PTO: A Trust Ontology for Pervasive Environments

Mohsen Taherian, Rasool Jalili, Morteza Amini
Computer Engineering Department
Sharif University of Technology

{*taherian@ce., jalili@, m_amini@ce.*}*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 confront with flexibility challenges and scalability problems. This is mainly because of that 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 have been proposed to deal with trust issues in pervasive environments, none of them considered the semantic relations among pervasive elements and specially 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 about other entities and use the calculated trust values to make decisions about granting or rejecting interactions. The use of ontology structure can make the model extendible to encompass other pervasive features such as *context awareness* in a simple way.**

I. Introduction

Pervasive computing environments, as a new generation of the computing environments after distributed and mobile computing ones, were introduced in 1991 with a new look at the future of the computing environments. In the 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 in this case since access control is required for external users. The portable handheld and embedded devices have severe limitations in their processing capabilities, memory capacities, software supports, 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, play an important role in enhancing security of pervasive environments. However, it is not considered in traditional access control models seriously [1]. Till now, several trust models have been proposed for pervasive environments including computational models and none-computational ones. In a computational trust model, the values of the entity's trust to another one are estimated. On the other hand, the aim of a non-computational trust model is only to find out whether 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 among pervasive devices and also dependencies among 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. Dependencies among trust categories mean the security relevance of categories to each others. For example, if an entity A has a high degree of trust to an entity B in getting a web service, we may expect A to have a high degree of trust to B in getting a mail service as a consequence.

Based on the knowledge of the authors, 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 may accept or reject interaction with other entities with regard to their trust values. Also, each entity can vote for another entity after a direct interaction with it. Furthermore, this model satisfies autonomy which is an important property of pervasive entities. A device in pervasive environment can define its security rules independently using the *SWRL* language [2], which is a semantic language for defining rules on ontology structures. Semantic Web Rule Language (SWRL) enables Horn-like rules to be combined with an OWL[1] knowledge base.

The rest of the paper is organized as follows; In section II, previous trust models proposed for pervasive environments are reviewed. Section III describes the proposed trust ontology for pervasive environments. The structure of our trust model and

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

[1]Ontology Web Language

978-0-7695-3096-3/08 $25.00 © 2008 IEEE
DOI 10.1109/WAINA.2008.39

its main components are discussed in section IV. Section V is devoted to a general evaluation of our trust model. Finally, we conclude the paper and introduce some future work in section VI.

II. 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 usually concentrated on a particular environment. The trust models for *web-based social networks* [4]–[6] and the ones for *peer-to-peer networks* [7], [8] are examples of these trust models. In this section, our focus is 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/or 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 beliefs of some other people who are trusted for us. Trust composition, on the other hand, is a necessary part of the trust inference algorithm which combines the beliefs obtained from different sources.

The trust model proposed by Kagal *et al*. in 2001 [9], [10] is one of the 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 environment 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 one of 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 and no entity has a degree of trust about other entities. Thus, interaction of entities with other ones in the same domain is not supervised.

Among the existing trust models for pervasive environments, the model proposed by Almenarez *et al*. in [11], [12], called *PTM*[2], is very popular. This trust model is a computational trust model and it is implemented on a wide range of pervasive devices. Considering two kinds of trust, *direct trust* and *recommendation trust* [11], the architecture of this model is divided into two parts; 1) *belief space* which assigns an initial 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 ones. If an entity wishes to interact with another one, it uses this protocol to acquire that entity's trustworthiness degree. In the first interaction 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. It is worthy to note that, the implementation of this model is added to security infrastructure of some pervasive devices to enhance their security [13].

All the mentioned approaches suffer from some common drawbacks in pervasive environments. Perhaps, the main drawback is not taking into account the semantic relations among pervasive devices and trust categories. We have defined a pervasive trust model based on ontology structure between autonomous entities. 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.

III. PTO: Pervasive Trust Ontology

In our proposed model, in addition to calculating the trust values from each entity to the others, the semantic relations among pervasive devices and relations among trust categories are considered using an ontology structure. Ontologies include computer-usable definitions of basic concepts in the domain and the relationships among them. They encode knowledge in a domain and also knowledge that spans domains. In this model, to represent trust relations among pervasive devices, a particular ontology is defined, called *PTO* (Pervasive Trust Ontology). As known, each ontology O contains a set of concepts (classes) C and a set of properties P. A *class* is a collection of individuals and a *property* is a collection of relationships between individuals (and data). A property that relates an individual to another individual is called *object property* and a property that maps an individual to a data literal is called *datatype property*. Like a mathematical function, a property has a domain and a range. While both domain and range of object properties are ontology classes, the range of datatype properties are data literals such as integer, time, etc. The OWL, which is recommended by W3C[3], is used to describe the trust ontology. The purpose of OWL is to provide an XML vocabulary to define classes, properties and their relationships. The benefit of OWL is that it facilitates much greater degree of inference than you get with RDF Schema. The formal notation of trust ontology is as follows:

```
PTO: Pervasive Trust Ontology ={C, OP, DP}

C: OWL Classes =
{Device, Category, DirectTrust, RecTrust,
CategoryRelation, InteractionAuthority}
```

[2]Pervasive Trust Management

[3]The World Wide Web Consortium (W3C) is the main international standards organization for the World Wide Web. (http://www.w3c.org).

```
OP: Object Properties =
{hasDirectTrust, hasRecTrust, trustedDevice,
trustedCategory, trustRelated, canInteract,
interactionDevice, interactionCategory}

DP: Datatype Properties =
{initialTrustValue, hasTrustValue,
hasRelevanceValue, interactionNo, updateTime}
```

In the rest of this section, the classes and properties of PTO are briefly introduced.

A. 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.

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 information 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 to create 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 dependencies among trust categories. Like direct trust and indirect trust relation, this relation among trust categories is an n-ary relation. The class `CategoryRelation` is defined to include instances of this n-ary relation. Finally, to indicate authorized interactions the class `InteractionAuthority` is defined.

B. Properties of the Trust Ontology

initialTrustValue: An instance of this datatype property assigns an initial trust value to a new arriving entity. This assignment is done by special agents called *trust manager* 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 are dependent on the policies of the trust manager.

Notice that all the trust values in the PTO are assumed to be float numbers in range $[1..10]$. This constraint can be applied on datatype properties using the *property restriction* concept.

hasDirectTrust: When an entity interacts 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. 1 shows the schema of `hasDirectTrust` property. In this figure, the properties which cooperate to establish a direct trust relation are illustrated. The class `DirectTrust` includes instances of the relation. The property `trustedDevice` determines the device at the other side of the trust relation. The property `interactionNo` identifies number of interactions which have been already done between these two entities. The property `hasTrustValue` assigns a trust value in range of $[1..10]$ to the trust relation, and the property `trustedCategory` characterizes the trust category in which the trust relation is set up.

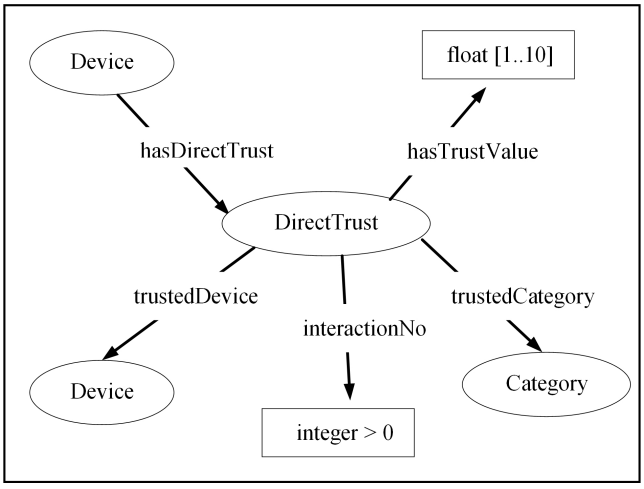

Fig. 1. $hasDirectTrust$ property

hasRecTrust: If an entity wants to begin a interaction 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 schema of this relation is illustrated in Fig. 2. The class `RecTrust` includes individuals of this relation. All attributes of this relation is similar to the direct trust relation except that instead of property `interactionNo`, the property

`updateTime` is considered. This property determines the time of last trust inference. Details of the inference algorithm are discussed in section IV.

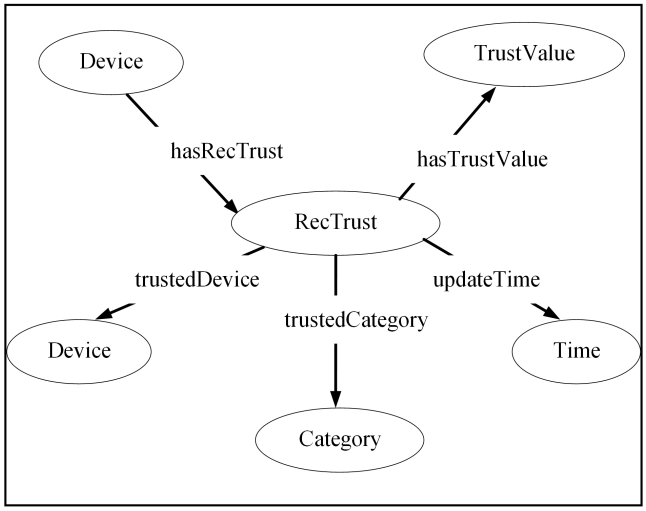

Fig. 2. $hasRecTrust$ property

trustRelated: The dependency between two categories of trust is modeled with this property. According to Fig. 3, the class `CategoryRelation` contains individuals of the relation. The property `trustedCategory` represents the related category and the property `hasRelevanceValue` allocates a float value in range of $[0..1]$ for the relevance degree.

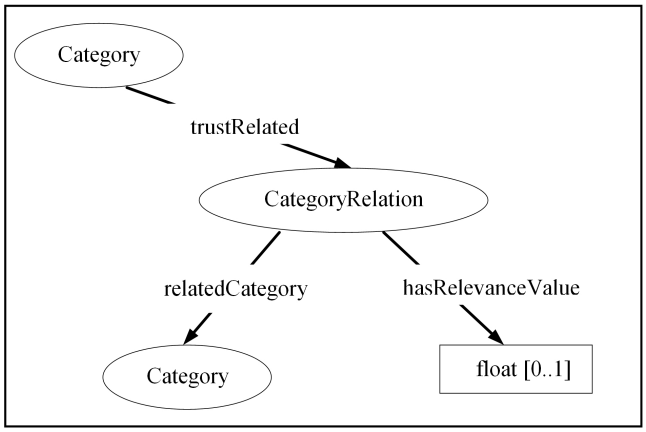

Fig. 3. $trustRelated$ property

canInteract: This property indicates that if interaction of a device with another one in a particular category is authorized or not. With regard to Fig. 4, the class `InteractionAuthority` contains individuals of the relation. The property `interactionDevice` and `interactionCategory` respectively represent the device and the category which the interaction will be granted to.

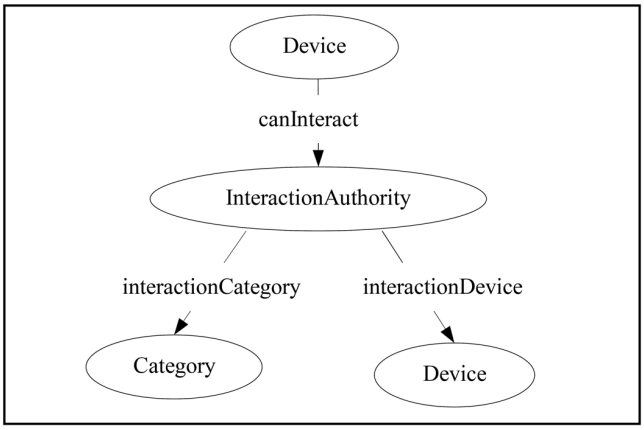

Fig. 4. $canInteract$ property

IV. The Trust Model

Based on PTO, a trust model is proposed to manage trust relations in a pervasive environment. The components of this model can be added to the pervasive devices to enhance security of interactions among them. A simple algorithm is suggested to infer trust values and update them. In this section, components of the proposed trust model are explained in detail.

A. Trust Manager

A pervasive environment, is divided into some 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 the devices, and holding the *base trust ontology*. The hierarchy of devices can be defined with using the `subClassOf` property of the ontology. The base trust ontology contains the relations among the trust categories and the 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 it is one of the `Device` subclasses. Then, an initial trust value is assigned to the entity. After updating the ontology, trust manager sends the file of the 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 the format of the alert message are out of the paper scope.

B. Security Rules

The autonomy of pervasive devices is one of their basic properties. In our model, each entity is independent of the other ones in defining security rules. The policies are described

in the *SWRL* language, which is a language to process and to query the ontologies which are written in the *OWL* language.

We use a simple pattern to define security rules. At the left side of the security rules, using all of the PTO classes and properties except the ones illustrated in Fig. 3 are allowed. The predefined classes, properties, and individuals of OWL and SWRL, such as `rdfs:subClassOf` and `swrlb:greaterThan`, are also acceptable. In this pattern, the right side of the rules can be only in the following form:

```
canInteract(d1, x) ∧
interactionDevice(x, d2) ∧
interactionCategory(x, c)
```

Although the above constraint reduces the flexibility of defining rules, it makes the trust model decidable. In the evaluation section, the effect of this constraint on decidability of the model is analyzed. An example of a security rule is provided below:

```
hasDirectTrust(pc1,?x) ∧
trustedDevice(?x,mobile1) ∧
trustedCategory(?x, execute-process) ∧
hasTrustValue(?x, ?z) ∧
swrlb:greaterThan(?z, 7) ∧
swrlx:createOWLThing(?y, ?x)
 →
canInteract(pc1, ?y) ∧
InteractionAuthority(?y) ∧
interactionDevice(?y, mobile1) ∧
interactionCategory(?y, execute-process)
```

One points in the mentioned example rule must be cleared. In the right side of SWRL rules, creation of a new individual is not allowed. Thus, an additional package, called `swrlx.owl` is added to PTO. This package has a property called `createOWLThing` which is used at the left side of a rule to create individuals which are needed at the right side of that rule. Then, at the right side, these created individuals can be assigned to the corresponding OWL classes. With regard to the pattern we just defined for the security rules, only one individual of the class `InteractionAuthority` must be cerated in a rule.

Before beginning the interaction, a device looks up in its security rules to find the matching rules. If the found rules are satisfied, the entity begins the interaction. Otherwise, the interaction would be denied.

C. Trust Inference

In any trust model, one of the main parts is the algorithm of inferring trust. Trust inference means calculating indirect trust values (or recommendation trust values). In addition to indirect trust, the way in which direct trust values are created is important too. 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. It is clear that to answer the sender, address of the sender must be located in the message. Also, a timeout must be declared by the 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. Equation (1) shows this operation.

$$T_{infer}(e_1, e_2, c_1) = \frac{\sum_{i \neq 1,2} T(e_i, e_2, c_1) \times T(e_1, e_i, c_1)}{\sum_{i \neq 1,2} T(e_1, e_i, c_1)} \quad (1)$$

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 the sender to the repliers causes that the 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 too. The time of inferring trust (t_{infer}) will be also stored in the ontology using `updateTime` property.

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 reaching more accurate results, like the *consensus operator* [14] used in [6], for pervasive devices which have considerable limitations on battery life, memory capacities, size, and performance, the simplicity factor is preferred.

D. 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 consists 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 be changed. 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).

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

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 $intNo$ 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 updating the direct trust value and the interaction

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

V. Model Evaluation

In this section, we first discuss on the decidability of the inference on the security rules of the model. Then, we try to highlight the advantages of this trust model over previous ones. As mentioned before, after firing each security rule, at most three predicate may be asserted in the rule base of the ontology. These predicates are corresponded to the properties `canInteract`, `interactionDevice`, `interactionCategory`. Suppose that in a pervasive domain, n is the number of devices and m is the number of categories. According to the domain and range of these properties and with any number of security rules, the $(n-1) \times m$ is the maximum number of predicates that can be asserted into the rule base of each device after inference process. On the other hand, with this constraint on the security rules which the right side of them are in a predefined format and no predicate of the right side can be used in the left side, there will be no loop in the rule base. These reasons demonstrate that the inference problem in the trust model is decidable.

Using ontology to model trust relations among pervasive devices makes the model capable of including semantic relations among pervasive devices. These semantic relations can be used in defining security policies. Combining OWL and semantic rules which is the idea of SWRL rules provide us a faster inference than other semantic rule languages. There are some algorithms to inference on SWRL rules which a particular one of them is *rete* algorithm [15].

Considering the dependencies 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. Consequently, new security rules can use this new concept to enhance their expressiveness.

In addition to considering semantic relations, our trust model is a computational trust model against the Kagal's trust model [9], [10]. Using 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 considering the autonomy of pervasive devices. Each device can define its private security rules independent of the others. This provides such flexibility for devices to employ both direct and indirect trust values in defining their security policies.

VI. Conclusions and Future Work

In this paper, we have introduced a new semantic-aware trust model for pervasive environments based on ontology concepts. A standard ontology, called *PTO*, 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. Based on PTO, a trust model is proposed to manage trust relations in a pervasive domain. This model calculates trust values for pervasive devices which can be used in granting or denying interactions among them. Each device has its own ontology and it can define its policies independent of other devices. Satisfying the matching rules is the condition of beginning interactions. Asserting some constraints on security rules made the inference problem of the trust model decidable.

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] C. English, P. Nixon, S. Terzis, A. McGettrick, and H. Lowe, "Dynamic trust models for ubiquitous computing environments," in *Ubicomp Security Workshop*, 2002.

[2] "Swrl: A semantic web rule language combining owl and ruleml," *http://www.w3.org/Submission/SWRL*.

[3] A. Abdul-Rahman and S. Hailes, "A distributed trust model," in *New Security Paradigms Workshop*. ACM Press, 1998, pp. 48–60.

[4] G. A. Golbeck, "Computing and applying trust in web-based social networks," Ph.D. dissertation, University of Maryland, 2005.

[5] G. A. Golbeck and J. Hendler, "Inferring binary trust relationships in web-based social networks," *ACM Transactions on Internet Technology*, vol. 6, no. 4, pp. 497–529, 2005.

[6] A. Josang, R. Hayward, and S. Pope, "Trust network analysis with subjective logic," in *Australasian Computer Science Conference (ACSC2006)*, Hobart, Australia, 2006, pp. 85–94.

[7] N. Griffiths, K. M. Chao, and M. Younas, "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: IEEE Computer Society, 2006, pp. 73–73.

[8] Y. Wang and J. Vassileva, "Trust and reputation model in peer-to-peer networks," in *3rd International Conference on Peer-to-Peer Computing (P2P 2003)*. IEEE Computer Society, 2003, pp. 150–157.

[9] L. Kagal, T. Finin, and A. Joshi, "Trust-based security in pervasive computing environments," *IEEE Computer*, vol. 34, no. 12, pp. 154–157, 2001.

[10] L. Kagal, T.Finin, and A.Joshi, "Moving from security to distributed trust in ubiquitous computing environments," *IEEE Computer*, 2001.

[11] F. Almenarez, A. Marin, C. Campo, and C. Garcia, "Ptm: A pervasive trust management model for dynamic open environments," in *First Workshop on Pervasive Security, Privacy and Trust PSPT04*, 2004.

[12] F. Almenarez, A. Marin, D. Diaz, and J. Sanchez, "Developing a model for trust management in pervasive devices," in *Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshop (PERCOMW06)*, 2006.

[13] "Ubisec project, pervasive trust management model (ptm)," *http://www.it.uc3m.es/ florina/ptm*.

[14] A. Josang, "The consensus operator for combining beliefs," *Artificial Intelligence Journal*, vol. 142, no. 1-2, pp. 157–170, 2002.

[15] C. Forgy, "Rete: A fast algorithm for the many pattern/ many object pattern match problem," *Artificial Intelligence 19*, pp. 17–37, 1982.