A Semantic-Aware Ontology-Based Trust Model for Pervasive Computing Environments*

Mohsen Taherian, Rasool Jalili, and Morteza Amini

Network Security Center, Department of Computer Engineering,
Sharif University of Technology, Tehran, Iran
{taherian,m_amini}@ce.sharif.edu,
jalili@sharif.edu

Abstract. Traditionally, to handle security for stand-alone computers and small networks, user authentication and access control mechanisms would be almost enough. However, considering distributed networks such as the *Internet* and *pervasive environments*, these kinds of approaches are confronted with flexibility challenges and scalability problems. This is mainly because open environments lack a central control, and users in them are not predetermined. In such ubiquitous computing environments, issues concerning security and trust become crucial. Adding *trust* to the existing security infrastructures would enhance the security of these environments. Although many trust models are proposed to deal with trust issues in pervasive environments, none of them considers the semantic relations exist among pervasive elements and especially among *trust categories*. Employing *Semantic Web* concepts, we propose a computational trust model based on the *ontology* structure, considering the mentioned semantic relations. In this model, each entity can calculate its trust in other entities and use the calculated trust values to make decisions about granting or rejecting collaborations. Using ontology structure can make the model extendible to encompass other pervasive features such as *context awareness* in a simple way.

1 Introduction

Nowadays, with the immense growth of available data and information which motivates moving toward distributed environments, security of users and information is getting more important than ever. With these distributed environments, existing challenges about security and data integrity in centralized environments, must be investigated more extensively. Many authentication and access control mechanisms have been proposed to deal with security issues in distributed environments. However, by increasing the distribution of information, and the arising open environments such as *pervasive computing environments*, existing security infrastructures are not adequate for new requirements of users from now on [6,14].

* This research is partially supported by Iran Telecommunication Research Center (ITRC).

C. Rong et al. (Eds.): ATC 2008, LNCS 5060, pp. 47–59, 2008.

Pervasive computing environments, as a new generation of computing environments after distributed and mobile computing, were introduced in 1991 with a new look at the future of computing environments. The aim of pervasive computing is to move computers and computing devices to the background and place them in human living environments such that they were hidden from humans. To this aim, computing devices must be designed in small sizes to locate in apartments, walls, and furnitures [15]. In pervasive computing environments, users expect to access resources and services anytime and anywhere, leading to serious security risks and problems with access control as these resources can now be accessed by almost anyone with a mobile device. Adding security to such open models is extremely difficult with problems at many levels. An architecture with a central authority can not be assumed and access control is required for external users. The portable hand-held and embedded devices have severe limitations in their processing capabilities, memory capacities, software support, and bandwidth characteristics. Existing security infrastructures deal with authentication and access control. These mechanisms are inadequate due to the increasing flexibility required by pervasive environments.

Trust, which is similar to the way security is handled in human societies, plays an important role in enhancing security of pervasive environments. However, it is not considered in traditional access control models seriously [7]. Till now, several trust models have been proposed for pervasive environments including computational models and none-computational ones. In a computational trust model, the entity's trust to another one is estimated. On the other hand, the aim of a non-computational trust model is only to find out if an entity is trusted or not. It is worthwhile to note that an entity can trust another one in different categories. For example, the device A may trust the device B in the category of reading a file, but A may give up trusting B in the category of writing a file. The semantic relations exist among pervasive devices and specially trust categories may significantly affect security policies. For instance, if we know a special device belongs to the family of PDAs, and also if we have a subsumption relation between PDAs and mobile devices, we can generalize the security rules defined for mobile devices to this particular device. Semantic relations among trust categories mean the security relevance of categories to each other. For example, If an entity A has a high degree of trust to an entity B in getting a web service, we expect A to have a high degree of trust to B in getting a mail service as a consequence.

None of published trust models for pervasive environments have considered the mentioned semantic relations yet. Employing *ontology* structure propounded in *Semantic Web*, we propose a new trust model for pervasive environments. This model, in addition to being a computational trust model, considers semantic relations among devices and among trust categories. Each entity can calculate its trust degree to other entities and make security decisions based on the calculated trust values. In fact, each entity can accept or reject collaboration with other entities with regard to their trust values. Also, each entity can vote for another entity after a direct collaboration with it. Furthermore, this model satisfies

autonomy which is an important property of pervasive entities. A pervasive device can define its security rules independently using the *SWRL* language [1], which is a semantic language for defining rules on ontology structures. The use of ontology structure, makes the model capable of encompassing other pervasive concepts such as *context awareness* in a simple way.

The rest of the paper is organized as follows; In section 2, previous trust models proposed for pervasive environments are reviewed. The structure of our trust model and its main components are discussed in section 3. Section 4 is devoted to explain the trust inference protocol and updating trust values. Finally, we conclude the paper and introduce some future work in section 5.

2 Related Work

Many trust models have been proposed for distributed environments. A small number of them, such as the one proposed by Abdul-Rahman in [3], were designed with such generality to be applicable in all distributed environments. Other cases concentrated on a particular environment. The trust models for *web-based social networks* [8,9,12] and the ones for *peer-to-peer networks* [10,16] are examples of these trust models. In this section, our review focuses on the trust models have been already suggested for pervasive computing environments. In almost all distributed trust models, there must be some basic services and facilities. *Trust inference* and *trust composition* are examples of such facilities. By trust inference, we mean calculating our belief to a statement based on the believes of some other people to whom we trust. Trust composition is a necessary part of a trust inference algorithm to combine the believes obtained from different sources.

The trust model proposed by Kagal *et al.* in 2001 [13,14] is one of the well-known trust models for pervasive computing environments. This model is not a computational trust model and uses certificates to determine whether an entity is trusted or not. In the Kagal's suggested architecture, each environments is divided into some security domains and for each security domain a *security agent* is leveraged. The security agent is responsible for defining security policies and applying them in the corresponding domain. Interfaces of available services in a domain are also provided by its security agent. When an external user requests a service offered in a domain, he must provide a certificate from the agents which are trusted for the security agent of the domain. Then, he must send its request accompanying the acquired certificates to the security agent. The security agent checks the validity of the certificates and responses the user's request. In fact, the Kagal's trust model is more likely to be a certificate-based access control model. In this model, an entity can be trustworthy or not from the security agent's point of view. An entity is not capable of calculating trust values of other entities and collaboration with inter-domain entities which are trusted for a security agent are not supervised.

Among the existing trust models for pervasive environments, the model proposed by Almenarez *et al.* in [4,5], called PTM[1], is so popular. This trust model is a computational trust model and it is implemented on a wide range of

[1] Pervasive Trust Management.

pervasive devices. Considering two kinds of trust, *direct trust* and *recommendation trust* [4], the architecture of this model is divided into two parts; 1) *belief space*, which assigns an initialize trust value to new arriving entities, and 2) *evidence space*, which updates the trust values of entities with respect to their behaviors over the time. To combine trust values, the weighted average operator (WAO) is used and values in the belief space are presented as fuzzy values. A recommendation protocol is defined to recommend an entity the trust values of other entities. If an entity wishes to collaborate with another one, it uses this protocol to acquire that entity's trustworthy degree. In the first collaboration of an entity, its initial trust value, which is assigned in the belief space, is considered. However, over the time, the entity's trust value changes with respect to the entity's behavior. The implementation of this model is added to security infrastructure of some pervasive devices to enhance their security [2].

The above mentioned approaches present drawbacks for open pervasive environments. Perhaps, the main drawback of them is not taking into account the semantic relations among pervasive devices and among trust categories. We have defined a pervasive trust model based on ontology structure between autonomous entities without central servers. Considering mentioned semantic relations makes the model capable of defining security rules with more flexibility. The model is also simple enough to implement in the very constrained devices which have strict resource constraints.

3 The Trust Model

In our proposed model, in addition to calculating the trust values from each entity to other entities, the semantic relations among pervasive devices and trust categories are considered using an ontology structure. In this section, the basic components of this model are introduced.

3.1 Trust Ontology

In this model, to represent trust relations among pervasive devices, a particular ontology is defined, called *trust ontology*. As known, each ontology O contains a set of concepts (classes) C and a set of properties P. The formal notation of trust ontology is defined as follows:

```
O={C,P}

C={Device, Category, TrustValue, DirectTrust, RecTrust,
   CategoryRelation, RelevanceValue, Time}

P={hasDirectTrust, hasRecTrust, initialTrustValue,
   trustedDevice, trustedCategory, hasTrustValue,
   trustRelated, relatedCategory, hasRelevanceValue,
   updateTime, collaborationNo}
```

Classes of the Trust Ontology. The class *Device* represents the available devices of pervasive environment such as users, sensors and PDAs. The class *Category* includes individuals which represent trust categories, e,g., login access or reading file. In fact, the trust category describes the semantics of a trust relation. The class $TrustValue$ contains the valid values of trust degrees. The float numbers in the range of [0..1] can be an example of these values.

Similar to many other trust models, two kinds of trust are considered in our model. First, *direct trust* which is given by the knowledge of an entitys nature or its past interactions in the physical world, without requesting information from other entities. Second trust type is *indirect trust* or *recommendation trust*. When two entities, unknown to each other, are willing to interact, they can request other entities to give information about the other party. This process of asking other entities and calculating the final trust value from the received answers is called *trust inference*.

To model the trust relations, for both direct trust and recommendation trust, some properties must be defined in the ontology. These properties have some attributes themselves. In Semantic Web languages, such as RDF and OWL, a property is a binary relation; it is used to link two individuals or an individual and a value. However, in some cases, the natural and convenient way to represent certain concepts is to use relations to link an individual to more than just one individual or value. These relations are n-ary relations. For instance, it might be required to represent properties of a relation, such as our certainty about it, relevance of a relation, and so on. One solution to this problem is creating an individual representing the relation instance itself, with links from the subject of the relation to this instance and with links from this instance to all participants that represent additional information about the instance. In the class definition of the ontology, an additional class is required to include instances of this n-ary relation itself. Classes $DirectTrust$ and $RecTrust$ are of such classes.

One of the main features of our model is considering semantic relations among trust categories. Like direct trust and indirect trust relation, the semantic relation among trust categories is n-ary relation. The class $CategoryRelation$ is defined to include instances of this n-ary relation.

The class $RelevanceValue$ defines the valid values for the relevance values among trust categories. Finally, the class $Time$ characterizes the time values in the model. Individuals of this class are used to hold the time of inferring trust values.

Properties of the Trust Ontology

- **initialTrustValue:** An instance of this property assigns to a new arriving entity an initial trust value. This assignment is done by special agents called *trust managers* which are described in the next section. One way is to assign different initial trust values to the new entity corresponding to different trust categories. Another way is to assign only one initial trust value for all trust categories. Concentrating on simplicity of the model, the latter one is considered in this paper. The criteria of assigning this value is dependent to

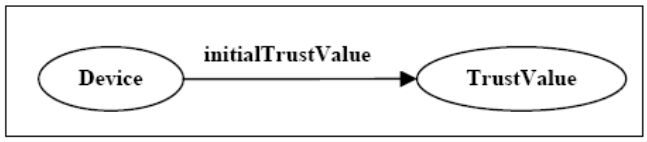


Fig. 1. *initialTrustValue* property

the policies of the trust manager. The schema of this property is shown in Fig. 1.

- **hasDirectTrust:** When an entity collaborates with another one, it gains a degree of trust about that entity. This type of trust is called *direct trust*. Since this relation is not a binary relation and it has some attributes, the pattern described before to define n-ary relations is used. Fig. 2 shows the schema of *hasDirectTrust* property. The class *DirectTrust* includes instances of the relation. The property *trustedDevice* determines the device that the trust relation is established with. The class $\mathbb{N}$ includes the natural numbers and the property *collaborationNo* identifies number of collaborations which are already done between these two entities. The property *hasTrustValue* assigns a trust value to the trust relation and the property *trustedCategory* characterizes the trust category in which the trust relation is set up.
- **hasRecTrust:** If an entity wants to begin a collaboration with another one, it may want to know the opinions of other entities about the other party. The trust value derived in this way is called *indirect trust* or *recommendation trust*. Like *hasDirectTrust*, this relation is also an n-ary relation. The

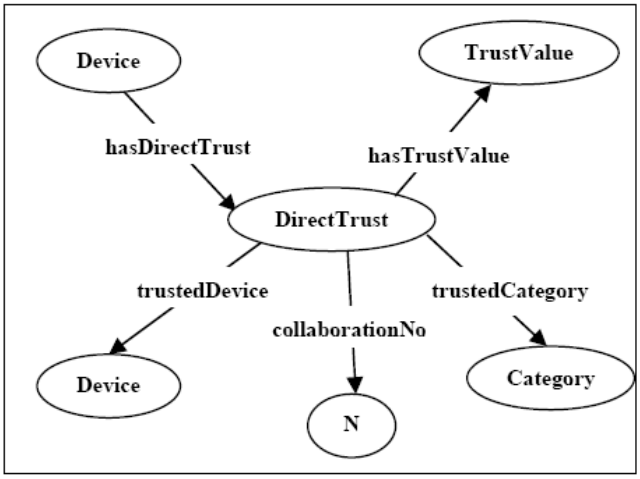


Fig. 2. *hasDirectTrust* property

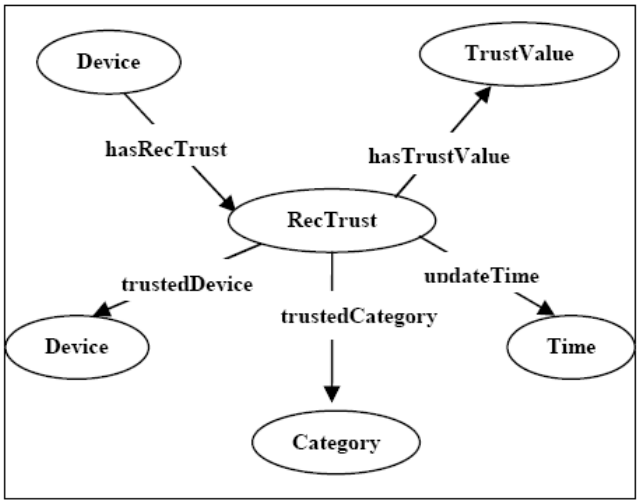


Fig. 3. *hasRecTrust* property

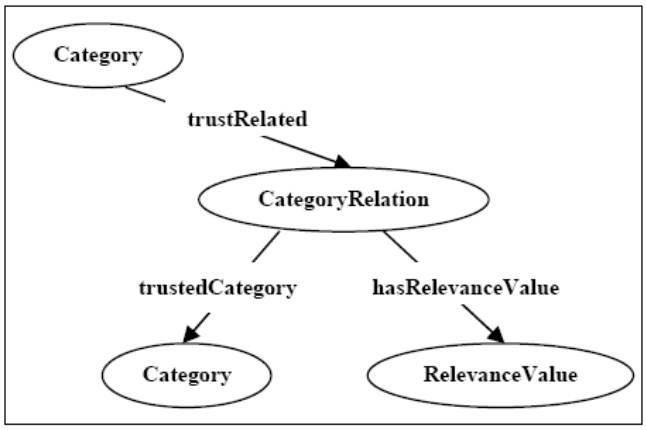


Fig. 4. *trustRelated* property

schema of this relation is illustrated in Fig. 3. The class *RecTrust* includes individuals of the relation. All attributes of this relation is similar to the direct trust relation except that instead of property *collaborationNo*, the property *updateTime* is added. This property determines the time of last trust inference. Details of the inference algorithm is discussed in section 4.

– **trustRelated:** The semantic relation between two categories of trust is modeled with this property. According to Fig. 4, the class *CategoryRelation* contains individuals of the relation itself. The property *trustedCategory*

represents the related category and the property $hasRelevanceValue$ assigns a relevance value to this relation. The class $RelevanceValue$ consists of the valid values for this relevancy.

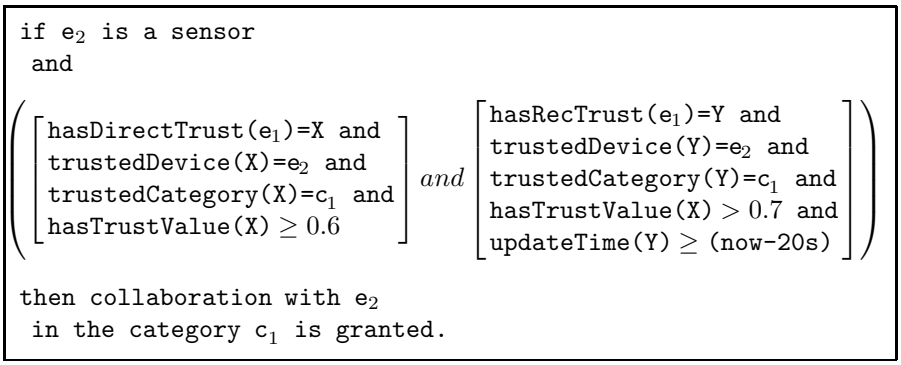
3.2 Trust Manager

A pervasive environment, is divided into different domains and each domain has a *trust manager*. The trust manager is responsible for assigning the initial trust values, defining semantic relations among trust categories, defining a hierarchy of devices, and holding the *base trust ontology*. The hierarchy of devices can be defined with using the $subClassOf$ property of an ontology. The base trust ontology, contains the relations among trust categories and hierarchy of pervasive devices. These relations can be defined by the security manager of each domain.

When a new entity enters a domain, it sends a message to the domain's trust manager and declares its physical specifications. According to these specifications and its own policies, the trust manager finds out if this new entity is an individual of the class *Device* or one of its subclasses. Then, an initial trust value is assigned to the entity. After updating the ontology, trust manager sends the file of ontology to the new entity. Thus, all entities receive the domain's base trust ontology when they enter the domain. The trust manager also broadcasts a new massage to update the ontologies of already existing entities. The structure and format of the alert message are out of this paper's scope.

3.3 Security Rules

The autonomy of pervasive devices is a basic property of them. In our model, each entity is independent of the other ones in defining security rules. The policies are described in the *SWRL* language, a language to process and to query the ontologies which are written in the *OWL* language. Although formats and patterns of defining security rules are not explained here, an example is given to understand the concept. Note that instead of SWRL syntax, a pseudocode syntax is used in order to make it more legible. Suppose that entity e_1 begins a collaboration with entity e_2 in the trust category c_1. A security rule for e_1 can be represented as:



Before beginning the collaboration, e_1 looks up in its security rules to find the matching rules. In this case, the mentioned security rule is matched with the collaboration properties. If the found rules are satisfied, the entity begins the collaboration. If no rule is matched with an interaction, granting or denying the interaction can be decided according to the security policies. In this approach, a possible problem is conflicting the rules, matched with a collaboration. Conflict resolution is out of the scope of this paper.

4 Trust Inference

In any trust model, one of the main parts is the algorithm of inferring trust. Trust inference means calculating indirect trust value (or recommendation trust value). In addition to indirect trust, the way in which direct trust values are created is important too. In this section, trust inference protocol and the method of updating both direct and indirect trust values are discussed. The approach of applying semantic relations among trust categories is also described in the following subsections.

4.1 Trust Inference Protocol

Suppose that device e_1 does not have any information about entity e_2 and it is willing to interact with e_2. Consider that this type of interaction needs a degree of trust in the trust category c_1. Now, e_1 needs to derive the trust value of e_2 by asking other entities. Thus, e_1 broadcasts a query message to other entities. The query part of this message is expressed in the SWRL language. Although the format and structure of messages are not discussed in this paper, an example of broadcasting query is illustrated below using pseudocode.

```
hasDirectTrust(x)=X and trustedDevice(X)=e2 and
trustedCategory(X)=c1 and hasTrustValue(X)= ?
```

In the above query, x matches with each entity that receives the message. It is clear that to answer the sender, address of sender must be located in the message. Also, a timeout must be declared by sender to ignore indefinite waiting. Each entity which has a direct trust to e_2 in the category c_1, replies e_1. After finishing the declared timeout, e_1 calculates the derived trust value with respect to delivered answers. The equation 1 shows this operation.

$$T_{infer}(e_1, e_2, c_1) = \frac{\sum_{i=1}^{n} T(e_i, e_2, c_1) \times T(e_1, e_i, c_1)}{T(e_1, e_i, c_1)} \quad (1)$$

The entities who reply e_1 are denoted by e_i. $T(e_i, e_2, c_1)$ is the value of direct trust from entity e_i to e_2 in the trust category c_1 and $T(e_1, e_i, c_1)$ is the direct trust value from e_1 to e_i in the trust category c_1. Considering trust value of

sender to repliers causes that answers from more reliable entities, having more impact on the inferred trust value. Note that if e_1 does not have a direct trust to e_i (e_1 has not done any interaction with e_i in the trust category c_1 yet.), it considers the initial trust value of e_i ($initialTrustValue(e_i)$) instead of $T(e_1, e_i, c_1)$. As it is mentioned before, this initial trust value is assigned by the trust manager. It is obvious that the inferred trust value will be located in the valid range of trust values defined by class $TrustValue$ of trust ontology. After computing the inferred trust value, e_1 updates its ontology. The time of inferring trust (t_{infer}) will be also located in the ontology using *updateTime* property. Updating the trust ontology of e_1 includes the following items:

```
hasRecTrust(e1)=X and trustedDevice(X)=e2 and
trustedCategory(X)=c1 and updateTime(X)=t_infer and
hasTrustValue(X)=T_infer(e1,e2,c1)
```

In our inference method, the weighted average operator (WAO) is used to combine the trust values. Although other distributed trust models use alternative operators to combine trust values which may cause getting more accurate results, like the *consensus operator* [11] used in [12], for pervasive devices which have considerable limitations on battery life, memory capacities, size, and performance, the simplicity is preferred to accuracy.

4.2 Updating the Trust Values

To update indirect trust values, different approaches can be used. One way is that each entity recalculates its trust value to another entity after a predefined time periods. Another way is to derive the trust value whenever a rule consisting the time constraint is fired up. A combination of these two approaches can be used too. In a pervasive domain, the security manager can choose one of the above methods.

Now, the question is that how direct trust values can change. In this model, after completing a transaction, entities can vote for each other. The new direct trust value can be computed by the equation 2.

$$T_{new}(e_1, e_2, c_1) = \frac{T_{old}(e_1, e_2, c_1) \times collaborationNo + vote(e_1, e_2, c_1)}{collaborationNo + 1} \quad (2)$$

In this equation, $T_{old}(e_1, e_2, c_1)$ represents the direct trust value from e_1 to e_2 in the category c_1 before beginning the transaction. The term $vote(e_1, e_2, c_1)$ represents the opinion of e_1 about e_2 in the category c_1 after completing the transaction and $collaborationNo$ is the number of transactions between e_1 and e_2 which have taken place in the category c_1 before this transaction. $T_{new}(e_1, e_2, c_1)$ is the new direct trust value from e_1 to e_2 in the category c_1. Updating the trust ontology of e_1 includes the following items:

hasDirectTrust(e_1)=X and trustedDevice(X)=e_2 and
trustedCategory(X)=c_1 and
collaborationNo(X)=collaborationNo(X)+1 and
hasTrustValue(X)=T_{new}(e_1,e_2,c_1)

Note that the new direct trust values can be alerted to the trust manager to take these values into account in its later decisions.

4.3 Semantic Relations among Trust Categories

Defining the *trustRelated* property in the trust ontology provides this possibility for the trust model to represent semantic relations among trust categories. Considering these relations, provide us security rules with more flexibility and higher security level. For example, assume that e_1 wishes to begin an interaction with e_2 which requires satisfaction of the following security rule:

hasDirectTrust(e_1)=X and trustedDevice(X)=e_2 and
trustedCategory(X)=c_1 and hasTrustValue(X) ≥ 0.6

Now, suppose that e_1 would collaborate with e_2 if it has the same degree of trust to e_2 in other categories which are related to c_1 with the relevancy value of 0.8. The security rule to support this requirement is shown below:

hasDirectTrust(e_1)=X and trustedDevice(X)=e_2 and
trustedCategory(X)=Y and trustRelated(Y)=Z and
relatedCategory(Z)=c_1 and hasRelevanceValue(Z) > 0.8
hasTrustValue(X) ≥ 0.6

5 Conclusions and Future Works

In this paper, we have introduced a new semantic-aware trust model for pervasive environments based on ontology concepts. A standard ontology, called *trust ontology*, is defined to support trust in pervasive environments. The trust ontology is represented with the OWL language and queries on the ontology can be expressed in existing rule languages such as SWRL. Using the ontology structure, the model provides a standard trust infrastructure for pervasive devices.

Using the weighted average operator (WAO), a simple inference protocol is proposed to calculate the indirect trust values. For pervasive devices which have significant limitations on battery life, memory capacities, size, and performance, the simplicity of inference protocol offers many benefits to calculate the indirect trust values. Another advantage of the model is taking into account the autonomy of pervasive devices. Each device can define its private security rules independent of other devices. There exist such flexibility for devices to employ both direct and indirect trust values in defining their security policies.

Considering the semantic relations among trust categories and defining hierarchical structure of pervasive devices, provides us more flexibility to define security rules. With this feature, a wide range of security policies can be expressed in a simple way. Adding more attributes of pervasive environments such as *context-awareness* is possible with making a little extension to the model. For example, suppose that we want to add a context variable such as the *location*. The property *hasLocation* and a class *validPlaces* can be defined in the trust ontology to support this context variable. New security rules can use this new concept to enhance their expressiveness.

Future work includes defining the structure of messages and patterns of security rules. Moving toward implementing this model on pervasive devices like PDAs and evaluating the performance impacts are also in our future plans.

References

1. Swrl: A semantic web rule language combining owl and ruleml, `http://www.w3.org/Submission/SWRL`
2. Ubisec project, pervasive trust management model (ptm), `http://www.it.uc3m.es/~florina/ptm`
3. Abdul-Rahman, A., Hailes, S.: A distributed trust model. In: New Security Paradigms Workshop, pp. 48–60. ACM Press, New York (1998)
4. Almenarez, F., Marin, A., Campo, C., Garcia, C.: Ptm: A pervasive trust management model for dynamic open environments. In: First Workshop on Pervasive Security, Privacy and Trust PSPT (2004)
5. Almenarez, F., Marin, A., Diaz, D., Sanchez, J.: Developing a model for trust management in pervasive devices. In: Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshop (PERCOMW 2006) (2006)
6. Blaze, M., Feigenbaum, J., Keromyts, A.D.: The role of trust management in distributed systems security. In: Secure Internet Programming, pp. 185–210 (1999)
7. English, C., Nixon, P., Terzis, S., McGettrick, A., Lowe, H.: Dynamic trust models for ubiquitous computing environments. In: Ubicomp Security Workshop (2002)
8. Golbeck, G.A.: Computing and Applying Trust in Web-Based Social Networks. PhD thesis, University of Maryland (2005)
9. Golbeck, G.A., James, H.: Inferring binary trust relationships in web-based social networks. ACM Transactions on Internet Technology 6(4), 497–529 (2005)
10. Griffiths, N., Chao, K.M., Younas, M.: Fuzzy trust for peer-to-peer systems. In: P2P Data and Knowledge Sharing Workshop (P2P/DAKS 2006), at the 26th International Conference on Distributed Computing Systems (ICDCS 2006), Lisbon, Portugal, pp. 73–73. IEEE Computer Society Press, Los Alamitos (2006)
11. Josang, A.: The consensus operator for combining beliefs. Artificial Intelligence Journal 142(1-2), 157–170 (2002)
12. Josang, A., Hayward, R., Pope, S.: Trust network analysis with subjective logic. In: Australasian Computer Science Conference (ACSC 2006), Hobart, Australia, pp. 85–94 (2006)
13. Kagal, L., Finin, T., Joshi, A.: Trust-based security in pervasive computing environments. IEEE Computer 34(12), 154–157 (2001)

14. Kagal, L., Finin, T., Joshi, A.: Moving from security to distributed trust in ubiquitous computing environments. IEEE Computer (2001)
15. Satyanarayanan, M.: Pervasive computing: Vision and challenges. IEEE Personal Communications 8(4), 10–17 (2001)
16. Wang, Y., Vassileva, J.: Trust and reputation model in peer-to-peer networks. In: 3rd International Conference on Peer-to-Peer Computing (P2P 2003), pp. 150–157. IEEE Computer Society, Los Alamitos (2003)