oGBAC—A Group Based Access Control Framework for Information Sharing in Online Social Networks

Donghui Hu, *Member, IEEE,* Chunya Hu, Yuqi Fan, Xintao Wu, *Member, IEEE,*

***Abstract*—Internet users receive various online social networks (OSNs) services, however, providers of OSNs do not always provide users fine-grained privacy protection mechanisms with sufficient privacy protection for shared resources. In this paper, we propose a formal Group-Based Access Control (oGBAC) framework for preventing privacy disclosure when sharing information within or among groups in OSNs. Our framework extends the group-centric Secure Information Sharing (g-SIS) models by adapting the concept of the group to OSNs. We impose some restrictions to the group and information flow among groups to ensure that operations cannot incur privacy disclosure when sharing information among friends in OSNs. In view of characteristics of OSNs and the requirements of secure information flow, the oGBAC model also incorporates some ideas from the Attribute-Based Access Control (ABAC) to develop information flow based rules using relationship among attributes (such as tags, time and security levels) of objects and subjects in OSNs. Administration related rules and access related rules are designed for each access operation of group based OSNs' information sharing. The security of oGBAC model is analyzed using formal methods. To demonstrate the usability of the oGBAC model, we implement the model with the Comparative Attribute-Based Encryption (CCP-CABE), and analyze the security and efficiency of the implemented system to prove the effectiveness of the implemented system.**

***Index Terms*—privacy protection, access control, social networks, attribute-based encryption, information flow.**

I. Introduction

ONLINE Social Networks (OSNs), such as Twitter, Flickr, Facebook, MySpace, LinkedIn and WeChat, have developed a huge number of Internet users over the past years. OSN users contact each other for various purposes including friendship, entertainment, social experience and knowledge sharing. In order to help users protect their private personal information, current OSNs often adopt a simple user-centric policy management mechanism which requires users to specify policies for managing access to their posted resources [1]. However, this mechanism mainly focuses on attaching attributes and policies to an object, and hence the access control is unable to be fine-grained. When the ownership relation of an object is changed because of operations such as reposting, the information may flow from one user to another user who should not have access to the object [2].

The insecure information flow comes from the interrelated OSNs' nodes which are connected by weak relationship in the virtual environment. If Alice becomes a friend of Bob in an OSN, Alice can access information from Bob's friends. Bob's friends may repost Bob's early posts some of which Bob does not want Alice to know. The access control models used in current OSNs cannot make users determine which friends can access what resources during what time accurately and dynamically. As a result, sensitive information may be disseminated to unwanted users. Furthermore, if a user does not realize the importance of protecting privacy and makes the private personal information known to the public, the information may be easily disseminated to malicious attackers, which may further cause illegal or criminal results.

Users in an OSN usually form organizations with relatively stable members, clear social tags and boundaries, called "groups" [3]. Recently, with more and more people joining in OSNs, it is popular to share information and ideas within and among OSN groups. For example, it is reported that every day, about 2,300,000 new groups are created in WeChat [4], the most popular OSN in China. It is very common for ordinary people in China to share various types of information through such kind of OSN groups. For OSNs, one user may have dozens to hundreds of groups, and the new shared information (like a picture or news) posted by a member of one group can be reposted to another group by members who belong to both groups. Due to the huge number of the members of a group as well as the total number of groups, we believe this kind of inter-group OSNs information sharing and dissemination can sharply enlarge the risk of information disclosure. Such risk may prevent the users from sharing useful information in OSN groups [5], defeating the purpose of OSNs in sharing information, exchanging the experiences, strengthening friendships and flourishing the culture of human beings. So, to encourage more users to share information in OSN groups, it is important to develop a more fine-grained privacy protection policy for the inter-group OSN information sharing.

In this paper, we propose a formal Group-Based Access Control (oGBAC) framework for preventing security violation and privacy disclosure when sharing information within or among groups in OSNs. Our framework extends the group-centric Secure Information Sharing (g-SIS) models [6]. The

Donghui Hu and Yuqi Fan are with the College of Computer Science and Information Engineering, Hefei university of technology, Hefei 230009 China (e-mail: hudh@hfut.edu.cn; yuqi.fan@hfut.edu.cn).

Chunya is with State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China (e-mail: hcy_0323@163.com).

Xintao Wu is with Department of Computer Science and Computer Engineering, University of Arkansas, Fayetteville, AR 72701 USA (e-mail: xintaowu@uark.edu).

g-SIS models are information sharing models where the rules of granting authorization are based on groups other than individuals. Thus, the g-SIS models are suitable for controlling inter-group information sharing. However, there are some limitations for the g-SIS models to be directly applied to the OSN group information sharing. First, the g-SIS models do not consider that there may exist untrusted operations such as reposting in OSNs. Second, the security levels of all groups in the g-SIS models are the same, so that users cannot define different relationship and status. Third, the g-SIS models cannot control the information flow among different groups. If a user is in two different groups at the same time, he can repost the resources from one group to the other, which may result in insecure information flow and disclose private information to some untrusted users.

In the proposed oGBAC, we impose some restrictions to the information flow among groups to ensure that operations cannot incur privacy disclosure when sharing information among friends in OSNs. In view of characteristics of OSNs and the requirements of secure information flow, the oGBAC incorporates some ideas from the Attribute-Based Access Control (ABAC) to develop information flow-based rules using relationship among attributes (such as tags, time and security levels) of objects and subjects in OSNs. We implement the model with the Comparative Attribute-Based Encryption (CCP-CABE), and achieve the objective of each oGBAC policy with cryptography algorithms. It is very hard to attack the CCP-CABE, so the security condition of each oGBAC policy is assured in the implementation. We analyze the security of the oGBAC model with information flow theorems, and also analyze the security and efficiency of the CCP-CABE based implementation system.

The rest of this paper is structured as follows. We begin with a discussion of related work about privacy protection and access control models of OSNs in Section II. In Section III, we describe our oGBAC conceptual framework. In Section IV, we describe the detailed polices and access control rules of our oGBAC framework. In Section V, we describe the implementation of the oGBAC model with CCP-CABE. In Section VI, we analyze the security and usability of the oGBAC model, and also analyze the security and efficiency of the CCP-CABE based implementation system. We draw our conclusions and point out future work in Section VII.

II. Related work

We describe the related work of privacy protection in OSNs from three perspectives.

A. HCI-based methods

The first perspective is from the Human-Computer Interaction (HCI). Ahern et al. [7] studied the factors that affect the privacy decision in mobile and online photo sharing. Miller and Edwards [8] investigated users' orientation toward image privacy and found that users are more concerned with their own privacy than others and the users also worry about their kids' privacy. Klemperer et al. [9] explored whether keywords and captions with which users tag their photos can be used to help users intuitively create and maintain access control policies. Hu et al. [5] conducted a questionnaire-based study with 183 Wechat users and found that there is a partial order based on either privacy levels or attribute tags for image sharing in social networks. Fogues et al. [10] conducted a study of 34 Facebook users and found that the access control policies that take into account groups, tags, and tie strength of relationships can be managed more easily than current approaches. The studies from the perspective of HCI can provide useful considerations and suggestions for the design of privacy policies of OSNs.

B. Machine learning-based methods

Some researchers used the machine learning-based methods to predict the privacy preferences and analyzing the sematic similarity among tags. Vyas et al. [11] studied the use of annotation data to predict users' privacy preferences and automatically derived policies for shared content based on a semantic analysis of tags. Squicciarini et al. [12] developed an Adaptive Policy Prediction (A3P) system to help users compose privacy settings for their online images. Zerr et al. [13, 14] developed a search engine for privacy oriented image search as well as a web service for supporting user decisions regarding image privacy. Squicciarini et al. [15] conducted a comprehensive study on large-scale image privacy classification which includes not only simple privacy classification models based on binary labels, but also models for more complex and multi-facet privacy settings. Hu et al. [16] proposed a framework of calculating the privacy level of a digital image based on perceptual hashing and semantic privacy rules. The calculated privacy level can be fed to the user before he/she shares the image to OSNs, and also can be used as a valuable reference for the design of more fine-grained OSNs' access control systems.

C. Security model-based methods

For the third perspective, researchers used the formal, semi-formal, or informal model to describe the access rules for sharing information in OSNs. Carminati et al. [17] proposed an extensible OSN access control model based on semantic web technologies. The main idea is to encode social network-related information by means of ontology. Squicciarini et al. [1] applied the game theory to model the problem of collective enforcement of privacy policies on shared data in OSNs. Shehab et al. [18] presented an access control framework to prevent the OSN applications from accessing user's personal data. Li et al. [19] investigated the effectiveness of privacy control mechanisms against privacy leakage from the perspective of information flow. The analysis revealed that the existing privacy control mechanisms do not protect the flow of personal information effectively. The authors also provided suggestions or remedies for OSN users to mitigate the risk of involuntary information leakage in OSNs. The above work mainly focused on the user-centric OSNs information sharing. However, the information disclosure within or among groups of OSNs did not get sufficient attention. Damen et al. [20] investigated the impact of privacy settings and tagging on the visibility of

information in Facebook, and analyzed the privacy leakage in the typical scenarios when using tagging among Facebook users. Krishnan et al. [6, 21] provided Group-Centric Secure Information-Sharing (g-SIS) models for isolated groups. The main objective of the g-SIS models is to bring users and information together in a group so as to facilitate sharing and collaboration within the group. As mentioned in [21], the metaphors "secure meeting room" and "subscription service" characterized the g-SIS approach. However, the g-SIS models are designed for isolated groups, and lack of considering the information flow among different groups. Thus, they cannot be directly applied to privacy protection of OSNs groups, where the groups are open and connected together due to the social network ethics of openness. Hinrichs et al. [22] proposed the Tag-Based Authorization (TBA) model, where relatively untrained users can choose descriptive tags for the systems subjects and objects, and security experts can write logical policies to define access authorizations using combinations of those tags. Etalle et al. [23] presented an extension of TBA that enables policy administration in distributed systems and resolves the issues in the TBA model such as tag verification and revocation.

Access control models for general computer or network application systems have been extensively studied. Bell and LaPadula [24] developed lattice-based access control models to deal with information flow in computer systems for the confidentiality requirement. Denning [25] investigated the mechanisms that guarantee secure information flow in a computer system via the lattice model. Sandhu [26] thought that though the lattice-based access control models are developed for the defense sector, the models can be used in most circumstances where information flow is critical. The Role-Based Access Control (RBAC) models [27] are widely used in the computer and network application systems, owning to their good attribute of customizable managements. However, RBAC has the weakness of leading to role explosion in the large business systems. Attribute-Based Access Control (ABAC) [28–30] is a logical access control model that controls access to objects by evaluating rules against the attributes of the entities (subject and object), actions and the environment relevant to a request. However, ABAC is in principal considered for application of computer or network systems features and cannot directly be applied to a social network.

Attribute-based encryption (ABE) implements ABAC in cryptography algorithms and provides secure and fine-grained access control for sharing data in an open environment. Sahai et al. [31] proposed a fuzzy identity-based encryption, which is the first approach of ABE. Soon after, more general ABE schemes were proposed, and they can be divided into the following two types: key-policy attribute-based encryption (KP-ABE) [32], and ciphertext-policy attribute-based encryption (CP-ABE) [33]. The main difference between the two schemes is that in KP-ABE the ciphertext is associated with a set of attributes, and its private key is associated with a monotonic access structure like a tree, which describes the user's identity; while in CP-ABE, the ciphertext is created with an access structure, which specifies the encryption policy. The private key is generated according to the user's attributes. This feature makes CP-ABE more suitable for access control applications. However, in the aforementioned ABE, the numerical comparison is time consuming. In the traditional CP-ABE scheme, all the attribute values are connected in the attribute range with OR. The time it takes increases with the range of attributes. The Comparative Attribute-Based Encryption (CCP-CABE) [34] scheme made a lot of improvements while supporting complete comparison relations, $<$, $>$, $\leq$ and $\geq$, in policy specification. Compared with CP-ABE, CCP-CABE not only has an advantage in computational complexity, but also has a good performance in storage. Regardless of the number of attributes in the access policy, the ciphertext length of the CCP-CABE is fixed. Therefore, we use CCP-CABE to implement the detailed policies given in the oGBAC.

III. Conceptual framework of oGBAC

We first introduce some basic concepts of the oGBAC framework. Different from traditional models that consist of three elements, i.e., subjects, objects, and operations, and two types of operations, i.e., read and write, the oGBAC model involves two additional elements, i.e., group and effective time, and one additional operation, i.e., repost. Formally, the oGBAC is represented by a quintuple of (U, O, G, P, T_L), where $U = \{u_1, u_2, \cdots, u_{n_u}\}$ represents the subjects (users of OSNs), $O = \{o_1, o_2, \cdots, o_{n_o}\}$ represents the objects (information shared by the users of OSNs), $G = \{g_1, g_2, \cdots, g_{n_g}\}$ represents the groups, $P = \{p_1, p_2, \cdots, p_{n_p}\}$ represents the operations, and $T_L = \{T_{L1}, T_{L2}, \cdots, T_{Ln_{TL}}\}$ represents the time intervals for ordered elements.

As mentioned in the Introduction, the "groups" are explicit social organizations with relatively stable members, clear social tags and boundaries. The relation type between the users and the groups is "many-to-many", which means, a user can belong to multiple groups, and a group can contain multiple users. The relation type between the objects and the groups is also "many-to-many". However, when an object is spread into different groups, a unique version should be maintained. We identify and retrieve each subject or object by a unique $Identity\ ID$ that is assigned to it.

Definition 1. *(Identity ID) Each subject (object) has a unique identity ID. Identity IDs of all subjects are represented as* $SIDS = \{sid_1, sid_2, \cdots, sid_{n_{sid}}\}$ *and identity IDs of all objects are represented as* $OIDS = \{oid_1, oid_2, \cdots, oid_{n_{oids}}\}$*, where* n_{sid} *and* n_{oids} *represent the numbers of subjects and objects in the model, respectively.*

In the oGBAC model, the security polices are developed based on the partial ordering of attributes of users, groups and objects. Three types of important attributes, namely security level, attribute tag and effective time period, are considered in our model.

Definition 2. *(Security Level) Each subject (object) is assigned a security level. Security levels can be expressed as* $SCLS = \{scl_1, scl_2, \cdots, scl_{n_{scls}}\}$*, where* n_{scls} *is the number of the security levels,* $SL_{mim} \leqslant scl_i \leqslant SL_{max}$*,* SL_{mim} *and* SL_{max} *are the minimal and the maximal security levels, respectively. The partial ordering relation between security*

levels can be expressed as $scl_i \leqslant scl_j$. $(SCLS, \leqslant)$ forms a bounded lattice.

Definition 3. *(Semantic tag) Each subject (object) is assigned a subset of semantic tags. Semantic tags can be expressed as $SCTS = \{sct_1, sct_2, \cdots, sct_{n_{scts}}\}$, where n_{scts} is the number of the semantic tags, $ST_{mim} \preceq sct_i \preceq ST_{max}$, ST_{mim} and ST_{max} are the minimal and the maximal semantic tags (in their partial ordering), respectively. The partial ordering relation between semantic tags can be expressed as $sct_i \preceq sct_j$. $(SCTS, \preceq)$ forms a bounded lattice.*

Definition 4. *(Effective Time Period) Each subject (object) is assigned an effective time period, which means during that time period a subject (object) can launch (or accept) an access. The domains of effective time periods for all subjects and objects are represented as $T_L = \{T_{L1}, T_{L2}, \cdots, T_{Ln_{TL}}\}$. Each T_{Li}=$[t_{si}, t_{ei}]$ represents the interval from the starting time t_{si} to the ending time t_{ei}. The partial ordering relation between effective time period can be expressed as $T_{Li} \subseteq T_{Lj}$. $(T_L, \subseteq)$ forms a bounded lattice.*

The attributes with values and partial ordering can be expressed abstractly as a triplet $< attr, val, R(val) >$, where $attr$ stands for attribute, val represents an attribute value in its domain VAL, and $R(val)$ represents relationship between different values of attributes. In general, there are two types of attributes: discrete and continuous. For the discrete attribute, attribute values generally has relationship on countable natural number; this relationship can be represented as $\{(val_1 \sim val_2)|val_1, val_2 \in VAL, \sim \in \{=, \neq, \leq, \geq, <, >\}\}$; for the continuous attribute, attribute values can be set-valued or atomic-valued. Set-valued attributes contain more than one atomic values and atomic-valued attributes contain only one atomic value. Attributes can be compared to constants or compared with each other, enabling relation-based access control. For attributes with continuous values, their relationships generally are partial ordering, such as inclusion relation, precedence relation and inheritance relation, which can be represented as $\{(val_1 \sim val_2)|val_1, val_2 \in VAL, \sim \in \{=, \neq, \preceq, \succeq, \prec, \succ\}\}$, where $val_1 \preceq val_2$ represents that val_1 is not larger than val_2 in partial ordering. For the three types of attributes in our model, *Security Level* and *Semantic Tag* are discrete, while *Effective Time Period* are continuous. To define the reasonable range of attributes values, we introduce the concept of attribute predicate ap which can be written as $attr \propto val$, $\propto \in \{=, \neq, \leq, \geq, \preceq, \succeq\}$.

For a user u, we use $u.SCTS$, $u.sct$, $u.SCLS$ and $u.scl$ to represent user u's set of semantic tags, a semantic tag, set of security levels and a security level, respectively. Without loss of generality, we assume the semantic tags and security levels are atomic-valued attributes. The partial ordering relation of semantic tags is different from that of security level and effective time period. We use the operators of "$\preceq$", "$\leqslant$" and "$\subseteq$" to express the partial ordering relation between semantic tags on $SCTS$, the partial ordering relation between security levels on $SCLS$, and the inclusion relation between effective times on time interval of T_L, respectively. The three attribute sets of $SCTS$, $SCLS$ and T_L, along with the operators of "$\preceq$", "$\leqslant$" and "$\subseteq$", form bounded lattices for each attribute, respectively. In Section VI-A, we will prove that bounded lattices can be formed in the combination of the attribute sets and their corresponding operators. Fig. 1 shows illustrative examples for three types of lattices based on each of the three attribute sets.

Example 1. *Fig. 1 (a) illustrates the partial ordering relation among security levels on $SCLS = \{L_1, L_2, L_3, L_4\}$, where $L_1 \leqslant L_2$, $L_2 \leqslant L_3$, and $L_3 \leqslant L_4$. For example, in the real OSNs applications, we can divide the group of friends into four levels, like $\{common\ fries,\ close\ friends,\ best\ friends,\ girl\ friends\ (or\ boy\ friends)\}$, with each level values 0, 1, 2, and 3, respectively.*

In the oGBAC model, we tend to use the security level to set up the partial ordering of the subjects and objects in a group, while using the semantic tags to set up the partial ordering of the subjects and objects among different groups.

Example 2. *Fig. 1 (b) illustrates the partial ordering relation between semantic tags on $SCTS$= {"life", "normal", "travel", "complaints", "knowledge", "mood", "status"}, where "life" $\preceq$ "normal", "life" $\preceq$ "travel", "life" $\preceq$ "complaints", "normal" $\preceq$ "knowledge", "travel" $\preceq$ "knowledge", "complaints" $\preceq$ "mood", "knowledge" $\preceq$ "status" and "mood" $\preceq$ "status".*

Example 3. *Fig. 1 (c) illustrates the partial ordering relation between effective times on time interval of $T_L = \{T_{L1}, T_{L2}, T_{L3}, T_{L4}, T_{L5}\}$, where $T_{L1} \subseteq T_{L2}$, $T_{L1} \subseteq T_{L3}$, $T_{L2} \subseteq T_{L4}$, $T_{L3} \subseteq T_{L4}$ and $T_{L4} \subseteq T_{L5}$. Assuming days as the time units used in the OSNs system, then a detailed example satisfying the partial ordering in Figure 1 (c) could be T_{L1}=[01/01/2018,12/30/2019], T_{L2}=[01/01/2016,12/30/2019], T_{L3}=[01/01/2017,12/30/2020], T_{L4}=[01/01/2010,12/30/2050] and T_{L5}=[01/01/1990,12/30/2100].*

In the oGBAC model, the effective time period is an important attribute used to design the privacy policies. The user can access an object only when the access time is in the effective time period of the user's group and also in the effective time period of the object. For example, if the effective time period of a group G is '[01/01/2017,12/30/2027]', and the effective time period of an object O is '[01/01/2018,12/30/2026]', then, at the time of '05/05/2018', a user in the group G can access the object O. Along with the attribute tags[1], our model can provide dynamic and fine-grained access control for the OSNs information sharing.

After describing the definitions of the security levels, semantic tags and the effective time period, we can define the operations in the oGBAC model.

Definition 5. *(Operation Set) Operations OPS in the oGBAC model can mainly be divided into two categories: administrative operations and access operations. The administrative*

[1]If not explicitly stated, in the following sections, we use attribute tags to stand for both semantic tags and security levels.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2018.2875697, IEEE Transactions on Dependable and Secure Computing

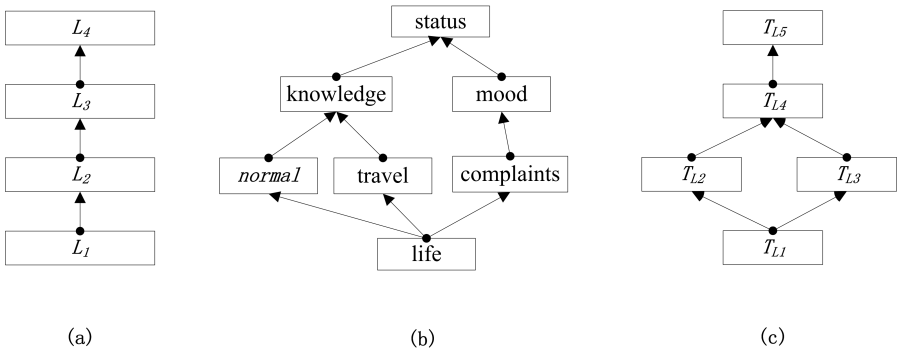


Fig. 1. Examples of three kinds of partial ordering relationships

TABLE I
THE ADMINISTRATIVE AND ACCESS OPERATIONS IN OGBAC MODEL

Operations	Interpretation
$create(u, g, Tag_g, Lev_g, t_0,)$	User u creates a group g with semantic tag of Tag_g and security level of Lev_g at time t_0;
$join(u, u', g, t)$	User u invites a user u' to join a group g at time t;
$remove(u, u', g, t)$	User u removes a user u' from a group g at time t;
$drop(u, g, t)$	User u quits a group g at time t.
$post(u, o, g, Teg_o, Lev_o, t)$	User u adds object o with semantic tag Teg_o and security level Lev_o to a group g at time t;
$read(u, o, g, t)$	User u reads object o in a group g at time t;
$delete(u, o, g, t)$	User u removes o from a group g at time t;
$write(u, o, v, g, t)$	User u writes or comments a version v of object o in a group g at time t;
$repost(u, o, v, g, g', t)$	User u reposts an object o from its source group g to destination group g' with new version of v at time t.

operations include $create$, $join$, $remove$ and $drop$ (a group). The access operations include $post$, $delete$, $read$, $write$ and $repost$ (an object in a group or among different groups).

Table I summarizes the four basic administrative operations and five basic access operations of the oGBAC model. Each operation is carried out in the environments of OSNs' groups. Though the attribute tags (of users, objects, and/or groups) are implicit in the formulas of the operations, they play an important role in determining whether the access operations can be approved. Detailed constraints of each policy related to the operations will be described in the next section. The key idea of the polices is to ensure a secure information flow according to the partial ordering relation among the attribute tags. The concept of the information flow is given as follows.

Definition 6. *(Information Flow) The information flow policy can be written as a triplet $< SK, \rightarrow, \odot >$, where SK is a set of attribute tags which express the property of elements, $\rightarrow$ is a binary can-flow relation on $SK \times SK$, and $\odot : SK \times SK \rightarrow SK$ is a binary composite operator. All the three components of an information flow policy are fixed and do not change over time. This definition allows objects to be created and destroyed dynamically while attribute tags cannot be created or destroyed dynamically. Based on this definition, information can flow from attribute tag A to attribute tag B under a given policy if and only if $A \rightarrow B$. $\odot$ is a binary composite operator which controls the flow of information based on its partial ordering relation. Similarly, $A \odot B = C$ means that the object which contains information from attribute tags of A and B should be labeled with attribute tag C.*

Information flow occurs when users access objects through certain operations. We use the methods of information flow to ensure that private information will not be accessed by users who should not access it in the open environment of OSNs, and hence to protect the privacy of the original information. By using the information flow policies, we can observe how the information flows from one group to another. If user u reads object o in a group, information will flow from o to u. If user u reposts object o from the object of u in a original group g to the version of v in a destination group g', information flows from o to v, and also results in an information flow from group g to group g'. If user u comments (writes) object o and generates version v, then information flows from u to v.

Finally, we give the definitions of the access control rule (or policy) of oGBAC as follows.

Definition 7. *(Access Control Rule Set) An access control rule set (or policy set) PLS defines all the access control rules used in an access control system. Each access control rule $PL_i \in PLS$ $(i = 1, 2, \cdots, n_{pls})$ describes in which condition a user u in a group g can access an object o with a given access operation of op at time t, where n_{pls} is the number of the total access control rules of the system. The procedure of access control can be defined as a judging function PL_i : $accept \leftarrow condition(u, g, o, op, t)$.*

The condition is a logical expression with sub conditions connected by the operators $\wedge$, $\vee$, and so on. Note that in the model, we only consider the judging function with the result of accept, which means the default access rule of the model is deny. In other words, if the condition of one of the n_{pls}

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2018.2875697, IEEE Transactions on Dependable and Secure Computing

rules is satisfied, then the access is accepted; if none of the n_{pls} rules is satisfied, the access is denied.

After the access, some access operations will change the attributes or relationship of the related subjects (users or groups) and objects, so we use the PL_{rsi} $(i = 1, 2, \cdots, n_{pls})$ to describe the results of the changes.

For each operation, we define an access control rule (or policy). In the next section, we will describe each policy. In section VI-A, we will analyze the security of the proposed policies, and the consistence and security of combining the different policies.

IV. Privacy policies

In this section, we divide access control rules in OSNs into two categories. The first category contains the administration related policies and the second category contains the access related policies.

For the convenience of description, in the following policies, we use U_u, G_u, U_g, O_g to denote all the users who have (direct) relationship with user u, all the groups that are created by the user u, all the users in a group g and all the objects that are contained in a group g ; use $o.soid$, $g.soid$ and $v.ooid$ to denote the original owner of an object, the original owner of a group and the original object of a version v, respectively. After an object o is posted in a group g, it may be reposed to other groups and form new versions. The new versions may be reposted again. To record and track the relationship of the original object and all of its versions, we introduce a tree tr, and use $create\_tree(tr, root, o.id)$, $add\_child\_tree(tr, root\_id, node_1, v_1.id, node_2, v_2.id)$ and $delete\_tree(tr, root\_id)$ to denote creating a tree tr with root's id value $o.id$, adding a child $node_2$ with key value $v_2.id$ to a $node_1$ with key value $v_1.id$ in the tree tr with root of $root\_id$, and deleting the tree tr with key value of the root is $o.id$, respectively.

A. Administration related rules

Four administration related rules, which give the conditions and results of the group related administrative operations, are described as follows.

Policy 1. *(policy of create) At time t_0, a user u can create a group with semantic tag of Tag_g and security level of Lev_g. Tag_g and Lev_g are constrained in the least upper and greatest lower bounds of partially ordered sets $SCTS$ and $SCLS$, respectively. After the operation, the G_u is enlarged by adding the new created group g. To the new created group g, the user can set the security level and the semantic tag of the group. The effective time period of the group starts from t_0. The policy can be written as:*

$$\begin{aligned} PL_1: & \ accept \leftarrow (op = create(u, g, Tag_g, Lev_g, t_0)) \\ & \wedge (g.gct = Tag_g) \wedge (g.gcl = Lev_g) \\ & \wedge (ST_{\min} \le g.sct \le ST_{\max}) \\ & \wedge (SL_{\min} \le g.scl \le SL_{\max}) \\ PL_{rs1}: & \ g \in G_u, g.soid = u, \\ & u.SCTS = u.SCTS + \{g.sct\}, g.t_L = [g.t_0, +\infty]. \end{aligned}$$

Note that in the above policy, the initial values of Tag_g and Lev_g are set by the users (or recommended by the system and finally decided by the users). After the operation, we assume the end time of the effective time period of the new created group is $+\infty$. In the real applications, the owner (or administrator) can set the value of the end time according to the requirement of privacy consideration.

Policy 2. *(policy of join) Assume user u is the owner of group g, and user u' is a direct friend. At time t, the user u can invite user u' to join group g. After the operation, U_g contains the new user u'. The security level of u' should be set to no lower than the security level of the group g (otherwise u' cannot access the object in the group); at the same time, user u' automatically gets the new semantic tag of group g. The policy can be written as:*

$$\begin{aligned} PL_2: & \ accept \leftarrow (op = join(u, u', g, t)) \wedge (g.soid = u) \\ & \wedge (u' \in U_u) \\ PL_{rs2}: & \ u' \in U_g, g.scl \le u'scl, \\ & u'.SCTS = u'.SCTS + \{g.sct\} \end{aligned}$$

In the real world of OSNs' applications, the users may create many groups for varied purposes. When the group owner finds some people in the group seldom interact with each other or some temporary groups are not needed, then for the convenience of management, the owner can drop those people or groups with the following policy.

Policy 3. *(policy of remove) Assume user u is the owner of group g, and u' is in group g. At any time t, user u can remove the user u' from the group g. After the operation, u' does not belong to U_g, and the semantic tag of g should be removed from the set of user u''s semantic tags. The policy can be written as:*

$$\begin{aligned} PL_3: & \ accept \leftarrow (op = remove(u, u', g, t)) \wedge (g.soid = u) \\ & \wedge (g.sct \in u'.SCTS) \\ PL_{rs3}: & \ u' \notin U_g, u'.SCTS = u'.SCTS - \{g.sct\} \end{aligned}$$

In the real world of OSNs' applications, the users of OSNs may create many groups for varied purposes. But one day the owner may find some people in some groups seldom interact with each other or some temporary groups have finished their task, then for the convenience of management, the owner can dismiss or drop the group with the following policy.

Policy 4. *(policy of drop) Assume user u is the owner of group g. At any time t, the subject can drop a group. After the operation, each user u' in the group should be removed from the group; group g is not in the list of the groups that owned by user u, and the end time of the effective time period of the group is t. The policy can be written as:*

$$\begin{aligned} PL_4: & \ accept \leftarrow (op = drop(u, g, t)) \wedge (g.soid = u) \\ PL_{rs4}: & \ (for\ all\ u' \in U_g, remove(u, u', g)), g \notin G_u, \\ & g.t_L = [g.t_0, g.t] \end{aligned}$$

Note that in the above two policies of join and remove, the condition of "user u is the owner of a group g" may be

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2018.2875697, IEEE Transactions on Dependable and Secure Computing

changed in the real-world OSNs. For example, in some OSNs, the condition may be relaxed to "user u is the administrator of group g", and there may be more than one administrator in a group. Also note that in this paper we do not provide the administrative policies for the management of attributes such as security level, semantic tag, etc. For this topic, users can learn more from other papers such as [23].

B. Access related rules

Five access related rules are given for the access operations within an OSN group or between two different OSN groups.

Policy 5. *(policy of post) At time t, user u can post object o with semantic tag of Tag_o and security level of Lev_o to group g. Tag_o and Lev_o are constrained in the least upper and greatest lower bounds of partially ordered sets $SCTS$ and $SCLS$, respectively. The additional requirement is that the semantic tag of the group is contained in the set of the user's semantic tags, the security level of user u is not lower than the security level of group g, and the current time t is in the effective time period of group g. After the operation of post is performed, the object is included in the list of all the objects in group g, the semantic tag of group g is added to the set of the semantic tags of object o, the security level of object o is set to no lower than the security level of group g, the owner of the object is the posting user u, the original object of o is o itself (that is to say, the initial value of $ooid$ is o itself, and note that the $ooid$ may be passed to the new versions of the object by the $repost$ operation), the effective time period of the posted object is from the posting time t to the end time of the effective time period of group g, and a tree tr with the key of the root $o.id$ is created. The policy of post can be written as:*

$$\begin{aligned} PL_5 : accept \leftarrow (op = post(u, o, g, Teg_o, Lev_o, t)) \\ \wedge(o.sct = Tag_o) \wedge (o.scl = Lev_o) \\ \wedge(ST_{\min} \le o.sct \le ST_{\max}) \\ \wedge(SL_{\min} \le o.scl \le SL_{\max}) \\ \wedge(g.sct \in u.SCTS) \\ \wedge(g.scl \le u.scl) \wedge (t \subseteq u.t_L) \\ PL_{rs5} : o \in O_g, o.SCTS = o.SCTS + \{g.sct\}, \\ g.scl \le o.scl, o.soid = u, o.ooid = o, \\ o.t_L = [t, g.t_L.t_e], create\_tree(tr, root, o.id) \end{aligned}$$

Note that in the above policy, Tag_o and Lev_o are set by the users (or recommended by the system and finally decided by the users). In the condition of the policy, the security level of object o is set to Lev_o, while in the result of the policy, it is set to no lower than the security level of group g, which means it may be adjusted by the policy implementation system (for example the initial value Lev_o is a relative value and is adjusted by the policy according to the absolute value of $g.scl$). In Section VI-E, we will discuss how the initial values Tag_o and Lev_o, as well as the initial values Tag_g and Lev_g in Policy 1, are defined in the real-world applications.

Policy 6. *(policy of delete) At time t, user u can delete object o from group g. The requirement to be satisfied is that the owner of the object is u and the object o has already been in group g at time t. After the operation, object o is no longer in O_g, the effective time period of the object ends at time t, the semantic tag of the group g is removed from the set of the semantic tags of object o, and all of the new versions (copies) of the object o in the tree of $tr_{o.ooid}$, which is formed by the previous $post$ and $repost$ operations, are deleted. The policy of delete can be written as:*

$$\begin{aligned} PL_6 : accept \leftarrow (op = (delete(u, o, g, t)) \wedge (o.soid = u) \\ \wedge(o \in O_g) \wedge (t \subseteq o.t_L)) \\ PL_{rs6} : o \notin O_g, o.t_L = [o.t_L.t_s, t], \\ o.SCTS = o.SCTS - \{g.sct\}, \\ (for\ all\ v \in tr_{o.ooid}, delete(v)) \end{aligned}$$

Policy 7. *(policy of read) At time t, user u' wants to read an object o in group g. The requirement to be satisfied is that user u' and object o should be in the same group g at the time t, the security level of the user u' should not be lower than the security level of the object o, the semantic tag of g should be contained in both the set of semantic tags of o and the set of the semantic tags of u', and both the effective time period of user u' and the effective time period of object o should be valid at access time t. After the operation of read is completed, the information flows from the object o to user u'. The policy of read can be written as:*

$$\begin{aligned} PL_7 : accept \leftarrow (op = read(u', o, g, t) \wedge (o \in O_g) \wedge (u' \in U_g) \\ \wedge(o.scl \le u'.scl) \wedge (g.sct \in (o.SCTS \cap u'.SCTS)) \\ \wedge(t \in u'.t_L \cap o.t_L) \\ PL_{rs7} : o \rightarrow u' \end{aligned}$$

Policy 8. *(policy of write) At time t, user u' wants to write object o in group g. The policy requires that the information can flow from object o to user u', which means the requirement is the same as the policy of read. After the operation of write is completed, the new version of object v is created, the security level of the new version v is set to no lower than the security level of object o, the semantic tag of the new version v is equal to that of group g, the effective time period of the new version v is set to from the current time t to the end time of the effective time period of object o. The policy of read can be written as:*

$$\begin{aligned} PL_8 : accept \leftarrow (op = write(u', o, v, g, t) \wedge (\exists(o \rightarrow u')) \\ PL_{rs8} : v \in O_g, o.scl \le v.scl, v.sct = g.sct, v.t_L = [t, o.t_L.te] \end{aligned}$$

Policy 9. *(policy of repost) At time t, user u' can repost object o from group g to another group g' and creates a new version v. It requires that u' belongs to both the group of g and the group of g', the semantic tag of g' is greater than the semantic tag of g (in its lattice based partial ordering relation), and user u' can read object o at time t. After the operation of repost is completed, user u' creates a new version v. The new created version v belongs to the object set of group g', the security level of v is equal to the security level of the original owner of object o, the semantic tag of v is equal to the semantic tag of g', the effective time period of object v is from the current access time t to the minimum of the end time of effective time*

TABLE II
SOME BASIC NOTATIONS USED IN THE CCP-CABE IMPLEMENTATION SYSTEM

Notation	Description
$\mathbb{A}$, A_i	attribute set, and the ith attribute;
$\mathcal{L}_u$	user u's attribute ranges;
$\mathcal{P}$	data owners access control policy;
$[t_{i,a}, t_{i,b}]$	attribute range on attribute A_i possessed by a data user;
$[t_{i,j}, t_{i,k}]$	range constraint on attribute A_i defined by $\mathcal{P}$;
$\{\rho_i, \rho_i'\}$	bound values associated with $[t_{i,j}, t_{i,k}]$ depending on the range relation over A_i;
$\{\lambda_i, \mu_i\}$	used to distinguish attribute A from other attributes;
$\{\tau_u, r_u\}$	used to distinguish u from other users.

period of the object o and the end time of effective time period of the group g', and as a result, the new version of the object v_2 (with $v_2.id$) is also added to the tree tr (with root of $ooid$) as the child of its parent $node_1$ with key value of $v_1.id$. At the same time, there is an information flow from the group g to group g' after the operation. The policy of repost can be written as:

$$PL_9 : accept \leftarrow (op = repost(u', o, v, g, g', t)) \wedge (u' \in (U_g \cap Ug')) \wedge (g.sct \preceq g'.sct) \wedge (\exists (o \rightarrow u'))$$

$$PL_{rs9} : v.soid = o.soid, v.ooid = o.ooid, v \in O_{g'}, v.scl = (o.soid).scl, v.sct = g'.sct, v.t_L = [t, min\{o.t_L.t_e, g.t_L.t_e\}], add\_child\_tree(tr, o.ooid, node_1, v_1.id, node_2, v_2.id), g \rightarrow g'$$

Note that in the aforementioned three polices (policy of $read$, $write$ and $repost$), it is not necessary that user u' is the owner of object o, and the object o may not be the original object (it may have been reposted from other groups).

V. IMPLEMENTATION WITH CCP-CABE

A. Overview

In general, Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [33] provides algorithms for the data owners to create access policies by designating attribute constraints and embedding the data access policies into the ciphertext, such that any data user has to satisfy the corresponding attributes to access the data. Considering that in some cases, the access control policy takes the form of range constraints (e.g., $t_{i,a} \preceq A_i \preceq t_{i,b}$), literature [34] proposed a Comparative Attribute-Based Encryption (CCP-CABE) which can predefine different range intersection relationships on different attributes. The conditions of each access control policies proposed in Section IV are in the forms of range constraints, or can be rewritten into the form of range constraints (e.g., the condition "$o.scl \leq u'.scl$" in the policy of read can be rewritten as "$o.scl \leq u'.scl \leq SL_{max}$"). Moreover, CCP-CABE can achieve high efficiency with minimal communication overhead. Hence in this paper we use CCP-CABE to implement the policies of oGBAC. In the implementation, we take image sharing in OSNs as an example. The main idea of the implementation is that, the attributes of each user can be defined in the administrative policies, and then when an object is first posted in a group (by post operation), its access conditions for each other access operations (read, write, repost and delete) are defined in the forms of range constraints and stored into the ciphertext. Later, when a user's attributes satisfy one of these constraints, the user can access the object through the operation related to the constraints. Some basic notations used in our CCP-CABE implementation system are listed in Table II. The users can find the detailed explanation of the parameters and security analysis of CCP-CABE in [34].

In literature [34], considering the poor computing power at the client side in the cloud environment, the complex calculations are performed by third parties. Therefore, the encryption algorithms are divided into two parts, $EncdDelegate$ and $Encryption$, and the decryption algorithms are divided into two parts, $DecDelegate$ and $Decryption$. In our implementation, we assume the server takes all the computing tasks, so we do not need delegated algorithms or keys. At the same time, to improve the efficiency of access operations (including read, write, repost and delete), our system computes a header for each of the operation. The attributes (except time) for each operation are stored together as an access header $\mathcal{H}'_{\mathcal{P}} = \{\mathcal{H}_{\mathcal{P}_{read}}, \mathcal{H}_{\mathcal{P}_{write}}, \mathcal{H}_{\mathcal{P}_{repost}}, \mathcal{H}_{\mathcal{P}_{delete}}\}$. In addition, to control the attribute of time and provide fine-grained access control based on the time factor, our system computes a new file header $\mathcal{H}_T$ using CCP-CABE encryption algorithm to encrypt the $\mathcal{H}'_{\mathcal{P}}$. The time header $\mathcal{H}_T$ and the encrypted access header $\mathcal{H}'_{\mathcal{P}}$ together compose the whole header $\mathcal{H}_{\mathcal{P}}$. The relationship of $\mathcal{H}_{\mathcal{P}}$ and $\mathcal{H}'_{\mathcal{P}}$ can been seen from Fig. 2, where we give the whole file structure of the implementation system (the details of each part of the file system will be further described in the following $Encrypt(GP, MK, \mathcal{P})$ algorithm). The detailed calculation process is shown as follows.

1) Setup ($\mathbb{A}$, k): When a user registers to the system, the user completes the subject's attributes set (SAS) needed by the system. The system server uses the definition of the association settings in CCP-CABE to generate a global parameters GP and the master keys MK. The server first chooses a bilinear map system $\mathbb{S}_N = (N = pq, \mathbb{G}, \mathbb{G}_T, e(\cdot,\cdot))$, where $N = pq$ is the RSA modulus, p and q are two large primes. We get two cyclic groups $\mathbb{G}$ and $\mathbb{G}_T$ with composite order n, where $n = n's$, and $\mathbb{G}_{n'}$ and $\mathbb{G}_s$ are two subgroups of $\mathbb{G}$ with order n' and s respectively. The server selects the random generators $\varrho \in \mathbb{G}_s$, $w \in \mathbb{G}$ and $\varphi \in \mathbb{G}_{n'}$, and chooses $\lambda_i, \mu_i \in \mathbb{Z}_n^*$ for each attribute A_i and chooses a cryptographic hash function $H : \{0,1\}^* \rightarrow \mathbb{G}$ to convert a binary attribute string into a group element. Finally, the server keeps its master key $MK = (p, q, n', \alpha, \beta)$, where α, β are random number in $\mathbb{Z}_n^*$; and publishes the global parameters $GP = (\mathbb{S}, \varrho, h, \omega, \eta, e(\varrho,\omega)^\alpha, \varphi, \{\lambda_i, \mu_i\}_{A_i \in \mathbb{A}}, H(\cdot))$, where $h = w^\beta$, $\eta = \varrho^{\frac{1}{\beta}}$ and e denotes a computable bilinear map: $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$.

2) KeyGen (GP, MK, u, $\mathcal{L}_u$): When user u' invites a friend u to join group g (in the administration policy of join), u has attributes $\mathcal{L}_u = \{\lambda_i, \mu_i\}_{A_i \in \mathbb{A}}$, where $\mathbb{A}$ is consistent with the SAS mentioned above. The server generates two unique integers $\tau_u, r_u \in \mathbb{Z}$ to distinguish u from other users in the system. The private key of user u is based on the relevant information of the user. Consequently, the private key SK_u of

u can be computed as:

$$\begin{aligned} SK_u &= (D_0^{(u)}, D_1^{(u)}, D_2^{(u)}, DK_u) \\ &= (\varrho^{\frac{\alpha+\tau_u}{\beta}}, \varrho^{\tau_u}(H(A))^{r_u}, \omega^{r_u}, (v_{\mathcal{L}_u})^{r_u}), \end{aligned} \tag{1}$$

where $v_{\mathcal{L}_u}$ is calculated by Multi-Dimensional Range Derivation Function (MRDF) (which is an order-preserving cryptographic map defined in [34]), $v_{\mathcal{L}_u} = v_{[t_{i,a},t_{i,b}]_{A_i\in\mathbb{A}}} = \varphi^{\Pi_{A_i\in\mathbb{A}}\lambda_i^{t_{i,a}}\mu_i^{Z-t_{i,b}}} \in \mathbb{G}_{n'}$.

Afterwards, the keys are transmitted to user u through secure channels and stored locally. In the implementation, each user in a group has four different private keys for different operations. For example, the operation of read can be expressed as:

$$\begin{aligned} SK_{u_{read}} &= (D_{read,0}^{(u)}, D_{read,1}^{(u)}, D_{read,2}^{(u)}, DK_{u_{read}}) \\ &= (\varrho^{\frac{\alpha+\tau_u}{\beta}}, \varrho^{\tau_u}(H(A_{read}))^{r_u}, \omega^{r_u}, (v_{\mathcal{L}_{u_{read}}}^{r_u})), \end{aligned} \tag{2}$$

where $v_{\mathcal{L}_{u_{read}}}^{r_u}$ is defined by user's attributes in read.

3) Encrypt (GP, MK, $\mathcal{P}$): Users can upload pictures to group g. At the same time, the users have five access operations (post, read, write, repost and delete) to perform on the pictures in the group g. When the user posts a picture to a group g, the system sets the accessing policy of the picture according to the user's and the group's context information (attributes). The server calculates the file head of the read / write / repost / delete using the encryption algorithm, and stores it with the encrypted picture on the server.

We assume that the access policy of an attribute constraint (which is described in the condition of each policy in Section IV) is denoted by $\mathcal{P} = \{\rho_i, \rho_i'\}_{A_i\in\mathbb{A}}$. First, according to the accessing policy, the system computes $v_{\mathcal{P}}$ by using MRDF as:

$$v_{\mathcal{P}} = v_{\{\rho_i,\rho_i'\}_{A_i\in\mathbb{A}}} = \varphi^{\Pi_{A_i\in\mathbb{A}}\lambda_i^{\rho_i}\mu_i^{Z-\rho_i'}}. \tag{3}$$

Then the system computes $\varepsilon_X \in \mathbb{Z}_n$, $C_X = h^{\varepsilon_X}$, $E_{\varepsilon_X} = (v_{\mathcal{P}_\mathcal{X}} w)^{\varepsilon}$, $E'_{\varepsilon_X} = (H(A))^{\varepsilon_X}$ and a session key $e_{k_X} = e(\varrho^\alpha, \omega)^{\varepsilon_X}$ for every different header. We use the same a_k, but four different e_{k_X} ($X \in \{read, write, repost, delete\}$) when computing the four file headers. In other words, we choose four different ε_X randomly to get four different session keys e_{k_X}. To improve the efficiency of computation, the system first generates a random key a_k to encrypt the target picture and uses e_{k_X} to encrypt the random key a_k for every header. At last, it outputs the access header $\mathcal{H}'_\mathcal{P}$ by computing the four sub-headers of $\mathcal{H}_{\mathcal{P}_{read}}$, $\mathcal{H}_{\mathcal{P}_{write}}$, $\mathcal{H}_{\mathcal{P}_{repost}}$ and $\mathcal{H}_{\mathcal{P}_{delete}}$ in turn:

$$\mathcal{H}'_\mathcal{P} = (\mathcal{H}_{\mathcal{P}_{read}}, \mathcal{H}_{\mathcal{P}_{write}}, \mathcal{H}_{\mathcal{P}_{repost}}, \mathcal{H}_{\mathcal{P}_{delete}}), \tag{4}$$

where $\mathcal{H}_{\mathcal{P}_X} = (\mathbb{E}_{e_{k_X}}(a_k), h^{\varepsilon_X}, (v_{\mathcal{P}_\mathcal{X}} w)^{\varepsilon_X}, (H(A_X))^{\varepsilon_X})$, $X \in \{read, write, repost, delete\}$.

After that, the access header $\mathcal{H}'_\mathcal{P}$ is encrypted again with the attribute of the effective time. Following the similar way by using the CCP-CABE Encryption, a session key $e_{k'}$ and a random key $a_{k'}$ are generated. The server computes a new file header $\mathcal{H}_T$. The access header $\mathcal{H}'_\mathcal{P}$ is encrypted using $a_{k'}$ with AES encryption, where $a_{k'}$ is protected by $e_{k'}$, and $e_{k'}$ can be derived from $\mathcal{H}_T$ using the CCP-CABE Decrypt algorithm.

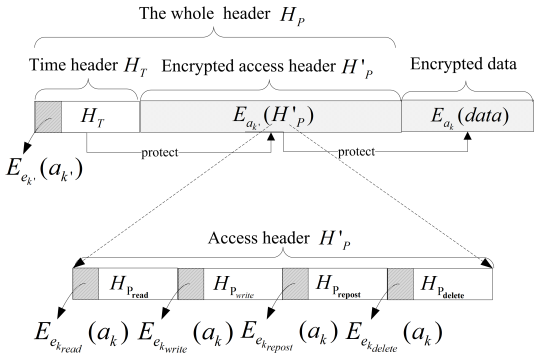


Fig. 2. The file structure in the implementation system

The time header and the encrypted object are stored in the server. The final file structure of the implementation system is shown in Fig. 2.

4) Decrypt ($\mathcal{H}_\mathcal{P}$, SK_u, $\mathcal{L}_u$, $\mathcal{P}$): In the first step, the server checks if the user's current attribute of time satisfies the constraint in time header $\mathcal{H}_T$. If so, it derives the session key $e_{k'}$ from the header, recovers the random key $a_{k'}$ by using the $e_{k'}$, and uses the $a_{k'}$ to decrypt the encrypted $\mathcal{H}'_\mathcal{P}$ and gets the $\mathcal{H}'_\mathcal{P}$ for the next computation.

The next step of decryption handles different operations (i.e., read, write, repost and delete) differently. According to the user's request, the server tries to decrypt the picture header using the CCP-CABE decryption algorithm. If it can output the random key a_k successfully, the user has the permission for his/her request; otherwise, the user has no permission for the request. Take the reading operation for example. If the user requests to read the object, the system finds the object's reading header $\mathcal{H}_{P_{read}}$. The calculation of CCP-CABE is as follows.

First, the server checks if $\mathcal{L}_{u_{read}}$ satisfies the access policy over all attributes. If yes, it computes $(v_{\mathcal{P}_{read}})^{r_u}$ from $(v_{\mathcal{L}_{u_{read}}})^{r_u}$.

The server also computes

$$\Gamma(\varepsilon_{read}) \leftarrow \frac{e(D_{read,1}^{(u)}, E_{\varepsilon_{read}})}{e((v_{\mathcal{P}_{read}}\omega)^{r_u}, E'_{\varepsilon_{read}})} \tag{5}$$

and

$$e_{k_{read}} = e(\varrho^\alpha, \omega)^{\varepsilon_{read}} = \frac{e(C_{read}, D_{read,0}^{(u)})}{\Gamma(\varepsilon_{read})}. \tag{6}$$

Finally, it outputs the secret key $e_{k_{read}}$. We can use $e_{k_{read}}$ to get the random key a_k and then use the a_k to decrypt the encrypted picture. If a user's attributes set cannot satisfy the access policy, the user's request is rejected.

More generally, the recovery of e_{k_X} ($X \in \{read, write, repost, delete\}$) can be obtained by using the following equations:

$$\Gamma(\varepsilon_X) \leftarrow \frac{e(D_{X,1}^{(u)}, E_{\varepsilon_X})}{e((v_{\mathcal{P}_X}\omega)^{r_u}, E'_{\varepsilon_X})} \tag{7}$$

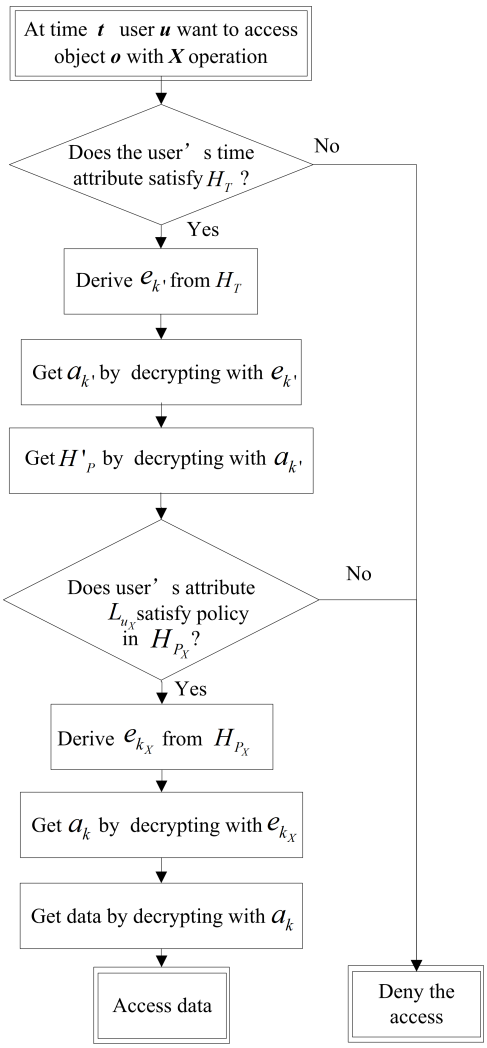


Fig. 3. The illustration of access procedure in the implemented system

and

$$e_{kX} = e(\varrho^{\alpha}, \omega)^{\varepsilon_X} = \frac{e(C_X, D_{X,0}^{(u)})}{\Gamma(\varepsilon_X)}. \quad (8)$$

The whole process of decryption (also is the process of access) in the implementation system is shown in Fig. 3.

B. Implementation of information flow policy

There are five operations that can be performed on the picture: post, read, write, repost and delete.

In the implementation, we do not strictly follow the detailed conditions of the policies described in Section IV. In fact, considering the efficiency, we only consider the main conditions and relax some of the other conditions. For example, each policy different time constraints. In the implementation, we simplify these conditions into "current access time t is in the effective time period of object o ($t \in o.T_L$)" and use a separate header, time header $\mathcal{H}_T$ to describe it. This kind of simplification and separation has an additional benefit, that is, if the time constraint is not satisfied, the access is denied directly without further comparing with the constraints in $\mathcal{H}'_{\mathcal{P}}$, hence it can improve the efficiency of the implementation system greatly. The related process can be learned from Fig. 3.

The implementation of each access operation is described as follows. Notice that in the following implementations, we suppose for a given object, the policies related to operations of read, write, repost and delete ($\mathcal{P}_{read}$, $\mathcal{P}_{write}$, $\mathcal{P}_{repost}$ and $\mathcal{P}_{delete}$) are defined in the post operation when the object is first posted in a group, while the policy of post itself is not considered.

1) Implementation of posting operations: When a user u wants to post an object o to a group g in the OSNs system, the system check if the user's attribute satisfies the access policy of post. If so, the object's security level ($o.scl$) and security tag ($o.sct$) are set to the initial values (Lev_0 and Teg_0) defined by the user, respectively. At the same time, the object's effective time ($o.T_L$) is set to from the current time to the group's expiration time. The object's owner id ($o.sid$) is set to the data owner's id in the system. According to the four accessing policies, the object is encrypted using CCP-CABE encryption algorithm for possible future access. The calculation process is as follows:

- Step 1: Preparation for access policy of read. The access policy of read can be expressed as $\mathcal{P}_{read} = (u.scl \in [o.scl, SL_{max}])\ AND\ (g.sct \in o.SCTS)$. Note that $(g.sct \in o.SCTS)$ is a relaxed condition compared with the original one described in Section IV, and it can be rewritten in the form of range constraint. If user u can satisfy the access policy of read, the information flows from o to u. We use the range of the user's security level and the range of the group's semantic tag to replace ρ_i and ρ'_i in the formula $v_p = v_{\{\rho_i,\rho'_i\}_{A_i \in \mathbb{A}}} = \varphi^{\Pi_{A_i \in \mathbb{A}} \lambda_i^{\rho_i} \mu_i^{Z-\rho'_i}}$. The v_p in $\mathcal{H}_{P_{read}}$ can be computed. Finally, we can get the encrypted reading header $\mathcal{H}_{\mathcal{P}_{read}}$.
- Step 2: Preparation for access policy of write. According to Policy 8, the requirement of operation write is that the information can flow from o to u. The access policy of write can be expressed the same as the access policy of read. Finally, we can get the encrypted writing header $\mathcal{H}_{\mathcal{P}_{write}}$, which is the same as $\mathcal{H}_{\mathcal{P}_{read}}$. So we only need to calculate the value of $\mathcal{H}_{\mathcal{P}_{read}}$.
- Step 3: Preparation for access policy of repost. The access policy of repost can be expressed as $\mathcal{P}_{repost} = (u'.scl \in [o.scl, SL_{max}])\ AND\ (g'.sct \in [g.sct, ST_{max}])\ AND\ (u' \in U_g)$. Note that $(u' \in U_g)$ is implemented by using the user's id, and can be rewritten in the form of range. We can see clearly that the front part of $\mathcal{P}_{repost}$ is the same as the access policy of read. Therefore, the system only needs to compute the remaining two parts. To save storage space, our reposting header is the result of $\mathcal{P}_{repost} = (g'.sct \in [g.sct, ST_{max}])\ AND\ (u' \in U_g)$. The calculation method is similar to that in step 1. Finally, we can get the encrypted reposting header $\mathcal{H}_{\mathcal{P}_{repost}}$.
- Step 4: Preparation for access policy of delete. The access policy of delete can be expressed as $\mathcal{P}_{delete} = (u.sid = o.sid)\ AND\ (o \in O_g)$. The calculation method is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2018.2875697, IEEE Transactions on Dependable and Secure Computing

similar to that in step 1. Finally, we can get the encrypted deleting header $\mathcal{H}_{\mathcal{P}_{delete}}$.

There exist some redundant calculations in the four accessing policies. To improve the computing efficiency, we may compute $\mathcal{H}_{P_{read}}$ first and use the result to obtain $\mathcal{H}_{\mathcal{P}_{write}}$ and $\mathcal{H}_{\mathcal{P}_{repost}}$. When we finish all the calculations above, we can get the four headers for system operations. These headers can be concatenated together as the access header $\mathcal{H}'_{\mathcal{P}}$.

At the same time, in order to implement the access control condition related to the attribute of effective time period and provide more fine-grained access control, we encrypt the access header again with time attribute. Since the header $\mathcal{H}'_{\mathcal{P}}$ is a plaintext, we use the object's effective time period ($o.T_L$) as the access strategy for CCP-CABE to encrypt the access header. We add a new header – the time header ($\mathcal{H}_T$) for the time control. The time header $\mathcal{H}_T$ is protected by a key sys_{key} (acting as the random key $a_{k'}$), which is updated by the server every day. If the current time is not in $o.T_L$, the server cannot decrypt the $\mathcal{H}_T$ and therefore no user can access the object; on the contrary, if the current time is in the effective time, the server can get the access header ($\mathcal{H}'_{\mathcal{P}}$), and make the users use their private keys to decrypt the reading/writing/reposting/deleting header.

2) Implementation of reading operation: First, user u provides a set of attributes for the reading operation $\mathcal{L}_{u_{read}}$ and his/her reading key in this group $SK_{u_{read}}$. The server tries to decrypt $\mathcal{H}_T$ using current sys_{key}. If the current time can satisfy the access policy of time in $\mathcal{H}_T$, the system outputs $\mathcal{H}_{\mathcal{P}_{read}}$. If $\mathcal{L}_{u_{read}}$ satisfies the access policy of read in $\mathcal{H}_{\mathcal{P}_{read}}$, the secret key can be computed using Eq. (6). Finally, user u can get the permission of the reading operation, i.e., the system accepts the user's read request for the object.

3) Implementation of writing operation: At time t, user u' wants to write the object o in a group g. First, u' provides a set of attributes for the writing operation $\mathcal{L}_{u_{write}}$ and his/her writing key $SK_{u_{write}}$ in this group. The server checks $o.T_L$ similarly as that it does in the reading operation. If $\mathcal{L}_{u_{write}}$ satisfies the access policy of write in $\mathcal{H}_{\mathcal{P}_{write}}$, the secret key can be computed using Eq. (8). Then, u' can get the permission of the writing operation, and the system accepts the user's request for the object. As a result of the writing operating, user u' posts a new version v to the system. The attributes of the new version v are the same as that of object o, except that v's security level ($v.scl$) is set to no lower than o's security level ($o.scl$), the v's semantic tag ($v.sct$) is equal to g's semantic tag ($g.sct$), and the effective time period of v is set from the current time t to the end time of the effective time period of object o. In this system, we use the comment feature as the writing operation. Therefore, we do not need to re-encrypt the object. Instead, the comment is encrypted as a new object. While the comment does not have a new random a_k when encrypted, it uses object o's a_k which can be obtained in $\mathcal{H}_{\mathcal{P}_{write}}$. The new version can also be read, written, reposted and deleted.

4) Implementation of reposting operation: At time t, user u' wants to repost the object o from group g to group g'. First, u' provides a set of attributes for the reposting operation $\mathcal{L}_{u_{repost}}$, as well as his/her reading key $SK_{u_{read}}$ of this group and the reposting key $SK_{u_{repost}}$ of the destination group. The server checks $o.T_L$ similarly to that it does in the reading operation. After that, if $\mathcal{L}_{u_{repost}}$ satisfies the attributes requested in $\mathcal{H}_{P_{read}}$ and $\mathcal{H}_{\mathcal{P}_{repost}}$, the secret key can be computed using Eq. (8). Then u' can get the permission of the reposting operation. When u' finishes the reposting operation, u' posts a new version v to the system. The attributes of the new version v are the same as that of object o, except that the security level of v is equal to the security level of the original owner of object o, the semantic tag of v is equal to the semantic tag of g, and the effective time period of object v is from the current access time t to the minimum of the end time of effective time period of object o and the end time of effective time period of group g.

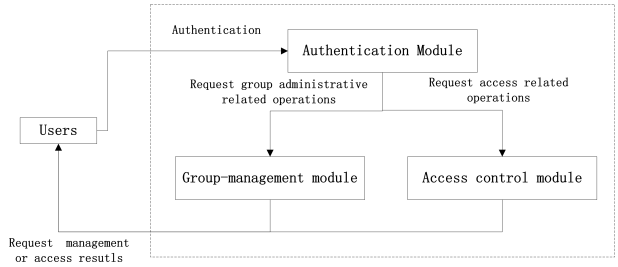


Fig. 4. The main architecture and procedures of the system

5) Implementation of deleting operation: At time t, user u wants to delete object o. First, u provides a set of attributes for the deleting operation $\mathcal{L}_{u_{delete}}$ and his/her reading key $SK_{u_{read}}$ of this group and the deleting key $SK_{u_{delete}}$ of the destination group. The server checks $o.T_L$ similarly as that in the reading operation. After that, if $\mathcal{L}_{u_{delete}}$ satisfies the attributes requested in $\mathcal{H}_{\mathcal{P}_{delete}}$, the secret key can be computed using Eq. (8). Then u can get the permission of the deleting operation. After that, $\mathcal{H}_T$ should be re-encrypted, and the effective time period of the object should end at time t. The deleted object does not need to be re-encrypted so as to reduce the computation time.

C. System architecture

The implementation of the system is divided into three main modules: authentication module, access control module and group management module. Fig. 4 shows the main architecture and procedures of the system. The involved procedures of the system are as follows:

1) The user sends a request to the access control system. The system first authenticates the user identity through the authentication module and then determines whether the user has been registered.
2) After the user passes the authentication module, the system enters the group management module or the access control module.
3) If the user requests to add/delete a group, the system enters the group management module. By verifying the group's attributes and the user's attributes, the system

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2018.2875697, IEEE Transactions on Dependable and Secure Computing

determines whether to accept the request. After that, the system modifies the order relationship. If the user request is to join/remove a user from a group, the system also enters the group management module. By verifying the group's attributes and the user's attributes, the system determines whether to accept the request. If the join (or remove) operation is accepted, the private key is sent to (or remove by) the user who imposes the operation.

4) If the user request to access the data, the system enters the access control module. According to the object information, the system finds the requested object. The system tries to decrypt the read/write/repost/delete header. If the user's attribute set satisfies the access policy that the header needs, the system can decrypt the header successfully and get the symmetric key a_k of the object. Finally, the system sends the requested data/operation to the user.

VI. Security and performance analysis

A. Security analysis of the proposed privacy policies

In our oGBAC model, we assume each group is charactered by several attributes including the security levels, semantic tags and effective time. Privacy policies are proposed based on the partial ordering relationship between these attributes. There is an operator "$\leqslant$" representing the partial ordering relation between security levels on $SCLS$, an operator "$\preceq$" representing the partial ordering relation between attribute tags on $SCTS$, and an operator "$\subseteq$" representing the inclusion relation between effective time intervals of T_L. Since the operations in different sets are closed, $(SCLS, \leqslant)$, $(SCTS, \preceq)$ and $(T_L, \subseteq)$ form algebraic systems respectively and they are all partially ordered sets.

First, we proof the product of the attributes in the oGBAC model forms a bounded lattice. Define the Cartesian product of $SCLS$ and $SCTS$ (i.e., $SCLS \times SCTS$) as a secure attribute L, where for any $l \in L$, $l = \{(scl, sct) | scl \in SCL \wedge sct \in SCT\}$. Let L_{min} and L_{max} denote the least and most secure attributes in the system respectively, then we have the following theorem.

Theorem 1. *If operator "$\triangleleft$" is in the set of secure attributes $SCLS \times SCTS$, $(scl_1, sct_1) \triangleleft (scl_2, sct_2)$ means $scl_1 \leq scl_2$ and $sct_1 \preceq sct_2$, then $(L, \triangleleft)$ forms a bounded lattice.*

We omit the detailed proof due to space limitations.

In the same way, we can present that secure attributes under the Cartesian products of $SCLS$, $SCTS$ and T_L (i.e., $SCLS \times SCTS \times T_L$) along with their operators also form a bounded lattice.

Theorem 2. *If operator "$\diamond$" is in the set of secure attributes $LST = SCLS \times SCTS \times T_L$, $(scl_1, sct_1, t_{L1}) \diamond (scl_2, sct_2, t_{L2})$ means $scl_1 \leq scl_2$, $sct_1 \preceq sct_2$ and $t_{L1} \subseteq t_{L2}$, then $(LST, \diamond)$ forms a bounded lattice.*

The security of the model comes from the security of the bounded lattice. In each policies of the oGBAC model, the conditions are constrained according to the partial orderings of the bounded lattice to make sure the information flow from attributes with lower orderings to attributes with higher (or equal) orderings. If an attribute a_1 is greater than or equal to an attribute a_2 in its lattice based partial ordering, we say that attribute a_1 dominates attribute a_2. We have the following proposition based on the information flow model proposed in [25].

Proposition 1. *If and only if the attribute tags (including the semantic tag, security level and/or the effective time period) of a user dominate the attribute tags of an object, the user can access the object (within the same group).*

Note that in our model, when the access control is related to a single group (like policy of $read$ and $write$), we mainly consider the security level attribute (because when a user and an object are in the same group, both of them have the tag of this group, so we do not need to consider the tag attribute); when the access control is related to two groups (like policy of $repost$), then we should consider the tag attributes of groups.

In the oGBAC model, there is also an implicit requirement. We write it as an proposition as follows.

Proposition 2. *If and only if a user and an object stay in the same group, the user can access the object.*

Proposition 2 is consistent with the basic requirement of the policies proposed in Section IV. That is to say, when a user u_1 posts an object o in his group g_1, another user u_2 in a different group has no chance to access object o, except that object o is reposted (may be reposted more than once) and a new version v is formed in user u_2's group.

It is easy to prove the security of each policy proposed in Section IV according to theorems and propositions shown above. Here, we focus on the analysis of the consistency of each policies and the correctness of multi-policies that are applied in turn in the "whole life-cycle" of the group based information sharing environment.

Theorem 3. *In our oGBAC model, if the setting of the initial system state is secure and the system state S_n is secure, after a legitimate operation, the following system state S_{n+1} is also secure.*

Proof. We simplified the whole access states of oGBAC as shown in Fig. 5. The states show an object's "whole life-cycle" sharing or access history. The initial state is the $post$ operation, and the ending state is $delete$ operation or time (effective time) expired. After an object is posted into a group, one must read the object first before he/she writes or reposts the object; after the reading operation, the object may be written or reposted again. We only need to analyze the security of six multi-operations: $post \rightarrow read$, $read \rightarrow write$, $read \rightarrow repost$, $write \rightarrow read$, $repost \rightarrow read$, and $other\ operations \rightarrow Delete\ or\ time\ expired$. Without loss of generality, in the proof, we assume that user u is the owner of the object o, and user u' is not the owner of the object o (though the user u' could also be the user u).

1) $post(u, o, g, Teg_0, Lev_0, t_1) \rightarrow read(u', o, g, t_2)$: Following Policy 7, after the owner of object o posts object o into group g, the semantic tag of group g is added to the set of the semantic tags of object o, the security

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2018.2875697, IEEE Transactions on Dependable and Secure Computing

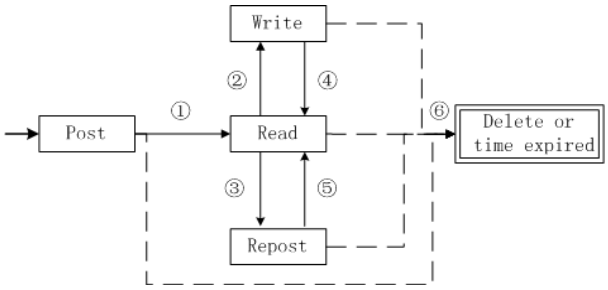


Fig. 5. The simplified access states of oGBAC model

level of object o is set to no lower than the security level of group g, and the effective time period of the posted object is from the current posting time t to the end time of the effective time period of group g. Before another user u' can read object o in group g, according to Proposition 2, u' and object o should stay in the same group of g, which means that user u' has already been invited by user u to $join$ group g. Following Policy 2, the security level of user u' is set to no lower than the security level of group g, and the semantic tag of the group g is added to the set of the semantic tags of user u'. According to the above analysis, the security levels of o and u' are not lower than those of g, and the semantic tag of g are in the sets of the semantic tags of both o and u'. Therefore, if the access time t_2 is in effective time periods of both user u' and object o, the condition $(o \in O_g) \wedge (u' \in U_g) \wedge (o.scl \leq u'.scl) \wedge (g.sct \in (o.SCTS \cap u'.SCTS)) \wedge (t \in u'.t_L \cap o.t_L)$ can be satisfied. According to Policy 7 and Proposition 1 (or Theorem 2), the access of $read(u', o, g, t_2)$ can be approved. So the consistency and the security of multi-operations $post(u, o, g, t_1) \rightarrow read(u', o, g, t_2)$ is proved.

2) $read(u', o, g, t_1) \rightarrow write(u', o, v, g, t_2)$: From Polices 7 and 8, we can see that the result of the read operation is $o \rightarrow u'$, and $o \rightarrow u'$ is just the condition of the write operation. According to Theorem 2, after the two operations, the information is assured to flow from attributes with lower orderings to the attributes with higher (or equal) orderings.
3) $read(u', o, g, t_1) \rightarrow repost(u', o, v, g, g', t_2)$: From Policies 7 and 9, we can see that the result of the read operation is $o \rightarrow u'$, which is one of the conditions of the repost operation. If the other two conditions (user u' belongs to both group g and group g', as well as the semantic tag of g' is greater than the semantic tag of g) are also satisfied, the repost operation can be approved. According to Theorem 2, after the two operations, the information is assured to flow from lower attributes to higher (or equal) ones.
4) $write(u, o, v, g, t_1) \rightarrow read(u', v, g, t_2)$: After the write operation, there is a new version of object v, with the security level not lower than that of the original object o, and the semantic tag equivalent to that of the group g. If another user u' wants to read the new version of the object from the same group g, the security level of the user u' should be not lower than the security level of object v, the semantic tag of g should be in the set of semantic tags of v (and also in the set of the semantic tags of u'). If the security level of user u' is not lower than the security level of the object v, then the other conditions of the second operation $read(u', v, g, t_2)$ are satisfied. So the consistency and the security of the two operations are proved.
5) $repost(u, o, v, g, g', t_1) \rightarrow read(u', v, g', t_2)$: The analysis is similar to that of $write(u, o, v, g, t_1) \rightarrow read(u', v, g, t_2)$.
6) $The\ \ operations\ \ except\ \ delete \rightarrow delete(u, o, g, t)\ \ or\ \ time\ \ expired$: If user u is the owner of object o, he/she can decide to delete the object during its effective time interval. After the delete operation, the effective time period of object o ends at deleted time t. In the case where the object is deleted or its effective time expires, the object cannot be accessed again (From Policies 5-9 we can see that all access related polices in the oGBAC model require the effective time condition), which assures the security when the whole access state (of an object) is finished.

□

B. Security analysis of the CCP-CABE based implementation

The implementation scheme is based on the CCP-CABE proposed in [34]. So the security of the implementation scheme is set up on the basis of the security in [34]. In [34], the security for MRDF, key collusion attacks and chosen ciphertext attacks are analyzed. Comparing with the original CCP-CABE in [34], our implementation has the changes of the two aspects: one is the use of multiple headers $(\mathcal{H}_{\mathcal{P}_{read}}, \mathcal{H}_{\mathcal{P}_{write}}, \mathcal{H}_{\mathcal{P}_{repost}}, \mathcal{H}_{\mathcal{P}_{delete}})$ in $\mathcal{H}'_{\mathcal{P}}$, the other is the use of two levels of encryption (the first level is provided by $\mathcal{H}_T$ and the second level is provided by $\mathcal{H}'_{\mathcal{P}}$). The following security analysis is focused on the two aspects.

Before analyzing the security, we give the definition of DLP assumption [34].

Definition 8. *(Discrete Logarithm Problem (DLP) Assumption) Given $(g_1, g_1^y) \in \mathbb{G}$ where $y \in \mathbb{Z}_n^*$, it is intractable to compute y.*

1) The security of multi-headers: Here, we assume that the attacker can obtain $\mathcal{H}'_{\mathcal{P}}$. We analyze that a_k is not available in the situation that the attacker has $\mathcal{H}'_{\mathcal{P}} = (\mathcal{H}_{\mathcal{P}_{read}}, \mathcal{H}_{\mathcal{P}_{write}}, \mathcal{H}_{\mathcal{P}_{repost}}, \mathcal{H}_{P_{delete}})$. Recall that for each $X \in \{read, write, repost, delete\}$, $\mathcal{H}_{\mathcal{P}_X} = (\mathbb{E}_{e_{k_X}}(a_k), h^{\varepsilon_X}, (v_{\mathcal{P}_X} w)^{\varepsilon_X}, (H(A_X))^{\varepsilon_X})$ and $e_{k_X} = e(\varrho^{\alpha}, \omega)^{\varepsilon_X}$. So we can see that the random key a_k is encrypted by different e_{k_X}s. Because the four e_{k_X}s are independent and have no relationship with each other, so we only need to consider the security risk of deriving e_{k_X} from one head of

$\mathcal{H}_{\mathcal{P}_X}$. From $e_{k_X} = e(\varrho^{\alpha}, \omega)^{\varepsilon_X}$ we can see e_{k_X} is protected by ε_X in $\mathcal{H}_{\mathcal{P}_X} = (\mathbb{E}_{e_{k_X}}(a_k), h^{\varepsilon_X}, (v_{\mathcal{P}_\mathcal{X}} w)^{\varepsilon_X}, (H(A_X))^{\varepsilon_X})$. According to the DLP assumption, it is hard to derive ε_X from $\mathcal{H}_{\mathcal{P}_X}$, so one cannot derive e_{k_X} from the head of $\mathcal{H}_{\mathcal{P}_X}$, and also cannot derive it from the head of $\mathcal{H}'_{\mathcal{P}}$. Because here $\mathcal{H}'_{\mathcal{P}}$ is in ciphertext and e_{k_X} is the protected key, so we can say header $\mathcal{H}'_{\mathcal{P}}$ is secure for chosen-ciphertext attacks.

2) The security of two-level encryption: As described in Section V, a_k is encrypted by AES encryption within two levels. In the first level, it is encrypted by AES encryption with e_{k_X} ($X \in \{read, write, repost, delete\}$) in different parts of the headers ($\mathcal{H}_{\mathcal{P}_{read}}, \mathcal{H}_{\mathcal{P}_{write}}, \mathcal{H}_{\mathcal{P}_{repost}}$ and $\mathcal{H}_{P_{delete}}$); in the second level, it is encrypted by AES encryption again when $\mathcal{H}'_{\mathcal{P}}$ is encrypted by AES encryption with $a_{k'}$. We analyze that the a_k is not available after two-level encryption. Because the two levels use symmetric encryption, we can just analyze the AES encryption scheme. In our scheme, we use double AES to encrypt the plaintext. Although there may be a meet-in-the-middle attack, we can lengthen the key a_k. Due to the security of the double-encryption of the AES encryption, a_k is not available by the attacker; that is, the secret key of a_k is well protected by double-encryption of the AES, and can only be obtained when the attributes of related entities (users, groups and objects) satisfy the security policies defined in the headers of $\mathcal{H}'_{\mathcal{P}}$ and $\mathcal{H}_{\mathcal{T}}$.

C. Efficiency analysis of the CCP-CABE based implementation

In this system, we have adopted a combination of symmetric encryption and asymmetric encryption. Asymmetric encryption takes more time than symmetric encryption. Meanwhile the time spent in symmetric encryption is related to the size of plaintext. So here we only evaluate the time consumed by asymmetric encryption, that is, the time it takes to generate the file header.

We have implemented our scheme on an experimental cloud computing environment. We simulate the encryption service and the storage service by using two local servers with 2.60 GHz Intel Core CPU and 4G RAM running Windows 10. The computational cost is computed by utilizing the Java Pairing-Based Cryptography (jPBC) library [35] without considering the communication cost. Similar to the work in [36], we use the bilinear map system $\mathbb{S}$ of composite order n where $n = sn'pq$ and $|p| = |q| = 128$ bits. In addition, we use AES-128 for the recursive encryption and decryption over the attribute domains. We show the practical computational cost of each operation in table III. It shows that the reading operation takes minimum time, while the other operations take more time than the reading operation. The reason is that the decryption algorithm takes much less time than the encryption algorithm. The writing and reposting operations generate a new encrypted version, and thus, they take more time than the posting operation. For the deleting operation, since after the deleting operation, the object simply changes the attribute of the valid time and does not change any attribute in the $H_{P'}$, the system only needs to recalculate H_T and does not need to recalculate $H_{P'}$. Thus, the time required for deleting operation is less than that required by the writing / reposting operations.

TABLE III
THE TIME TAKEN FOR EACH OPERATION

Operations	time(mm)
posting operation	9749.08
reading operation	326.21
writing operation	10067.53
reposting operation	10278.62
deleting operation	1221.67

D. Usability analysis

In this subsection, we take the typical Facebook privacy leakage scenarios described in literature [20] as an example to analyze the usability of the proposed policies, especially for the policy of repost, which involve the information flow among different OSNs' groups. In literature [20], there are three scenarios and each scenario has two or three cases. For simplicity, we rewrite the scenarios and focus on the ones with privacy leakage. All scenarios assume there are three users, Alice, Bob and Eve. Note that there is a slight difference between the operations (such as "post" and "tag") used in Facebook and in our model. However, we also can map the corresponding operations used in Facebook into (the combination of) the operations of our model and solve the problems that cannot be solved by Facebook.

Scenario 1. *Alice and Bob are friends, while Eve is not their friend. Alice posts some content on Bob's profile. Bob assigns visibility "friends of friends" to the post. Eve wants to view the information posted on Bob's profile without the latter knowing it. To this end, Eve becomes a friend of Alice. As the post has visibility "friends of friends", Eve can view the post and all comments in its reposts. So, there is a privacy leakage from Bob's profile to Eve.*

In Facebook, a user can post content on another uses's profile. The former is the data provider and the latter is the data host. In scenario 1, due to the visibility "friends of friends", Eve can read the post hosted by Bob, and thus there is a privacy leakage from Bob to Eve. Now, we can represent the scenario with oGBAC model, as shown in Fig. 6. Let $g =$ "$group\ of\ Bob's\ friends$ and $g' =$ "$group\ of\ Alice's\ friends$. According to Policy of repost, at time t_1, Alice can repost object o from group g to group g' and forms a new version v ($repost(Alice, o, v, g, g', t_1)$), if the following conditions can be satisfied: the semantic tag of g' is greater than the semantic tag of g in their partial ordering, Alice is in both group g and g' and the conditions that Alice can read object o. Apparently, the latter two conditions have been satisfied because Alice is the friend of Bob and the object is posted by her into Bob's profile, so she can read it. As the result of the repost operation, the security level of v is set to equal to that of original object o ($v.scl = (o.soid).scl$). So, later, if Eve becomes the friend of Alice, Eve is now in the group of g'. If Eve wants to view object v ($read(Eve, v, g', t_2)$), according to Policy of read, it should be satisfied that the security level of Eve is not less than that of v ($v.scl \leq Eve.scl$). Because $v.scl = (o.soid).scl$, Eve cannot view o if her security level is less than that of o. And

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2018.2875697, IEEE Transactions on Dependable and Secure Computing

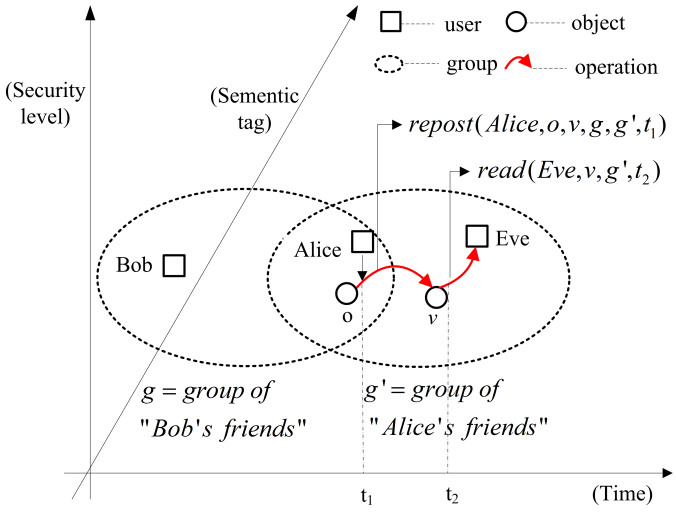


Fig. 6. Representation typical reposting scenario by oGBAC model

Bob can increase the security level of the object o to prevent it from being viewed by other users.

Scenario 2. *Bob is a friend of Alice and Eve (while Alice is not a friend of Eve). Alice posts some content about Bob on her profile and tags Bob. The visibility of the post is set by Alice to "friends of friends". Because of the tag, the visibility of the post also includes the "friends of friends" of Bob. In addition, a copy of the post appears on Bob's profile (Bob is the data host for the copy). The visibility of this copy is "friends" (of Bob) based on the default profile setting of Bob for tags. Because of the friendship relations between the users, Eve is part of the visibility of the post. Alice realizes Eve is in the visibility of the post and wants to stop sharing it with Eve. Alice changes the visibility to "friends" or "only me", However, Eve can still see the content in the post, as she can see the copy on Bob's profile. So, there is a privacy leakage from Alice's profile to Eve.*

In Facebook, a user can tag a post or an image. In this scenario, for some content in Alice's profile, when Alice tags Bob, Alice is the tag issuer and Bob is the tag target. The analysis of how to use oGBAC model to represent this scenario is similar to Scenario 1. Let $g =$ "$group\ of\ Alice's\ friends$ and $g' =$ "$group\ of\ Bob's\ friends$. After Alice tags Bob on an object o at time t_1, the object o is reposted from group g to group g' and forms a new version v ($repost(Alice, o, v, g, g', t_1)$), with $v.scl = o.soid.scl = o.scl$. However, in oGBAC model, though Eve is in the group of g', she cannot read object v except that her security level is not smaller than that of v ($v.scl \leq Eve.scl$). Because $v.scl = o.scl$, the condition is equal to $o.scl \leq Eve.scl$. Thus Eve may not view the reposed object v if the security level is low, and Alice can increase the security level of the shared object (for example, in the scenario the mentioned "only me" means to set the security level to the maximal one) to prevent it from being shared by Eve. By this way, oGBAC model can resolve the information leakage risk in Scenario 2.

Scenario 3. *Bob and Eve are fiends, while Alice is not a friend of Bob and Eve. Bob uploads an embarrassing photo of Alice in an album. Bob tags Eve in the image, making an instance of the image appear on Eve's profile. The visibility of the image is Bob and Eve's friends. At this point, Alice may not be aware that a photo of her has been uploaded. A friend of Bob tags Alice to make sure everyone knows which person is the user of the image. Alice is notified that she has been tagged and, thus, she becomes aware of the existence of the image. In response, Alice deletes the image from her profile. However, the image still appears on Bob's profile, including the tag pointing to Alice. Alice decides to remove the tag as well. However, the image in Bob's profile and its copy in Eve's profile remain visible. This is a special kind of privacy leakage, which results the posting photo of other person without directly tagging or notifying the person.*

This is a special case, because the owner and poster of the embarrassing photo are Bob, while the content (privacy information) is about Alice. It is difficult to handle such case because Facebook allows users to post an image about a person while tagging others. This is because it is difficult to let the OSN system authorize a user (Alice) the permission of deleting the images posted by other people (Bob), even though the posted image involves the privacy of the former user (Alice). So in the situation described here, by using oGBAC model, Alice can track the original user who first posts the image (she later is aware of the existence of the image after others (except Bob) tag her, which means new version of the photo is reposted from others to her. And by $v.soid = o.soid$ in the result of repost operation, she can learn the information of the original user who posts the image), and then let (or persuade) the original user who first posts the image to delete the original image and all its versions by Policy of delete ($for\ all\ v \in tr_{o.ooid}, delete(v)$).

E. Further discussion

First, we compare our model with the ordinary tag-based access control model. In [22], Hinrichs et al. proposed a tag-based authorization (TBA), which combines the ease of use of extensional systems while still maintaining a meaningful degree of the expressiveness of logical systems. In [23], the authors further extended TBA to support policy administration. TBA is designed for tag-based access control applications, it is flexible by separation of tagging and policy writing, and it is somewhat comprehensive by considering the real-world application issues such as tag ontology, policy delegation and administration. Our oGBAC is designed for (group based) social network information sharing, where privacy protection is the main objective of the designed policies, and the objective is achieved by making the information flow according to the partial ordering of the social network attributes. Besides tag, time and security level are also considered in oGBAC. Moreover, the attributes of the oGBAC are not constrained in tag, time and security level, and other attributes also can be adopted if needed. However, as studied in [10], a lot of attributes and their combinations on the one hand increase the time and task complexity in the real-world applications, on the other hand they are not needed. Existing study results have

shown that several key attributes and their combinations are good enough [10]. In our previous work [5], we conducted a questionnaire-based study among OSNs' users and found that there is a partial order based on either privacy levels or attribute tags for image sharing in OSNs. The study results of [10] and [5] in a certain way support the feasibility of oGBAC.

Second, we further discuss how to set the initial tag and/or security level in the policy of create and policy of post proposed in Section IV. Tagging is a relatively mature technique which has been used in some OSNs such as Facebook. With the rapid development of machine learning based object recognition techniques (such as R-CNN [37], Fast R-CNN [38] and Faster R-CNN [39]), automatic object recognition and tagging become easier and easier. As to the security level, it also can be defined by users, or first recommended by the system and then decided by the users. Our previous work [16] proposed a framework to predict and recommend the privacy level of a digital image by the matching of perceptual image hashing. Before posting an object into OSNs, the system can automatically predict the tag or privacy level of the object by machine learning based technologies and recommend it to users, and the users can make the final decision to accept or adjust the recommendation. However, how to set the partial ordering of the tags and security levels for the sharing objects and subjects needs further study. In our previous work [5], we have designed an HCI based method to "find out" the partial ordering of security levels and tags for sharing images in OSNs. More accurate and personalized recommendation methods can be further studied.

Thirdly, we discuss the privacy leakage issue caused by operation of "repost" (sharing information among different groups). In oGBAC, the proposed polices can record the version of objects shared to different groups by a tree, and finally, if the owner wants to delete the object, he can delete every version of the object through traveling the tree. However, in the real-world applications, the problems may be more intricate. For example, a shared user may download a shared object, rename it into another object and share it on his own OSNs' profile. In this situation, other techniques, such as data fingerprint and information forensics, should also be incorporated in resolving these problems.

VII. Conclusions and future work

With a growing need of information sharing and the new development of the Internet, there is a strong demand of new security models for personal information protection when people share information. In this paper, we presented the oGBAC model to meet the need of privacy protection in social network information sharing applications. We have conducted a survey of the related studies to find the limitations in existing privacy mechanisms of OSNs.

Based on the survey findings, we proposed the oGBAC model which combines the classic access control model with the information flow control policies. This model can ensure that the sharing information flows within or among OSN groups according to the partial ordering of OSNs' environmental attributes (such as security level, semantic tag and effective time period). We implement the model with CCP-CABE to demonstrate the effectiveness of the model. The security and other performance of the model as well as the implemented system are analyzed.

Our model could be improved in terms of efficiency and accuracy when deployed in real-world applications. The social networks are much more complex than the operating system and other environments where traditional access control policies are used. The methods of information flow cannot solve the problems like untrusted nodes. Some OSN users do not care about their access control policies. If users do not have enough knowledge about privacy and security, they may have problems when they first use our model. So, in our future work, we will further improve the model to simplify the operations and make it much more automatic and convenient. We will also consider how to more closely combine the machine learning method with our model. For example, we can use the deep learning method to predict the security level, semantic tag of each subject or object, and input the prediction results into the oGBAC model to make it more automatic.

Acknowledgment

The authors thank Xinling Shi and Haitong Hao for their comments and design of some policies presented in this paper. The authors also thank the anonymous reviewers for their constructive comments that have helped improve the quality of this paper.

References

[1] A. C. Squicciarini, M. Shehab, and J. Wede, "Privacy policies for shared content in social network sites," *The VLDB Journal*, vol. 19, no. 6, pp. 777–796, Dec. 2010. [Online]. Available: http://dx.doi.org/10.1007/s00778-010-0193-7

[2] N. B. Ellison, C. Steifield, and C. Lampe, "The benefits of facebook friends: social capital and college students use of online social network sites," *Journal of Computer-Mediated Communication*, vol. 12, no. 4, pp. 1143–1168, 2007.

[3] Y.-P. Guan, Z.-Q. You, and X.-P. Han, "Reconstruction of social group networks from friendship networks using a tag-based model," *Physica A: Statistical Mechanics and its Applications*, vol. 463, pp. 485 – 492, 2016.

[4] J. Qiu, Y. Li, J. Tang, Z. Lu, H. Ye, B. Chen, Q. Yang, and J. E. Hopcroft, "The lifecycle and cascade of wechat social messaging groups," in *Proceedings of the 25th International Conference on World Wide Web*, ser. WWW '16, 2016, pp. 311–320.

[5] X. Hu, D. Hu, S. Zheng, W. Li, F. Chen, Z. Shu, and L. Wang, "How people share digital images in social networks: a questionnaire-based study of privacy decisions and access control," *Multimedia Tools and Applications*, Feb 2017. [Online]. Available: https://doi.org/10.1007/s11042-017-4402-x

[6] R. Krishnan, R. Sandhu, J. Niu, and W. H. Winsborough, "A conceptual framework for group-centric secure information sharing," in *Proceedings of the 4th*

International Symposium on Information, Computer, and Communications Security, ser. ASIACCS '09. New York, NY, USA: ACM, 2009, pp. 384–387. [Online]. Available: http://doi.acm.org/10.1145/1533057.1533111

[7] S. Ahern, D. Eckles, N. S. Good, S. King, M. Naaman, and R. Nair, "Over-exposed?: Privacy patterns and considerations in online and mobile photo sharing," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '07. New York, NY, USA: ACM, 2007, pp. 357–366. [Online]. Available: http://doi.acm.org/10.1145/1240624.1240683

[8] A. D. Miller and W. K. Edwards, "Give and take: A study of consumer photo-sharing culture and practice," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '07. New York, NY, USA: ACM, 2007, pp. 347–356.

[9] P. Klemperer, Y. Liang, M. Mazurek, M. Sleeper, B. Ur, L. Bauer, L. F. Cranor, N. Gupta, and M. Reiter, "Tag, you can see it!: Using tags for access control in photo sharing," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '12. New York, NY, USA: ACM, 2012, pp. 377–386.

[10] R. L. Fogues, J. M. Such, A. Espinosa, and A. Garcia-Fornes, "Exploring the viability of tie strength and tags in access controls for photo sharing," in *Proceedings of the Symposium on Applied Computing*, ser. SAC '17. New York, NY, USA: ACM, 2017, pp. 1082–1085. [Online]. Available: http://doi.acm.org/10.1145/3019612.3019909

[11] N. Vyas, A. Squicciarini, C.-C. Chang, and D. Yao, "Towards automatic privacy management in web 2.0 with semantic analysis on annotations," in *Collaborative Computing: Networking, Applications and Worksharing, 2009. CollaborateCom 2009. 5th International Conference on*, Nov 2009, pp. 1–10.

[12] A. C. Squicciarini, S. Sundareswaran, D. Lin, and J. Wede, "A3p: Adaptive policy prediction for shared images over popular content sharing sites," in *Proceedings of the 22Nd ACM Conference on Hypertext and Hypermedia*, ser. HT '11. New York, NY, USA: ACM, 2011, pp. 261–270. [Online]. Available: http://doi.acm.org/10.1145/1995966.1996000

[13] S. Zerr, S. Siersdorfer, J. Hare, and E. Demidova, "Privacy-aware image classification and search," in *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '12. New York, NY, USA: ACM, 2012, pp. 35–44.

[14] S. Zerr, S. Siersdorfer, and J. Hare, "Picalert!: A system for privacy-aware image classification and retrieval," in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, ser. CIKM '12. New York, NY, USA: ACM, 2012, pp. 2710–2712. [Online]. Available: http://doi.acm.org/10.1145/2396761.2398735

[15] A. C. Squicciarini, C. Caragea, and R. Balakavi, "Analyzing images' privacy for the modern web," in *Proceedings of the 25th ACM Conference on Hypertext and Social Media*, ser. HT '14. New York, NY, USA: ACM, 2014, pp. 136–147. [Online]. Available: http://doi.acm.org/10.1145/2631775.2631803

[16] D. Hu, F. Chen, X. Wu, and Z. Zhao, "A framework of privacy decision recommendation for image sharing in online social networks," in *IEEE First International Conference on Data Science in Cyberspace*, ser. DSC '16. IEEE, 2016, pp. 243–251.

[17] B. Carminati, E. Ferrari, R. Heatherly, M. Kantarcioglu, and B. Thuraisingham, "Semantic web-based social network access control," *Computers & Security*, vol. 30, no. 2C3, pp. 108 – 115, 2011, special Issue on Access Control Methods and Technologies.

[18] M. Shehab, A. Squicciarini, G.-J. Ahn, and I. Kokkinou, "Access control for online social networks third party applications," *Computers & Security*, vol. 31, no. 8, pp. 897 – 911, 2012.

[19] Y. Li, Y. Li, Q. Yan, and R. H. Deng, "Privacy leakage analysis in online social networks," *Computers & Security*, vol. 49, pp. 239 – 254, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404814001588

[20] S. Damen and N. Zannone, "Privacy implications of privacy settings and tagging in facebook," in *The Workshop on Secure Data Management*, 2013, pp. 121–138.

[21] R. Krishnan, J. Niu, R. Sandhu, and W. H. Winsborough, "Group-centric secure information-sharing models for isolated groups," *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 3, pp. 23:1–23:29, Nov. 2011. [Online]. Available: http://doi.acm.org/10.1145/2043621.2043623

[22] T. L. Hinrichs, W. C. Garrison, A. J. Lee, S. Saunders, and J. C. Mitchell, "Tba : A hybrid of logic and extensional access control systems," in *Formal Aspects of Security and Trust*, G. Barthe, A. Datta, and S. Etalle, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 198–213.

[23] S. Etalle, T. L. Hinrichs, A. J. Lee, D. Trivellato, and N. Zannone, "Policy administration in tag-based authorization," in *Foundations and Practice of Security*, J. Garcia-Alfaro, F. Cuppens, N. Cuppens-Boulahia, A. Miri, and N. Tawbi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 162–179.

[24] D. Bell and L. LaPadula, "Secure computer systems: Mathematical foundations and model," Bedford, Mass., Tech. Rep. M74-244, 1975.

[25] D. E. Denning, "A lattice model of secure information flow," *Commun. ACM*, vol. 19, no. 5, pp. 236–243, May 1976. [Online]. Available: http://doi.acm.org/10.1145/360051.360056

[26] R. S. Sandhu, "Lattice-based access control models," *Computer*, vol. 26, no. 11, pp. 9–19, Nov 1993.

[27] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman, "Role-based access control models," *Computer*, vol. 29, no. 2, pp. 38–47, Feb 1996.

[28] L. Wang, D. Wijesekera, and S. Jajodia, "A logic-based framework for attribute based access control," in *Proceedings of the 2004 ACM Workshop on Formal*

Methods in Security Engineering, ser. FMSE '04. New York, NY, USA: ACM, 2004, pp. 45–55. [Online]. Available: http://doi.acm.org/10.1145/1029133.1029140

[29] R. Sandhu, "Attribute-based access control models and beyond," in *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, ser. ASIA CCS '15. New York, NY, USA: ACM, 2015, pp. 677–677. [Online]. Available: http://doi.acm.org/10.1145/2714576.2749229

[30] Y. Yu, Y. Li, B. Yang, W. Susilo, G. Yang, and J. Bai, "Attribute-based cloud data integrity auditing for secure outsourced storage," *IEEE Transactions on Emerging Topics in Computing*, vol. PP, no. 99, pp. 1–13, 2017.

[31] A. Sahai, B. Waters *et al.*, "Fuzzy identity-based encryption." in *Eurocrypt*, vol. 3494. Springer, 2005, pp. 457–473.

[32] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM conference on Computer and communications security*. Acm, 2006, pp. 89–98.

[33] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Security and Privacy, 2007. SP'07. IEEE Symposium on*. IEEE, 2007, pp. 321–334.

[34] Z. Wang, D. Huang, Y. Zhu, B. Li, and C.-J. Chung, "Efficient attribute-based comparable data access control," *IEEE Transactions on computers*, vol. 64, no. 12, pp. 3430–3443, 2015.

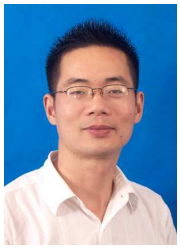
[35] A. De Caro and V. Iovino, "jpbc: Java pairing based cryptography," in *Computers and communications (ISCC), 2011 IEEE Symposium on*. IEEE, 2011, pp. 850–855.

[36] Y. Zhu, H. Hu, G.-J. Ahn, M. Yu, and H. Zhao, "Comparison-based encryption for fine-grained access control in clouds," in *Proceedings of the second ACM conference on Data and Application Security and Privacy*. ACM, 2012, pp. 105–116.

[37] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," pp. 580–587, 2013.

[38] R. Girshick, "Fast r-cnn," *Computer Science*, 2015.

[39] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: towards real-time object detection with region proposal networks," in *International Conference on Neural Information Processing Systems*, 2015, pp. 91–99.



Donghui Hu is an associate professor in the College of Computer Science and Information Engineering at Hefei University of Technology, China. He got his Ph.D. in Information Security at Wuhan University in 2009. He received his BS degree in Mathematics from the Anhui Normal University of China in 1995, an ME degree in Computer Science and Engineering from the the University of Science and Technology of China in 2004. His major research interests include network and information security.

Chunya Hu is Master's student in University of Chinese Academy of Sciences and received BS degree from HeiFei University of Technology in 2017. Her current research focuses on lattice cipher.

Yuqi Fan is an associate professor and an assistant to the Dean in the College of Computer Science and Information Engineering at Hefei University of Technology, China. He got his Ph.D. in Computer Science and Engineering at Wright State University in 2009. He received both B.S. and M.S. degrees in computer science and engineering from Hefei University of Technology in 1999 and 2003, respectively. His research interests include computer networks, SDN, cloud computing, cyber-phsical systems, and wavelength-division-multiplexing (WDM) network design and optimization.

Xintao Wu is a Professor in the Department of Computer Science and Computer Engineering at the University of Arkansas. He held a faculty position at the College of Computing and Informatics at the University of North Carolina at Charlotte from 2001 to 2014. He got his Ph.D. degree in Information Technology from George Mason University in 2001. He received his BS degree in Information Science from the University of Science and Technology of China in 1994, an ME degree in Computer Engineering from the Chinese Academy of Space Technology in 1997. His major research interests include data mining and knowledge discovery, data privacy and security.