Hindawi
Security and Communication Networks
Volume 2018, Article ID 7384194, 11 pages
https://doi.org/10.1155/2018/7384194

Research Article

HAC: Hybrid Access Control for Online Social Networks

Fangfang Shan,[1,2] **Hui Li**,[1] **Fenghua Li**,[3,4] **Yunchuan Guo**,[3] **and Ben Niu**[3]

[1]*State Key Laboratory of Integrated Service Network, School of Cyber Engineering, Xidian University, Xi'an 710071, China*
[2]*School of Computer Science, Zhongyuan University of Technology, Zhengzhou 450000, China*
[3]*State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China*
[4]*School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100093, China*

Correspondence should be addressed to Yunchuan Guo; guoyunchuan@iie.ac.cn

Received 23 October 2017; Accepted 2 April 2018; Published 17 May 2018

Academic Editor: Raymond Choo

The rapid development of communication and network technologies including mobile networks and GPS presents new characteristics of OSNs. These new characteristics pose extra requirements on the access control schemes of OSNs, which cannot be satisfied by relationship-based access control currently. In this paper, we propose a hybrid access control model (HAC) which leverages attributes and relationships to control access to resources. A new policy specification language is developed to define policies considering the relationships and attributes of users. A path checking algorithm is proposed to figure out whether paths between two users can fit in with the hybrid policy. We develop a prototype system and demonstrate the feasibility of the proposed model.

1. Introduction

Online social networks (OSNs) have attracted widespread popularity nowadays. Users can conveniently share personal information with friends via OSNs. More than 300 hours of videos are uploaded to YouTube and nearly 25 million photos are posted to Instagram every minute [1]. Considering the fact that sensitive information may be leaked, the protection of users' privacy becomes a challenging task. To address this problem, researchers have proposed the relationship-based access control (ReBAC) mechanism [2–5]. Resource owners specify access control policies on the basis of relationships between individual users. By maintaining the balance between ease of use and flexibility, ReBAC has been commonly applied in real OSN systems. It has also been recognized as one of the most straightforward and useful ways in protecting user's privacy.

With the development of mobile technologies, plenty of smart devices are connected to the network. These devices may generate a large amount of private information, such as location and health status, and then share the information through OSNs [6]. In general, mobile technology contributes the following features to the online social networks.

(i) More and more privacy information collected through smart mobile devices may be uploaded to online social networks.

(ii) Privacy information collected by smart mobile devices can be used for access control scheme of OSNs.

These features have brought about new challenges for access control schemes of online social networks. For example, Alice is shopping at Bergdorf Goodman in New York. She has recorded a video of the megamall with her Google glasses and published it in OSN to see if any female friends can give her some pieces of advice in choosing cosmetics. Perhaps some of her friends nearby may come to have lunch together. She does not want every friend to know what she is doing now, so friends who do not live in New York will not be granted access to this video. The widely used relationship-based access control methods cannot describe the attribute "location: in New York" and cannot meet Alice's needs. To provide finer-grained access control over private data generated by wearable devices or m-health, researchers should take attributes such as location, profession, and gender into consideration.

The access control mechanism named UURAC_{A} [4] was the first mechanism that took attributes of users into

consideration. Better expression ability and finer-grained access control policies are characteristic of it. However, the attribute-based policy is separated from the relationship-based one. In other words, the policy of $UURAC_A$ falls into two parts: the relationship-based policy and the attribute-based policy. For this reason, $UURAC_A$ can merely figure out common attributes of one or several users on the relationship path instead of specifically identifying different attribute of different users.

In this paper, we propose a hybrid access control model based on both attribute and relationship. It designs a new language of policy specification to specify policies based on attributes and relationships. Compared with the study of Cheng et al. [4], the policy specification language is characteristic of better expression effectiveness and easier usage. It presents detailed instructions on the policy specification language and several application examples. A path checking algorithm is proposed to find out whether paths between two users involved in OSNs would fit in with the attribute-based policy. The path checking algorithm is implemented to conduct experiments to validate the feasibility and performance of the scheme.

2. Related Work

As large amounts of private personal data are created by Web 2.0 applications, Carrie [7] summarized four technical requirements of access control mechanisms for social networks based on Web 2.0 technologies. He emphasized that the access control mechanisms for Web 2.0-based social networks should have characteristics of relationship-based, fine-grained, interoperability, and sticky policies and named it relationship-based access control (ReBAC).

To meet these requirements and protect privacy of social network users, researchers have proposed a variety of access control mechanisms for OSNs. These mechanisms are broadly divided into three categories. Methods of the first type leverage relationships between users and resources to constrain the access of privacy information. Some researchers introduce modal and hybrid logic into their access control model of OSNs. Others make use of cryptography to prevent unwanted access.

Most access control schemes made use of various relationships between users and resources to protect sensitive information in OSNs. In [8], the authors tried to define access control policies based on the type, depth, and trust level of relationships between web-based social networks (WBSNs) users. They proposed an access control model for WBSNs which is characterized by using certificates for verifying the authenticity of relationships and enforcing a rule-based access control approach at the client side. Carminati et al. [9] extended the mechanism presented in [8] by providing details on the enforcement of the access control model. They defined two protocols to verify the authenticity of relationships and analyzed the security of them. In [10, 11], Carminati et al. leveraged OWL, SWRL, and semantic web technologies to express filtering, administration, and authorization policies. They proposed an access control model to describe the relationships between users and resources. A relationship-based access control model was proposed by Cheng et al. [12] which utilized user-to-user, user-to-resource, and resource-to-resource relationships to define access control policies. It can be used to capture controls of administrative activities of users together with other normal usage activities. In [13], the authors presented an object-to-object relationship-based access control model (OOReBAC) which leveraged relationships between objects to control access of them. In [14], the authors presented a graph-based access control model which can be used in various systems, not just social networks. It introduced concepts of path conditions and principal matching and has better policy expression ability and request evaluation efficiency.

With the development of semantic web technology, some researchers considered using modal and hybrid logic in their access control schemes of OSNs. Masoumzadeh and Joshi [15] presented a scheme of access control based on ontology that can be used on semantic web-based social networks to support both system and user policies. Fong et al. [16] formalized the privacy preservation mechanism of Facebook-style social network systems and proposed an access control model for them. The policies of this model are able to express access control requirements such as common friends and clique. Fong [17] defined a modal logic-based language to specify and do composition of ReBAC policies. He presented a case study of EHR systems to prove that ReBAC can be used in application domains other than social computing. In [3], the authors demonstrated that policy language proposed in [17] was incomplete and it was unable to express all ReBAC policies. They extended the policy language of [17] to identify vertex and support for disjoint intermediaries and proved it to be representationally complete. As an extension of modal logic, Bruns et al. [18] utilized hybrid logic to specify policies and enforce access control decisions in relationship-based access control approach. A fragment of hybrid logic was used to express complex relationship-based access control policies such as "at least three friends". Several other works [19–21] also utilized the hybrid logic to support better expression capacity of access control needs.

Researchers then considered adopting cryptography and other technologies to the access control mechanisms of OSNs. Anwar and Fong [22] designed a visualization tool to show the result of policy configurations. In [23], the authors presented an access control mechanism to protect textual contents in online social networks which is enforced to be transparent to content publishers and readers. The proposed system leveraged automatic semantic annotation to analyze the semantics of the contents in order to generate different versions according to types of readers. Apart from relationship-based access control mechanisms, other works concentrated on security protocols that leverage cryptographic techniques to achieve access control goals [24–30].

3. HAC Model Foundation

This section presents the foundation of HAC, including the attributes in OSN, the social graph with attributes, and the model components.

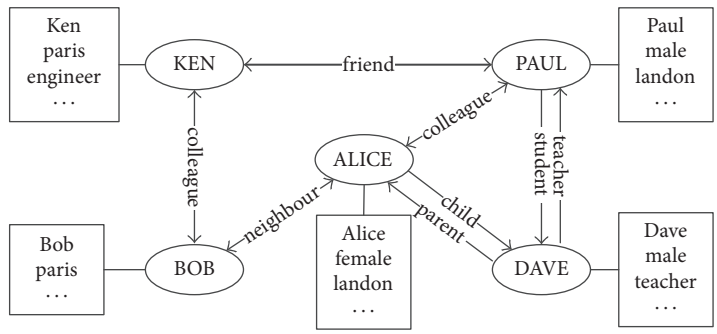


Figure 1: A sample social graph.

3.1. Attributes in OSN. Most of the recent access control schemes for OSN make access control decision based on relationships. By considering the relationships only, data owners cannot make proper access control policies based on location and time. Recent studies have shown that attribute-based access control (ABAC) can provide flexible and fine-grained access control in dynamic distributed systems [31–34]. As only the attributes of the subject, object, and environment are considered, most current solutions of typical ABAC schemes cannot be directly used in OSNs. Relationship of users should also be checked here. When registering an OSN account, users are always required to submit personal information, such as name and gender. This personal information is recognized as profile attribute, which can be used to define policies.

Attributes are categorized into profile attributes and relationship attributes in HAC.

Profile Attribute. Profile attribute contains information of environment and identity, or characteristics of users. In HAC, profile attributes fall into two types: user-defined attribute and objective attribute. The user-defined attributes are specified by the profile owner, such as gender, name, job, hometown, and hobbies. In contrast, the objective attributes are gained or defined by the OSN systems, such as time, location, and IP address.

Relationship Attribute. Relationship attribute is used to describe type, weight, and other information of relationships in OSNs.

3.2. Social Graph with Attributes. As shown in Figure 1, the researchers use a directed labeled simple graph G to abstract an OSN. The nodes of G represent users while edges represent the relationships between them. Each user is associated with a profile containing his profile attributes. Relationships between users are noted as relationship attributes. The social graph of HAC contains two types of information: (1) users and their profile attributes and (2) relationship attributes between the users.

The researchers use a triple $G = \langle N, \Sigma, E\rangle$ to describe the social graph in an OSN.

$N = \{\langle u, pro\_attr\_group\rangle \mid u \in U,\ pro\_attr\_group \in P\}$ denotes the set of nodes (or vertices) of the graph, containing profile attribute information of users in an OSN. U represents user set while P is the set of profile attributes.

$\sum = \{r_1, r_2, \ldots, r_n\}$ is the set of relationship attributes, each element of which represents a relationship attribute.

$E = N \times N \times \sum$ is the set of edges in the social graph, representing relationship attributes between users in the OSN.

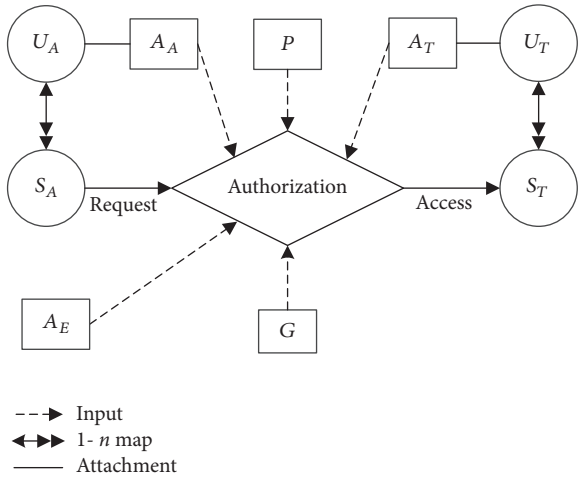


Figure 2: Model overview.

3.3. HAC Model Components. Figure 2 shows the conceptual diagram of HAC. Model components are access requester, target user, access requester attribute, target user attribute, environment attribute, sessions of access requester and target user, policy, authorization, and social graph.

Access *requester* (u_a) is a registered human being who may initiate an access request to a profile or a resource of a target user in the OSN. Each access requester has a set of *attributes* (a_a) to describe his personal information, such as gender, name, job, and hometown. *Target user* (u_t) is a registered human being whose profile or resource is the recipient of access. Each target user is also related to a set of *attributes* (a_t). *Environment attribute* (a_e) represents environment information used in access control procedure, such as time, location, and IP address. *Session of access requester* (s_a) is an active instance of an access requester logged into the OSN system while *session of target user* (s_t) represents

the active instance of a target user. *Policy* (p) is defined by the target user based on various attributes governing the access of his profile or resources. According to different kinds of attributes, policies are categorized into profile attribute-based policy and relationship attribute-based policy. *Social graph* (G) describes the relationships between users on social networks. It is denoted as a directed labeled simple graph. *Authorization* is an abstract function with attributes, request, social graph, and policy as inputs. It makes a decision to grant or deny the access of the target user. *Request* represents an access requester's initiation of access. It is described as a tuple $\langle u_a, operation, target\rangle$, where u_a is the access requester, *target* may be a profile or resource of the target user, and *operation* represents the access that can be performed on targets.

4. HAC Policy Specification and Evaluation

In this section, the researchers present policy language, policy specification, and the policy evaluation of HAC.

4.1. Policy Language. Policy of HAC is defined by the target user. It constrains the profile and relationship attributes of users along the relationship path. The policy language is defined as follows.

(i) N and E are user (or node) set and relationship (or edge) set, respectively.

(ii) $PN = \{n_1, n_2, \ldots, n_s\}$ $(1 \le s \le S)$ is the profile attribute name set for users (or nodes), where S is the number of profile attribute names.

(iii) $PV = \{v_1, v_2, \ldots, v_m\}$ $(1 \le m \le M)$ is the predefined profile attribute value set for users (or nodes), where M is the number of profile attribute values.

(iv) $RA = \{r_1, r_2, \ldots, r_k\}$ $(1 \le k \le K)$ is the predefined attribute set for relationships (or edges), where K denotes the number of relationship attributes.

(v) $AT(n)$ and $AT(e)$ are attributes of node n and edge e, respectively. $AT(n) \in PN \times PV$ is a binary relation on sets PN and PV, and $AT(e) \in RA$. The node attribute is a profile attribute while the edge attribute is a relationship attribute.

Profile attribute of a node is a binary relation on profile attribute name set and profile attribute value set. A profile attribute-based policy rule is composed of a profile attribute name, a relationship specifier, and a profile attribute value as shown below.

(i) *profile attribute name*, $AT(n)$, *profile attribute value*.

Note that the profile attribute-based policy rule is specified by the data owner. For example, policy rule *R1* says, "the current user's profession must be teacher". Policy rule *R2* requests the current user to be an adult.

R1: *profession* = "*teacher*"

R2: *age* > 18

A complete attribute-based policy rule is composed of one relationship attribute and several profile attributes as shown below.

(i) $[relationship\_attribute_1, (profile\_attribute_{11}; \ldots; profile\_attribute_{1n})] \ \ldots \ [relationship\_attribute_h, (profile\ attribute_{h1}; \ldots; profile\_attribute_{hn})]$

For example, *R3* indicates that "the requester must be a teacher living in New York, and he should be a friend of the data owner". *R4* specifies a rule saying that "only the adult male colleagues of the owner can access the resource". In *R5*, the relationship attribute is not used and it is shown as "-", which indicates that the relationship is not constrained. *R5* requires that the location of the requester should be London and the access time must between 2017-09-05 and 2017-10-05.

R3: [f, (*occupation* = "*teacher*"; *howetown* = "*New York*")]

R4: [c, (*gender* = "*male*"; *age* > 18)]

R5: [-, (*time* ∈ "2017-09-05: 2017-10-05"; *location* = "*London*")]

Table 1: Grammar for path sentences.

path_sentence ::= *path_word* | *path_word connector path_word*
connector ::= ∨ | ∧
path_word ::= "(" *path* "," *hop_count* ")"
hop_count ::= *num*
Path ::= *attr_specs* | *attr_specs path*
attr_specs ::= "[" *rel_attr* "," *pro_attr_group* "]"
rel_attr ::= r_1 | r_2 | .. | r_k | *RA* where $RA = \{r_1, r_2, \ldots, r_k\}$
pro_attr_group ::= *pro_attr_pair* | *pro_attr_pair pro_attr_group*
pro_attr_pair ::= "(" *pro_attr_name* "," *pro_attr_value* ")"
pro_attr_name ::= n_1 | n_2 | .. | n_l | *PN* where $PN = \{n_1, n_2, \ldots, n_l\}$
pro_attr_value ::= v_1 | v_2 | .. | v_m | *PV* where $PV = \{v_1, v_2, \ldots, v_m\}$
num ::= [0 - 9]+

4.2. Policy Specification. Policies are evaluated according to the paths between the access requester and the target user in social graph. The access control policy is composed of an operation and a path sentence. As shown in Table 1, the syntax of the path sentence is defined with Backus-Naur Form (BNF).

A path sentence consists of several path words that are connected by connectors. Every *path word* is composed of *path* and *hopcount*. The path specifies the relationships from the access requester to the target user. The *hopcount* represents the maximal number of edges along the path.

Unlike UURAC and UURAC_A, our path is aiming at expressing policies based on attributes. The researchers use relationship attribute and profile attribute to express the restriction on users and the relationship between them.

Several examples are given to show how to use hybrid rules to express the access control need in OSNs.

Example 1 (relationship attribute and profile attribute policy). If Jim wants to allow some users to access his photos, those users should share a common friend named "Jack" with him and their occupation must be doctor. He can specify a policy like this:

P1: $\langle$*photo_access*, ([f, (*name* = "*Jack*")] [f, (*occupation* = "*Doctor*";)], 2)$\rangle$

If Jim wants to show his photos to his friend Jack or his colleagues who are interested in medicine, the policy can be specified as below:

```
(1)  for all (attr_specs) of path
(2)      path_reg_exp = path_reg_exp + attr_spec.rel_attr
(3)  if (attr_specs) is the last one on the path then
(4)      return path_reg_exp
(5)  else
(6)      break
```

Algorithm 1: *RegularExpressionTrans*(*path*).

```
(1)  currentPath ← NIL; h ← 0
(2)  nodeHistory ← start
(3)  path_reg_exp ← RegularExpressionTrans(path)
(4)  if hopcount ≠ 0 then
(5)      return ADFS(start)
```

Algorithm 2: *PathCheck* (*G, path, hopcount, start, end*).

P2: ⟨*photo_access*, ([f, (*name* = "*Jack*")], 1) ∨ ([*c*, (*interest* = "*medicine*")], 1)⟩

For *P1*, the system has to figure out the basic path (ff, 2) according to the social graph and examine profile attributes of users along the path. If the access requester is a doctor and he has a common friend named "Jack" with Jim, he may get the permission. For *P2*, two kinds of access requesters can get the permission. First, the user's name is equal to "Jack", and an (f, 1) path is found between him and Jim. Second, the user is interested in medicine and there is a (c, 1) path in the social graph between him and Jim.

For each policy, the last attribute spec restrains the attributes of the access requester.

Example 2 (relationship attribute policy). Profile attributes of the following policy are empty. The policy specifies that coworkers of Jim's friends can access his profile. Policies like this can capture UURAC policies.

P3: ⟨*profile_access*, ([f, (-)][*c*, (-)], 2)⟩

4.3. Policy Evaluation. Algorithms of policy evaluation are presented in this section. The algorithms are used to evaluate whether the access requests should be granted. The algorithms have to find a required path between the access requester and the target user according to the social graph. The required path found in the social graph may ensure that the relationships between the access requester and the target user can satisfy the hybrid policy.

Regular Expression Transformation Procedure. Relationship attributes can be extracted from the path to form a regular expression as shown in Algorithm 1. For a path (*attr_specs* | *attr_specs path*), the algorithm traverses each attribute specification in the path and gets the relation attribute. All relation attributes are then catenated to be a regular expression which is used to match the paths in the social graph.

Path Checking Algorithm. Algorithm 2 shows the path checking method of HAC. It takes the social graph *G*, the *path*, the number of relationship attributes' limit *hopcount*, the source node *start*, and the target node *end* as input. It returns a Boolean value, of which the output T means the access request will be granted and F means denied. Here, the source node *start* and the target node *end* represent the target user and the access requester, respectively.

Similar to [8], the path checking method leverages a depth first search (DFS) to find the proper path in the social graph. Without the limit of *hopcount*, DFS may search along a path in the social graph too deep to find the proper path. Operations in OSNs always occur between the people with close relationships. Limited with *hopcount*, DFS is suitable for our model. The researchers improve the DFS to cope with profile attribute check and name it ADFS.

The variable *currentPath* is used to hold the node sequence traversed from the source node *start* to the current node. Variable h is used to tell if the *currentPath* exceeds the limit of *hopcount*. All nodes traversed are recorded by variable *nodeHistory*. These variables are initialized with *NIL*, 0, and the source node *start*, respectively. The main procedure gets regular expression of relationship attributes through *RegularExpressionTrans*(*path*). Then, it launches traversal function *ADFS*() with parameter *start* to find out if the proper path exists in the social graph.

The function *ADFS*() is shown in Algorithm 3. It takes u as the only parameter. If the algorithm takes a further step from the node u and makes $h + 1$ bigger than the *hopcount*, it returns F. Otherwise, the further step is legal. *ADFS*() picks up one edge (u, v, σ) starting with node u in the social graph. Then, the algorithm faces five cases. For *if 1*, the current target node v belongs to the variable *currentPath*. This means that the edge (u, v, σ) has been visited. The algorithm breaks from the loop. For *if 2*, the node v is unvisited and it is exactly the target node *end*. *ADFS*() first checks whether the relationship attribute of the current edge (u, v, σ) is equal to the dth regular expression of relationship attributes extracted from the attribute-based access control policy. If not, the algorithm breaks. Otherwise, it means that the path between *start* and *end* matching the regular expression is found. If the profile attribute of node v fits the requirement of the attribute-based access control policy, the algorithm sets h to be $h + 1$ and concatenates the edge to *currentPath*. Then, it saves the node v in *nodeHistory*. In *if 3*, the node v is unvisited, and it is exactly the target node *end*. But the relationship attribute of the current edge (u, v) is not equal to the hth regular expression of relationship attributes extracted from the attribute-based access control policy. The algorithm will break from the current loop. In *if 4*, node v is unvisited and it is not the target node *end*. The relationship σ is not *path_reg_exp*[h], and the algorithm breaks from *if 4*. In *if 5*, node v is unvisited, and it is not the target node *end*. The relationship attribute of the current edge (u, v, σ) is equal to the dth regular expression of relationship attributes. The algorithm sets h to be $h + 1$, concatenates (u, v, σ) to *currentPath*, sets v to be *currentNode*, and saves the current node to *nodeHistory*. Then, the function *ADFS*() is called recursively from node v. If a matching path is found, the algorithm will return T from

```
(1)  if h + 1 > hopcount then
(2)    return F
(3)  else
(4)    for all (v, σ) where (u, v, σ) in G do
(5)     if 1 (v ∈ currentPath)
(6)      break
(7)     if 2 ((v ∉ currentPath) && (v = end))
(8)      if (path_reg_exp[h] != σ) then
(9)          break
(10)     if (! match (path.attre_spec[h].pro_attr_group, v.pro_atr_group)) then
(11)         break
(12)     h ← h + 1; currentPah ← currentPath.(u, v, σ)
(13)     currentNode ← v
(14)     nodeHistory ← nodeHistory.(currentNode)
(15)     return T
(16)    if 3 ((v ∉ currentPath) && (v = end) && (path_reg_exp[h] != σ))
(17)     break
(18)    if 4 ((v ∉ currentPath) && (v ≠ end) && (path_reg_exp[h] != σ))
(19)     break
(20)    if 5 ((v ∉ currentPath) && (v ≠ end) && (path_reg_exp[h] = σ))
(21)         h = h + 1; currentPath ← currentPath.(u, v, σ)
(22)         currentNode ← v
(23)         nodeHistory ← nodeHistory.(currentNode)
(24)        if (ADFS(v)) then
(25)            return T
(26)        else
(27)            break
(28)     if h = 0 then
(29)        return F
(30)    else
(31)       h = h − 1; currentPath ← currentPath\(u, v, σ)
(32)       nodeHistory ← nodeHistory\currentNode
(33)     return F
```

Algorithm 3: *ADFS*(*u*).

if 5. Otherwise, the algorithm will set h to be $h - 1$, remove (u, v, σ) from *currentPath*, remove v from *nodeHistory*, and move to another edge from node u.

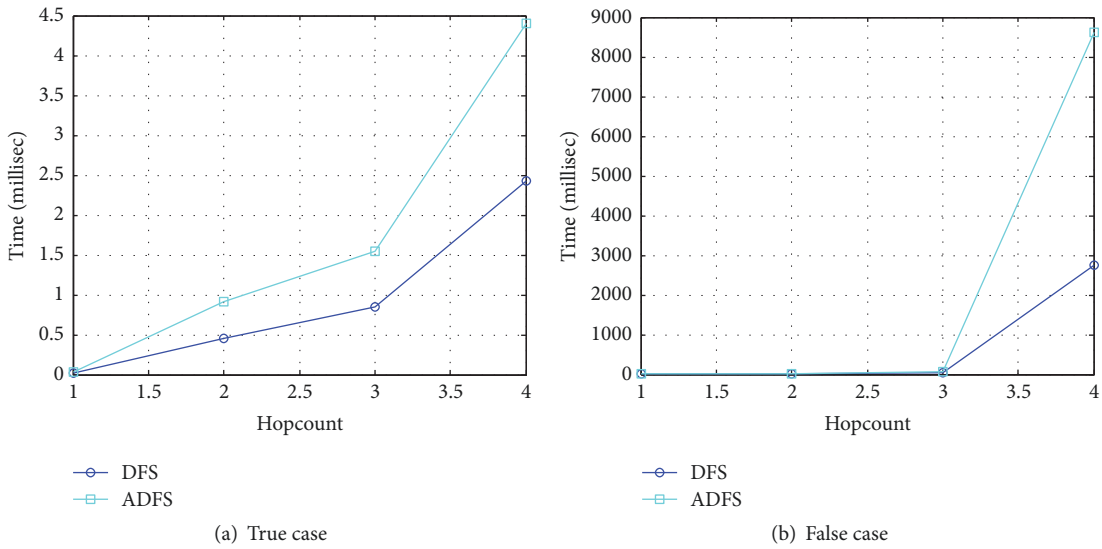
The algorithm will test all paths from *start* to *end*. If any of them fits the access control policy, the algorithm will return T. Otherwise, it may return F as the depth of each path which will be checked is constrained by *hopcount*. We use *min* and *max* to represent the minimum and maximum out degree of node on the social graph, respectively. The time complexity of this algorithm will fall into the range of $O(min^{hopcount})$ and $O(max^{hopcount})$. The check of profile attribute will bring about extra overhead to the algorithm. In the first experiment, we evaluated this overhead which proved it to be acceptable. This experiment is presented in Section 5. So, the path checking algorithm of HAC is effective.

5. Implementation

This section presents the implementation of the path checking algorithm. Five sets of experiments are arranged to test the usability and performance of the algorithm. The researchers implement the algorithm in Java and store the social graph and sample access control policies in MySQL databases. All the experiments are conducted on a machine with 4 GB memory and an Intel quad-core CPU at 3.6 GHz which runs the operating system of an Ubuntu 12.04 image.

5.1. Datasets. When selecting datasets in the organization of the experiments, there are two choices as reported in [35]: public available real datasets and synthetic datasets. As collected from real-world OSN systems, most of the public available datasets do not consider multiple relationship types or attribute information [36]. In order to meet the requirements, this necessary information should be added manually. However, if the dataset is modified, it may no longer present user behaviors [36]. Hence, synthetic dataset is a better choice for us in this evaluation. The researchers generate a random regular graph with n nodes, and each node has a fixed number of edges. Graphs with different nodes and edges can be created by changing parameters n and d according to the need of the experiments.

In the first experiment, to evaluate the effect of attribute evaluation on performance of the algorithm, the researchers test the time of ADFS to make an access control decision and compare it with the one without attribute support described in [36] (DFS). To do this, the researchers comment out the



(a) True case (b) False case

Figure 3: Time of path checking.

code of the profile attribute match. As discussed in [36], n is set to be 1000. On average, main users of Facebook have 173.6 friends as described in [37]. Then, the researchers set that each node has 174 randomly selected neighbors. Only one relationship attribute is considered in this experiment. Five profile attributes, such as user name, gender, career, date of birth, and hometown, are assigned to each node. The user names are set to be different from each other. The gender of each user is randomly selected from male and female. The researchers collected twenty different careers in the career set which are randomly assigned to each user in the social graph. The date of birth is randomly chosen between 1927 and 2007. The hometown of each node is randomly selected from a location set which includes twenty cities. Two sets of experiments are arranged. One set returns the true and the other returns the false. Each 4-hopcount policy runs five times over the two algorithms on 1000 nodes randomly selected from the node set. The average value of those 5000 runs is the final result.

Before the path checking algorithm is invoked, the relationship attributes should be extracted from the path to form a regular expression by the regular expression transformation procedure. This procedure is also called preprocessing. In order to confirm that the policy language is appropriate to be used in an attribute-based access control model, the researchers should make sure that the preprocessing would not take too much time compared with the ADFS algorithm. So, the time of preprocessing is evaluated in the second experiment where the parameters are set to be the same as those in the first experiment.

In the near future, OSNs may support more than one type of relationship. To evaluate if HAC can meet the access control needs of multiple types of relationships in OSNs, in the third experiment, the researchers discuss the variety of time with the increase of the relationship attribute types. The number of relationship attribute types varies from 1 to 8. Other parameters are the same as in the first experiment.

To evaluate how the scale of OSN will impact the performance of the algorithm, in the fourth experiment, the researchers test the variety of time with the number of nodes in the social graph. The number of nodes is set to be 1000, 2000, 5000, and 20000, respectively. The rest of the parameters are the same as in the first experiment.

To evaluate how he density of OSN will impact the performance of the algorithm, in the last experiment, the researchers examine the variety of time with the number of neighbors. The number of neighbors is set to be 100, 174, 200, and 500, respectively. The social graph becomes denser as the number grows. Other parameters are kept consistent with the first experiment.

5.2. Performance. DFS and ADFS algorithms are compared in the first experiment. The researchers consider four policies with different numbers of relationship attributes (hopcount) which varied from 1 to 4. Figure 3 presents the result of the experiment. It takes slightly more time for ADFS to make an access control decision than DFS does, as ADFS takes profile attribute check after finding a qualified relationship attribute. More relationship attributes mean more profile attributes should be checked. So, the time gap of those two algorithms increases as the number of relationship attributes grows. The slight increase of time in ADFS is acceptable, as the use of the profile attribute makes access control policy more powerful. In the false case experiments, it takes significantly more time to finish the path evaluation as more paths in the social graph have to be checked.

Figure 4 presents the result of the second experiment. It shows a comparison of the time of preprocessing and ADFS algorithm. Both of them increase along with the number of *hopcounts*. For each *hopcount*, the time of preprocessing is nearly a tenth of ADFS. This result is acceptable which indicates that the policy language is appropriate to be used in an attribute-based access control model.

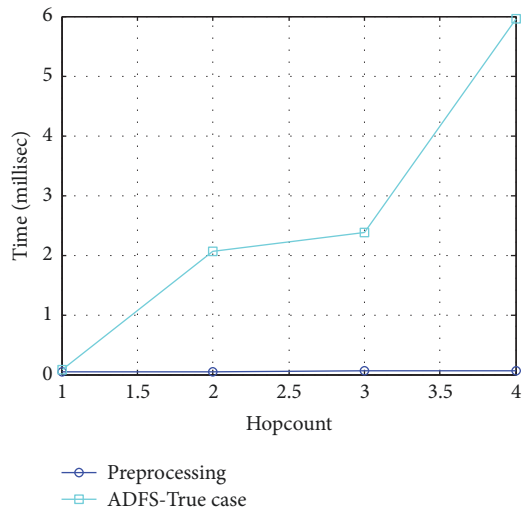
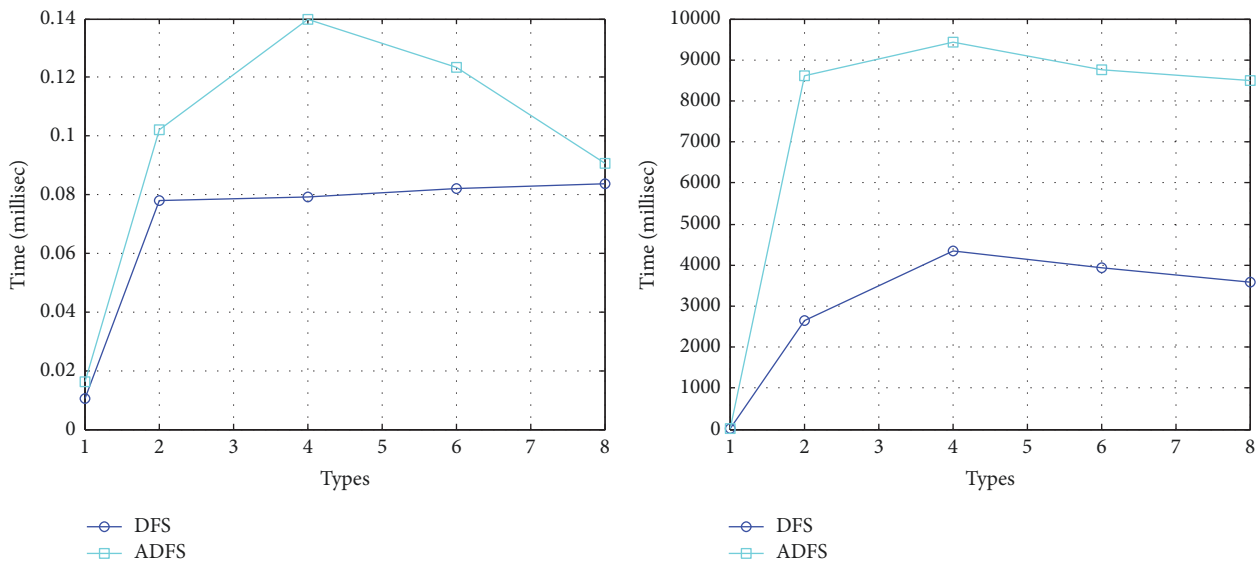


Figure 4: Time of preprocessing.



(a) True case (b) False case

Figure 5: Time of path checking versus types (hopcount = 4).

The result of the third experiment is shown in Figure 5. Since people always tend to share information with friends within a close distance, 4-hopcount policies are used here. In both true and false cases, the time of ADFS to make access control decision increases linearly with the number of relationship attribute types. It reaches the peak value as 4 types of relationship attributes exist in the social graph. A larger number of relationship attributes may not affect the time of policy check as 4-hopcount policies are used in the experiment. The result means that our algorithm will work well with the future social networks as more relationship attribute types will be supported.

Figure 6 shows the result of the fourth experiment. Time of path checking grows with the increase of nodes from 1000 to 20000. In the true case, it takes no more than 1 millisecond to make the access control decision. For the false case, the time it takes becomes longer because all possible paths should be checked.

The result of the last experiment is presented in Figure 7. In both true and false cases, the time of path checking increases as the social graph gets denser. The time grows sharply as the number of degrees exceeds 200. In most social network systems, the number of friends may be approximately 200 [37], so ADFS algorithm is feasible.

6. Comparison

This section discusses several related works of relationship-based access control schemes and compares HAC with [4, 5, 16, 36] (see Table 2).

The first column of Table 2 represents nine characteristics discussed in this section. The next three columns represent

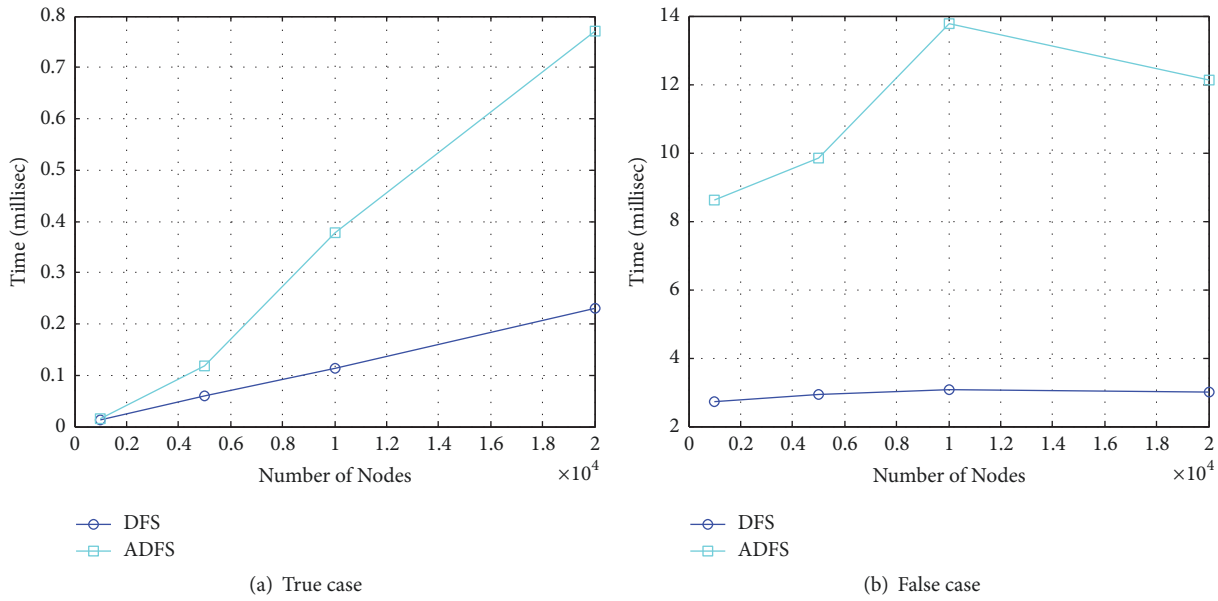


Figure 6: Time of path checking versus number of nodes (hopcount = 4).

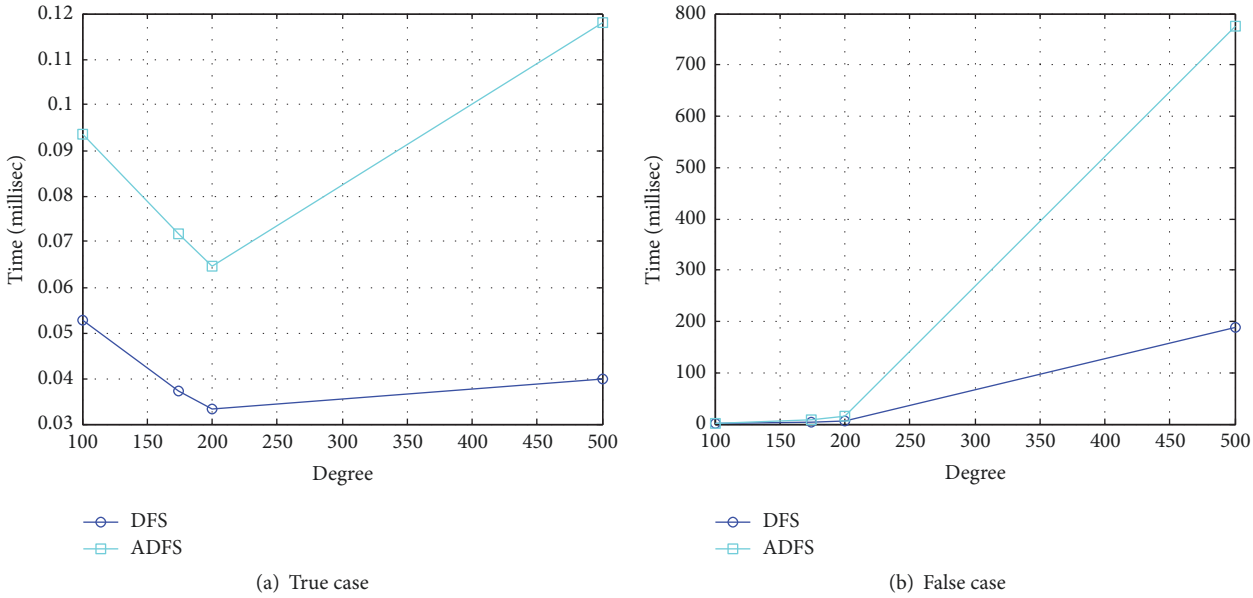


Figure 7: Time of path checking versus degree (hopcount = 4).

the characteristics of the access control schemes discussed above. Characteristics of HAC are listed in the last column.

The scheme in [16] is a formal algebraic access control model for Facebook-style systems. But user attributes and relationships beyond friendship are not supported in this model. OSNs access control models presented in [5, 36] have similar user graphs to HAC. However, these models do not explicitly take into account user attributes.

This work is similar to [4]. Despite its flexibility, UURAC_A [4] is still far from perfect. It does not support specific user attribute. More concretely, its policy specification language can merely figure out common attribute requirements of one or several users on the relationship path, lacking specification ability of different attribute requirements of different users along the path. Additionally, it cannot describe some policies, such as "the adult colleagues of my friend Tom can access the resource", which requires the attribute of my friend "name is Tom" and the attribute of the colleagues of my friend Tom "age > 18". Besides, compared with UURAC_A, HAC is simple and easy to understand. It is easier for users in the OSNs to set up access control policies with HAC.

7. Conclusion

This research proposes an attribute and relationship-based hybrid access control model HAC for OSNs based on two

Table 2: Comparison.

	Fong et al. [16]	UURAC [5, 36]	$UURAC_A$ [4]	HAC
Multiple Relationship Types		√	√	√
User Profile Attributes			√	√
Specific User Attribute				√
User-user Relationship	√	√	√	√
Directional Relationship		√	√	√
Relationship Depth	√	√	√	√
Policy Individualization	√	√	√	√
Attribute Composition	none	none	attributes of user set	attributes of exact user
Relationship Description	ff	path pattern of different types	path pattern of different types	exact type sequence

aspects, including policy language and path checking. The policy language contributes to the literature on ReBAC by allowing users to specify spatial, temporal, and historical based policies with better expressiveness and flexibility. This research also presents several attribute and relationship-based hybrid policies and formally expresses them in the proposed policy language. Path checking algorithm enables users to figure out whether an access request can be satisfied. A prototype is implemented, and several experiments are evaluated to validate the feasibility of the scheme. HAC is advantageous compared with existing OSN access control models in terms of the expressiveness ability of policy language and the evaluation algorithm of access request.

In the future, researchers plan to improve the hybrid policy language to gain better expressiveness ability and support for more relationship types including one-to-many relationship and temporary relationship.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The research activities described in this paper have been conducted within the Research Project "the National Key Research and Development Program of China (2016YFB0801001)" and "General Program of National Natural Science Foundation of China (61672515)".

References

[1] J. Xiong, Y. Zhang, X. Li, M. Lin, Z. Yao, and G. Liu, "RSE-PoW: a role symmetric encryption PoW scheme with authorized deduplication for multimedia data," *Mobile Networks and Applications*, pp. 1–14, 2017.

[2] P. W. L. Fong, "Relationship-based access control: protection model and policy language," in *Proceedings of the 1st ACM Conference on Data and Application Security and Privacy, CODASPY '11*, pp. 191–201, February 2011.

[3] P. W. L. Fong and I. Siahaan, "Relationship-based access control policies and their policy languages," in *Proceedings of the 16th ACM Symposium on Access Control Models and Technologies, SACMAT '11*, pp. 51–60, June 2011.

[4] Y. Cheng, J. Park, and R. Sandhu, "Attribute-aware relationship-based access control for online social networks," in *Proceedings of the IFIP Annual Conference on Data and Applications Security and Privacy*, pp. 292–306, Springer, Berlin, Germany, 2014.

[5] Y. Cheng, J. Park, and R. Sandhu, "A user-to-user relationship-based access control model for online social networks," in *Proceedings of the IFIP Annual Conference on Data and Applications Security and Privacy*, pp. 8–24, Springer, Berlin, Germany, July 2012.

[6] D. Wu, J. Yan, H. Wang, D. Wu, and R. Wang, "Social attribute aware incentive mechanism for device-to-device video distribution," *IEEE Transactions on Multimedia*, vol. 19, no. 8, pp. 1908–1920, 2017.

[7] E. G. Carrie, "Access control requirements for web 2.0 security and privacy," in *Proceedings of the IEEE Web 2.0 privacy and security workship (W2SP '07)*, 2007.

[8] B. Carminati, E. Ferrari, and A. Perego, "Rule-based access control for social networks," in *Proceedings of the Move to Meaningful Internet Systems Workshops, OTM '06*, pp. 1734–1744, Springer, Berlin, Germany, 2006.

[9] B. Carminati, E. Ferrari, and A. Perego, "Enforcing access control in Web-based social networks," *ACM Transactions on Information and System Security*, vol. 13, no. 1, pp. 1–38, 2009.

[10] B. Carminati, E. Ferrari, R. Heatherly, M. Kantarcioglu, and B. Thurainsingham, "A semantic web based framework for social network access control," in *Proceedings of the 14th ACM Symposium on Access Control Models and Technologies, SACMAT 2009*, pp. 177–186, June 2009.

[11] B. Carminati, E. Ferrari, R. Heatherly, M. Kantarcioglu, and B. Thuraisingham, "Semantic web-based social network access control," *Computers & Security*, vol. 30, no. 2-3, pp. 108–115, 2011.

[12] Y. Cheng, J. Park, and R. Sandhu, "Relationship-based access control for online social networks: Beyond user-to-user relationships," in *Proceedings of the Proceedings of the Privacy, Security, Risk and Trust*, pp. 646–655, 2012.

[13] T. Ahmed, F. Patwa, and R. Sandhu, "Object-to-object relationship-based access control: Model and multi-cloud demonstration," in *Proceedings of the 17th IEEE International Conference on Information Reuse and Integration, IRI '16*, pp. 297–304, July 2016.

[14] J. Crampton and J. Sellwood, "Path conditions and principal matching: a new approach to access control," in *Proceedings of the 19th ACM Symposium on Access Control Models and Technologies, SACMAT '14*, pp. 187–198, 2014.

[15] A. Masoumzadeh and J. Joshi, "OSNAC: an ontology-based access control model for social networking systems," in *Proceedings of the 2nd IEEE International Conference on Social*

Computing, SocialCom 2010, 2nd IEEE International Conference on Privacy, Security, Risk and Trust, PASSAT '10, pp. 751–759, August 2010.

[16] P. W. Fong, M. Anwar, and Z. Zhao, "A privacy preservation model for facebook-style social network systems," in *Proceedings of the European Symposium on Research in Computer Security*, pp. 303–320, Springer, Berlin, Germany, September 2009.

[17] P. W. Fong, "Relationship-based access control: Protection model and policy language," in *Proceedings of the 1st ACM Conference on Data and Application Security and Privacy, CODASPY '11*, pp. 191–201, February 2011.

[18] G. Bruns, P. W. Fong, I. Siahaan, and M. Huth, "Relationship-based access control: its expression and enforcement through hybrid logic," in *Proceedings of the second ACM conference on Data and Application Security and Privacy*, pp. 117–124, Feburary 2012.

[19] E. Tarameshloo and P. W. Fong, "Access control models for Geo-Social Computing systems," in *Proceedings of the 19th ACM Symposium on Access Control Models and Technologies, SACMAT '14*, pp. 115–126, June 2014.

[20] E. Tarameshloo, P. W. L. Fong, and P. Mohassel, "On protection in federated social computing systems," in *Proceedings of the 4th ACM Conference on Data and Application Security and Privacy, CODASPY '14*, pp. 75–86, March 2014.

[21] M. Cramer, J. Pang, and Y. Zhang, "A logical approach to restricting access in online social networks," in *Proceedings of the 20th ACM Symposium on Access Control Models and Technologies, SACMAT '15*, pp. 75–86, June 2015.

[22] M. Anwar and P. W. L. Fong, "A visualization tool for evaluating access control policies in facebook-style social network systems," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing, SAC '12*, pp. 1443–1450, Italy, March 2012.

[23] M. Imran-Daud, D. Sánchez, and A. Viejo, "Privacy-driven access control in social networks by means of automatic semantic annotation," *Computer Communications*, vol. 76, pp. 12–25, 2016.

[24] B. Carminati and E. Ferrari, "Privacy-aware collaborative access control in web-based social networks," in *Proceedings of the IFIP Annual Conference on Data and Applications Security and Privacy*, pp. 81–96, Springer, Berlin, Germany, July 2008.

[25] J. Domingo-Ferrer, A. Viejo, F. Sebé, and Ú. González-Nicolás, "Privacy homomorphisms for social networks with private relationships," *Computer Networks*, vol. 52, no. 15, pp. 3007–3016, 2008.

[26] B. Carminati and E. Ferrari, "Enforcing relationships privacy through collaborative access control in web-based social networks," in *Proceedings of the Proceedings of the Collaborative Computing: Networking, Applications and Worksharing*, pp. 1–9, November 2009.

[27] K. B. Frikken and P. Srinivas, "Key allocation schemes for private social networks," in *Proceedings of the Proceedings of the 8th ACM workshop on Privacy in the electronic society*, pp. 11–20, November 2009.

[28] G. Mezzour, A. Perrig, V. Gligor, and P. Papadimitratos, "Privacy-preserving relationship path discovery in social networks," in *Proceedings of the International Conference on Cryptology and Network Security*, pp. 189–208, Springer, Berlin, Germany, December 2009.

[29] M. Xue, B. Carminati, and E. Ferrari, "P3D-privacy-preserving path discovery in decentralized online social networks," in *Proceedings of the Computer Software and Applications*, pp. 48–57, July 2011.

[30] M. Backes, M. Maffei, and K. Pecina, "A Security API for Distributed Social Networks," in *Proceedings of the NDSS*, vol. 11, pp. 35–51, February 2011.

[31] X. Jin, R. Krishnan, and R. Sandhu, "A unified attribute-based access control model covering DAC, MAC and RBAC," in *Proceedings of the IFIP Annual Conference on Data and Applications Security and Privacy*, pp. 41–55, Springer, Berlin, Germany, July 2012.

[32] H.-B. Shen and F. Hong, "An attribute-based access control model for web services," in *Proceedings of the 7th International Conference on Parallel and Distributed Computing, Applications and Technologies, PDCAT '06*, pp. 74–79, December 2006.

[33] E. Yuan and J. Tong, "Attributed based access control (ABAC) for web services," in *Proceedings of the Web Services*, pp. 561–569, July 2005.

[34] X. Li, S. Tang, L. Xu, H. Wang, and J. Chen, "Two-Factor Data Access Control With Efficient Revocation for Multi-Authority Cloud Storage Systems," *IEEE Access*, vol. 5, pp. 393–405, 2017.

[35] B. Carminati, E. Ferrari, and J. Girardi, "Performance analysis of relationship-based access control in OSNs," in *Proceedings of the 13th International Conference on Information Reuse and Integration, IRI '12*, pp. 449–456, August 2012.

[36] Y. Cheng, J. Park, and R. Sandhu, "An access control model for online social networks using user-to-user relationships," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 4, pp. 424–436, 2016.

[37] S. W. Lee and J. Lee, "A comparative study of KakaoStory and facebook: focusing on use patterns and use motives," *Telematics and Informatics*, vol. 34, no. 1, pp. 220–229, 2017.