of the 33rd international conference on Very large data bases*, VLDB '07, VLDB Endowment, 2007; 123–134.
37. Rivest RL, Fu K, Fu KE. Group sharing and random access in cryptographic storage file systems. *Technical Report*, Masters thesis, MIT 1999.
38. Harrington A, Jensen C. Cryptographic access control in a distributed file system. *Proceedings of the eighth ACM symposium on Access control models and technologies*, SACMAT '03, ACM: New York, NY, USA, 2003; 158–165.
39. Singh A, Liu L, Ahamad M. Privacy analysis and enhancements for data sharing in *nix systems. *Int. J. Inf. Comput. Secur.* January 2008; **2**:376–410.
40. Singh A, Liu L. Sharoes: A data sharing platform for outsourced enterprise storage environments.

Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on, 2008; 993 –1002.
41. De Santis A, Ferrara AL, Masucci B. Enforcing the security of a time-bound hierarchical key assignment scheme. *Information Sciences* 2006; **176**(12):1684–1694.
42. Tang Q, Mitchell CJ. Comments on a cryptographic key assignment scheme. *Comput. Stand. Interfaces* 2005; **27**(3):323–326.
43. Ateniese G, De Santis A, Ferrara AL, Masucci B. Provably-secure time-bound hierarchical key assignment schemes. *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, ACM: New York, NY, USA, 2006; 288–297.
44. De Santis A, Ferrara AL, Masucci B. New constructions for provably-secure time-bound hierarchical key assignment schemes. *SACMAT '07: Proceedings of the 12th ACM symposium on Access control models and technologies*, ACM: New York, NY, USA, 2007; 133–138.