Managing Ad-Hoc Trust Relationships in Pervasive Computing Environments

Florina Almenárez, Andrés Marín, Celeste Campo, Carlos García
Dept. Telematic Engineering, Carlos III University of Madrid
Avda. Universidad 30, 28911 Leganés (Madrid), Spain
{florina, amarin, celeste, cgr}@it.uc3m.es,
http://www.it.uc3m.es/pervasive

Abstract

Trust is considered as a fundamental aspect for inter-domain relationships in dynamic open environments. In this paper, we have reviewed existing trust models and their drawbacks when they are applied to pervasive computing. We present PTM, a new automatised decentralised trust management model for pervasive computing environments. PTM overcomes the challenges posed by dynamic open environments, making use of the autonomy and cooperable behaviour of the entities. Our model facilitates ad-hoc trust relationships, captures entities dynamic behaviour along time, and allows trust information exchange through a recommendation protocol. PTM has been validated according to the variable parameters.

I. Introduction

Technological advances are bringing Mark Weiser's 1991 vision [1] closer to reality. Pervasive or ubiquitous computing is "invisible, everywhere computing that does not live on a personal device of any sort, but is in the woodwork everywhere" [2]. So, it involves open spaces comprising multiple devices with certain capabilities. Communication, computing and storage capabilities keep growing while size is getting smaller. Enhanced autonomy and mobility allow these devices to engage in ad-hoc networks within smart environments in order to cooperate among them, called PerNets (pervasive ad-hoc networks). PerNets must be secured in order to avoid information disclosure, modification, unsuitable use of resources, and attacks from harmful entities.

Public Key Infrastructures (PKIs) are commonly used for management of security in fixed networks, especially for authentication of communicating parties. PKIs are facilities for representing and managing trust relationships. Trust depends on certification chains which are extended from "trust anchor" points (Authority Certification, CA). Trust models in PKI are generally hierarchical (usually top-down like in Visa SET [3], or PEM [4]), though practical approaches allow for mixing hierarchies (by issuing cross certificates which establish direct certification paths), leading in occasions to mesh of hierarchies. There is a trade off in keeping the trees manageable, while distributing the load among the authorities, and in any case the trusted root being very sensitive points of the scheme. Furthermore, trust relationships are often configured manually by administrators and require some kind of agreement between authorities; each party must implicitly trust the root CA, or trust anchors to authenticate other entities. Establishing trust models across inter-domains with different root CAs becomes a problem of quite a different degree.

In PerNets, the usability of such models is not so clear. Let us illustrate this with an example. Let N be an ad-hoc network, and let u be an unknown device which intends to join N. For u to join N it would be necessary to perform the following actions: a) to verify that the CA (issuer of u's certificate) is trustworthy; b) to verify the validity of u's certificate; c) to apply control access policies. But, what would happen if u's CA is not preconfigured to be trustworthy in N, or if it is not reachable from N, or if there are no preconfigured control access policies?. Shall N prevent u from joining? Is u to be granted access to services and information offered by other members of N? Is there any mechanism to share this information? A complementary model is therefore needed to handle inter-domains trust relationships in ad-hoc (pervasive) environments.

We propose a new automatised decentralised trust model based on public key which is independent of the security infrastructure. Trust computations are performed and stored by pervasive devices, minimising user intervention as much as possible. Therefore, these computations must be very simple. Trust is dynamic in our approach, allowing the constitution of ad-hoc trust relationships, where partners can be unknown at the beginning of the relationship, since interactions between highly mobile devices are very frequent in pervasive computing. The trust may change along time according to the behaviour's entities. Our approach tries to model trust in the real world, where it is found in a wide range of values. Trust plays an important role in the cooperation and interaction between real world entities. We intend to translate these behaviours to ad-hoc networks and pervasive environments in general, so that entities can exhibit autonomous, dynamic, and cooperable behaviours.

Section II gives a brief explanation of the different approaches about trust models and security models based on trust. In section III, we present our Pervasive Trust Management Model (PTM), its architecture, a recommendation protocol and the equations for trust calculation. Next, in section IV we make an analysis of the belief combination mechanisms. In section V we have validated our model according to the security level, and the variable parameter (β). Finally, we summarise and mention our future research directions in section VI.

II. Related Work

In the last decades, trust models such as [5], [6], [7], [8], and security models based on trust relationships such as [9], [10], [11] have been defined. A trust model for granting or denying access to certain services, using services in unknown environments, exchanging data with strangers is shown in [12]. Finally, an european project, SECURE[13], presents a trust and risk framework.

Marsh [5] presents an overview formalised of trust as a social phenomenon. It embedded trust within artificial agents to provide transactions between them. Marsh's approach is focused on situational trust (i.e. based on situations) which is important in cooperative situations. For that, it uses subjective variables such as situation utility, situation importance, cooperation threshold, perceived risk, among others. It does not define a trust management model nor was designed for open dynamic environments.

Although the Thomas Beth's approach [6] is not defined as a trust model, it proposes a formal method for the evaluation of trustworthiness which can be used to accept or reject an entity as being suitable for sensitive tasks. Each trust value denotes trusting an entity to perform a specific task. It is an extension of [14]. It was not designed to manage the inter-domain trust relationships but task performance. However, it has been the basis of some trust models ([8]).

Blaze et al. propose a decentralised trust model [7] to manage trust, formulating security policies and security credentials, determining whether particular credentials satisfy the policies and deferring trust to third parties. A language for specifying trusted actions and trust relationships is defined. This system neither solves the entire trust management problem nor guarantees the security of the systems using it. It is focused on the authorisation to applications. In addition, the policy definition is complicated and not easy to understand for nonprogrammers [11].

In [8] and [15], trust management is performed by a distributed trust model and a recommendation protocol. Trust categorisation is used and trust is given values for agents and recommenders within each category. Just like PGP, the chains of recommendation are too large. Fourth party recommendation is accepted in this model. In addition, the recommendation protocol is used to measure the agent's trustworthy on a specific category (service). In the model, reputation is used as an indirect mechanism for trust building, however it is not suited for all communities. It does not consider that trust changes along the time. In addition, the notion of context is very important for pervasive computing, but it is not considered here.

Pretty Good Privacy (PGP) [9] was created primarily for encrypting e-mail messages using public or conventional key cryptography. Nowadays, it is used mainly to encrypt local files. Key authenticity is provided through a trust model, but key distribution and key management is not defined in the model. Moreover, a wide trust architecture called 'web of trust' is considered, lacking of fixed or formal certification paths (i.e. each user is responsible for retrieving keys from public rings or they can also be obtained from e-mail message bodies, but this is not defined in the model). The resulting model achieves distribution at the price of uncertain authenticity. Although PGP was designed to work in fixed networks and robust devices, it has desirable characteristics to be used in PerNets, lacking only a recommendation protocol and key management protocol.

Jøsang [10] describes a method for computing authenticity based on certificates, on key binding, and on trust relationships. It uses two elements: first, an opinion model and evidence model to represent the trust. Second, a subjective logic to combine opinions by using logic operators. One of the logic operator is the consensus operator which is used for combining beliefs within the Dempster-Shafer belief theory [16]. However, this operator does not take into account the reliability of the information sources, since it assumes that all sources are equally reliable. It is a powerful model, but it could be too complex to be implemented in limited devices interacting in dynamic environments. The complexity is given by the configuration and calculation of several subjective parameters.

L. Kagal et al. ([11], [17]) give a security approach based on trust for pervasive computing. Trust management is performed with security policies, credentials and trust delegations. A security agent in each domain is responsible of the trust management. X.509 certificates are used as authentication mechanism, and authorisation certificates are generated by the security agents as authorisation mechanism. A fixed device is required (within each domain) to offer on-line connectivity with the involved CAs. This approach is good for closed spaces, but it is not suitable for dynamic open environments. In each domain there is a single failure point (security agent) and requires on-line connectivity to a fixed network infrastructure to be functional.

Finally, SECURE project [13] presents a formal trust and risk framework to secure collaboration between ubiquitous computer systems. The aim is to support collaborative tasks similar to [5], for that, mutual trust is used. The access control is performed using categories and role (extended RBAC). The risk is calculated based on all the possible outcomes for each operation, therefore, human intervention is required.

The above mentioned approaches present drawbacks for pervasive environments and open spaces. We have defined a pervasive trust model between autonomous entities without central servers. It guarantees ad-hoc secure interactions. It performs a distributed management of trust relationships and captures the human's behaviour in order to predict potential attacks. It is simple enough to implement in the very limited devices which have strict resource constraints. This trust model is the basis for authentication and authorisation. The model is formally presented in the next section.

III. PTM: Pervasive Trust Management Model

Granting access to join a PerNet, to access some resource, or allowing interactions among entities in general, are decisions motivated by trust. Trust between entities could be based on a single CA (a single trust domain) or on multiple CAs of

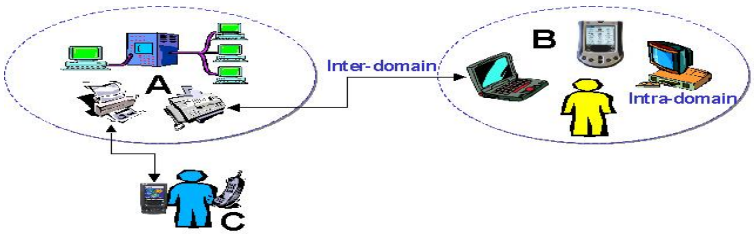

Fig. 1. PTM Scenario

different trust domains. Current PKIs models make difficult to implement such inter-domains relationships. Our model therefore overcomes the challenges posed by entities interacting each other.

A. Definition of Trust

Several trust definitions have been given at psychology, economy, sociology, mathematics, etc. We have based our definition of trust on Jøsang [18] as *the belief that an entity has about other entity, from past experiences, knowledge about the entity's nature and/or recommendations from trusted entities. This belief express an expectation on the entity's behaviour, which implies a risk.*

We want to remark the subjective nature of trust as being an expectation on the behaviour, from the definition above. Trust is founded on particular beliefs, and this fact together with the dynamic PerNets motivates us to define PTM. Distribution allows that new entities easily join the system, it is possible to find multiple paths or trust between entities, it avoids the dependence on a central server, and it has not a single failure point. In addition, our trust definition assumes an abstract trust value because of the simplicity required since for storing trust value by each task would not be scaleable in limited devices. According to Marsh [5], this is a general trust in another entity which represents the amount of trust entity A has in another entity B; it is not relative to any specific situation.

In our model, we assume that all entities are autonomous and some of them are mobile. Entities in our model can be persons, organizations, departments, etc., and its devices as we can see in the Fig. 1. Thus, we establish trust relationships between entities. Each entity manages its own security like PGP. If there are established trust relationships among CAs these would be used; but an entity can also create its own trust relationships. So, each entity handles a protected list of trustworthy and untrustworthy entities, the trust value (degree) associated with them, behaviour's information, the public key and the public key's validity.

B. Properties of Trust

Trust relationships between entities are required to fulfil certain properties. Let us define three entities A, B and C within PerNet N. We define $R(A, B)$ as a function of trust relationship between A and B ($A \rightarrow B$). R is defined as a continuous function ranging from 0 to 1, being these values the extreme cases of false and true respectively. We rely on fuzzy logic (which is based on polyvalent logic from Lukasiewicz) because it enables us more granularity than Boolean logic. So, we include intermediate states between these extremes. We define that 1 is complete trust, 0.5 ignorance (no trust) and 0 is complete distrust. We define a trust path $P(A, B, C)$ as trust relationships between A and C through B. In addition, we use temporal logic through the symbols Gx (always x), Fx (sometimes x) and xUy (is true until y is true); these symbols represent future time. Finally, we define positive actions as a^+ and negative actions as a^-. Then, the properties of the trust relationships supported by PTM are the following:

- **Reflexive**. Every entity trusts itself: $\forall A | G(R(A, A) = 1)$.
- **Non-symmetrical**. If A trusts B, not necessarily B trusts A. $\exists A, B | (\exists R(A, B) \nRightarrow \exists R(B, A))$.
  $\exists R(A, B), R(B, A) | R(A, B) = \alpha \wedge R(B, A) = \beta \rightarrow F(\alpha = \beta)$.
- **Conditionally transitive**. This property is only applied between unknown entities which can become known through a trust path P. If A trusts B and B trusts C, then A conditionally trusts C. $\exists R(A, B), R(B, C) \wedge \nexists R(A, C) U \exists P(A, B, C)$.
  $\exists R(A, B), R(B, C), P(A, B, C) | R(A, B) = \alpha \wedge R(B, C) = \gamma \Rightarrow F(R(A, C) \leq \alpha.\gamma)$ for each $P(A, B, C)$.
- **Dynamic**. Trust changes along the time. $\exists R(A, B) = \alpha | G(a^+ \rightarrow R(A, B) \geq \alpha) \wedge G(a^- \rightarrow R(A, B) < \alpha)$.

C. Trust Management Model Architecture

In a specific context[1], the entities establish trust relationships either in a direct or an indirect way. This allows us to develop our beliefs about another entity. Once these relationships are established, the trust degree changes according to the entity's behaviour by providing feedback about entity's performance during the interaction. We define the behaviour as our evidence space, which modifies the belief space.

[1]The context is related to environment's information instead of categories.

Our architecture clearly shows the components that comprise a security service: software, hardware and network (Fig. 2) and the properties described above. It includes the phases of trust relationships similar to real world as we can see in the next subsections.

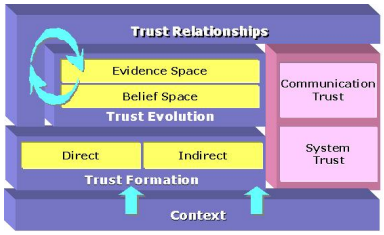

Fig. 2. PTM Architecture

D. How is the trust formed?

Entities joining a PerNet for the first time do not have evidence of past experiences to establish a trust value. In order to establish an initial value, we have two information sources: previous knowledge (direct) or recommendation (indirect).

a) **Direct:** Previous knowledge is given by the entity's nature or past interactions in the physical world, without requesting information to a trusted third party. Then, we assign an initial value as the ignorance value, for instance, which is increased by the user manually or with additional information.

b) **Indirect:** The presence of a trusted entity allows the recommendation. Recommendation is based on an evidence or opinion according to the global view of the recommender entities. When two unknown entities to each other are willing to interact, some trust knowledge is needed. It can be the case that there exists a third entity trusted by both of them. In that case, the trusted entity (B) may be able to recommend another entity (C) to (A) through either a *recommended trust value* or a *certificate issued by the recommender*. Both mechanism are called "recommendations", and in both cases a trust value is needed to calculate the A trust degree in C, so:

- When A is provided with the recommended trust value given by B (R_B), R_B would be the trust value.
- When the recommendation is given by a certificate issued by the recommender B, the certificate could incorporate an extension with the trust degree that B has in C. In such case, A would use this value as R_B. In case there is no such extension we assume a R_B as 1. There is a conflict between the dynamic nature of trust and stamping a degree of trust in a certificate. This does not necessarily implies the need to revoke a certificate (in our model) when the certificate issuer changes its trust degree on the certificate holder. A wise implementation could use trust aging techniques, since the certificate issuing time is known.

In order to calculate the final A trust degree in C, R_B is weighted by our trust degree on the recommender B, so that $R(A,C) = R_B.R(A,B)$. This result can be derived from the third property (Section III-B). However, we will often have more than one recommendation, then we will compute the trust degree as the average of all recommendations (R_{B_i}) weighted by the trust degree of the recommender ($R(A,B_i)$):

$$R(A,C) = \frac{1}{n}\sum_{i=1}^{n} R_{B_i}.R(A,B_i) \tag{1}$$

Eq. (1) will not fulfil the conditionally transitive property (III-B) for all recommendation paths n, but only for some of them. The only possibility to fulfil this property is to choose the minimum among the weighted recommendations. Nevertheless, this function would be extremely conservative, and often it would be the case that we end up trusting on the opinion of the less trusted recommender. Thus, the conditionally transitive property is defined for each path, but if there are several available paths, it does not rule the combination of them.

We make a weighted average of the recommendations (eq. 1) because the recommender's trust degree is important for evaluating the reliability of the sources; unlike the belief combination model proposed by Dempster-Shafer [16] and the consensus operator by Jøsang [19] which assume equally reliable sources. We believe in the recommendations as long as we trust the entity. In dynamic environments, it is very useful since we could find both trusted entities and malicious ones. Besides, we could find known entities and unknown entities. In the section IV, we make a belief (recommendation) combination analysis by comparing the weighted combination with mentioned above approaches.

Now, a question turns up: How do you exchange recommended trust values? Well, we define a recommendation protocol that it can be invoked by any entity to improve its trust knowledge.

E. Pervasive Recommendation Protocol

When an unknown entity wants to interact with us, we request information to close entities, but we only take into account the replies from trusted entities. Recommendation process includes at least three entities:

1) Requester (A), who requests a recommendation about another entity.
2) Target peer (C), unknown entity on which recommendation is requested.
3) Recommender(s) (B), who sends a recommendation.

Unlike another recommendation mechanisms such as the Distributed Trust Model [8], we only accept recommendations from third trusted parties. We do not accept recommendations from fourth entities which are not trusted for us. That is, we do not build a complex graph of recommendation. In addition, we request information about the trust degree of an entity instead of requesting information about a specific service. As mentioned above, our aim is to know the trustworthiness of an entity.

Our recommendation protocol defines three messages to exchange information between entities. We have defined these messages taking into account the optimization of the wireless link use and security, hence it is a simple protocol. In the protocol implementation figures we can show the protocol flow.

1) Recommendation Request (RRQST). It allows the request of trust information about an unknown entity. For instance, C wants to interact with A. A checks the information supplied by C, If $C's$ CA is unknown, A requests recommendations (sending a $RRQST$) to close entities. A would wait some time x to receive recommendations. A only considers as recommendations those from trusted entities. When the time expires, the trust value is calculated using the equation 1. The Fig. 3 shows a request of recommendation and the replies processing. The RRQST message format is the following:
RRQST ::= $\{Request\_ID, Requester\_ID, Target\_Peer\_ID, Timestamp\}$
Where, *Request_ID* is the Request Identifier which is unique, this field allows the requester to know what message is being replied after. *Requester_ID* is the Requester Identifier; it could be formed by a unique name, or a certificate. *Target_Peer_ID* is the Target Peer Identifier. *Timestamp* is the request time in order to avoid replay attacks and to discard any old $RRQST$ that may still be floating around in the system.
From recommendations, we assign trust values according to external information, but an extreme case is when nobody has information about C, then A should decide whether granting access to C or not. A can verify the C's public key using a challenge-response method. The trust value assigned to C would be the ignorance value. However, we could think to assign a trust value depending on internal information ([20]), that is, how much are we endangering? Could we overcome a deceit? Do we have protetion mechanisms against threats?. So, the initial trust value would be assigned from a trust policy instead of a default value. To evaluate internal information is an implicit way to value the risk. Future experiences will help us to maintain or change this decision.

```
request_recommendation() {
    broadcast(RRQST(Request_ID, Requester_ID, Target_ID, Timestamp));
    Total_rec = 0;
    timeout (x, EXPIRED);
    loop {
        R = hear_network(RRPLY(Request_ID, Recommender_ID, Timestamp, Trust_Value));
        if T_recommender ≥ α {
            Trust += Trust_Value*T_recommender;
            Total_rec + +;
        }
    }
    EXPIRED:
    if (Total_rec == 0) Trust = ignorance_value;
    else Trust = 1/Total_rec *Trust;
    return Trust;
}
```

Fig. 3. RRQST Message Implementation

2) Recommendation Reply (RRPLY). This message is used to reply a RRQST message. The request is replied by those entities that know C with $RRPLY$:
RRPLY ::= $\{Request\_ID, Recommender\_ID, Timestamp, Trust\_Value\}$
Where, *Recommender_ID* is the Recommender Identifier. *Timestamp* is the reply time, and *Trust_value* is the degree of trust associated with the target peer.
RRPLY is sent by unicast to the requester by those devices that have information about the target peer. The Fig. 4 shows the sending of a reply.

```
send_reply_recommendation() {
    A = hear_network(RRQST(Request_ID, Requester_ID, Target_ID, Timestamp));
    if known(Target_ID) {
        Trust_Value = search_trust_value(Target_ID);
        unicast(RRPLY(Request_ID, Recommender_ID, Timestamp, Trust_Value));
    }
}
```

Fig. 4. RRPLY Message Implementation

3) Recommendation Alert (RALRT). It warns close entities about a harmful entity having performed a malicious action. The Fig.5 shows the sending of a $RALRT$ message and the processing of this message in case of receiving it.
RALRT ::= {Sender_ID, Target_Peer_ID, Timestamp}
RALRT is sent to all close entities. When we have received a RALRT message, first we must verify the sender's message:
- If the message is sent from an unknown entity, the message would be ignored.
- If Sender's trust $<$ Target_Peer's trust, the message would be ignored.
- If Sender's trust = Target_Peer's trust, Target_Peer's trust would be decreased.
- If Sender's trust $>$ Target_Peer's trust, Target_Peer's trust would be decreased or removed.

```
send_alert() {
    broadcast(RALRT(Sender_ID, Target_ID, Timestamp));
}

receive_alert() {
    A = hear_network(RALRT(Sender_ID, Target_ID, Timestamp));
    if unknown(Sender_ID) ignore(message);
    else
        if T_sender < T_target_peer ignore(message);
        else
            if T_sender ≥ T_target_peer decrease(T_target_peer);
}
```

Fig. 5. RALRT Message Implementation

To provide messages with integrity and confidentiality we can use secure sockets (SSL) or security protocols implemented in lower layers.

F. How does trust evolves?

We have an initial trust value which is often a consequence of a complex set of beliefs, perceptions and interpretations on the evidences. But this value is not static since it changes according to the entity's behaviour (positive and negative experiences) along the time. Trust learning is gradual and dynamic. We can see two spaces separated, similar to the Jøsang's model [18]: the **belief space** and the **evidence space**.

c) **Belief Space:** Our belief space is formed from either the previous knowledge about an entity or the evidences obtained. When we assign an initial trust value using any mechanism, we have an opinion about that entity. This opinion will change in accordance with the entity's behaviour. The opinion is described as a set of propositions (fuzzy logic). These propositions express the ownership degree of an entity to the set of trustworthy entities through a quantitative adverb. For example, *A believes that B is very trustworthy* or *A trusts B* carry the same meaning. We use doxastic logic as a syntax for the propositions, where B_b^a means "a believes b": $B_b^a[complete \mid high \mid medium \mid low][trust \mid distrust]$

The quantitative adverb is equivalent to the trust value associated with the trust range from complete distrust to complete trust as shown in the Fig. 6. The compound propositions allow us to build access control policies.

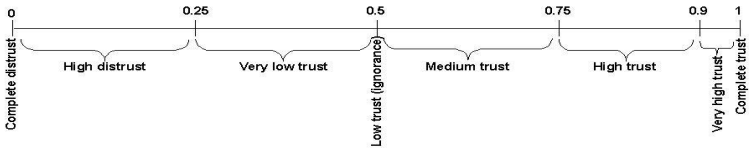

Fig. 6. Trust Clasification

d) **Evidence Space:** Evidence space is formed by past and current experiences. The experiences are facts in the entity's knowledge base representing the information about the trustworthy entities and untrustworthy entities. It is important that each entity stores information about untrustworthy entities because distrust is different from not to have any trust.

Experiences represent behaviour patterns of each entity. These patterns help us to evaluate the new trust values in inverse proportion to the entity's negative actions. Likewise, we measure the entity's behaviour according to its actions[2]. Actions can be positive or negative. However, we assume that all negative actions are not the same that is the reason because we distinguish between wrong actions and malicious actions: *Positive*, i.e. right actions; *Wrong*, i.e. bad actions that do not cause any damage or cause mild damages; and *Malicious*, i.e. harmful actions such as attacks.

Accesses to authorised resources and suitable use of them, they are considered right actions. An entity can make wrong actions by mistake (accidentally) or intentionally, but it is difficult to know. Then we consider as wrong actions those slightly incorrect actions like trying to access to unauthorised resources, or overuse of local resources. Depending on the context, these actions can be interpreted as malicious. Malicious actions are attacks such as sending the same request n at a high rate, misrepresentation of entities, sending viruses, unauthorised access to resources for copying, modifying or destroying data, etc.

We define a function in order to calculate the *action value* V_a. This value is calculated taking into account the performed action weight, but this value is penalised or rewarded by the past behaviour. This function increases or decreases according to the performed positive and negative actions respectively, so:

$$V_a = (1 - \frac{A_N}{Total_a}).{W_a}^{(m)}$$

Where $1 - \frac{A_N}{Total_a}$ means the past behaviour. This value tends to zero (0) when the behaviour is negative, and it tends to 1 when the behaviour is positive. A_N is the number of negative performed actions and $Total_a$ is the total number of performed actions. It comprises the historical behaviour of the entity.

[2]Action modelling is beyond the scope of this paper.

W_a is the action's weight according to its nature (positive, wrong, and malicious). According to the fuzzy logic W_a for positive actions is 1, for wrong actions is 0.5 and for malicious actions is 0. Positive actions have associated weight equal 1 because they are correct actions. Wrong actions have associated 0.5 because they are considered bad actions erroneously. Finally, we associate weight 0 to malicious actions because when we are attacked, we should remove the trust and warn to another entities.

The parameter m is the security level, where $m \geq 1$. This security level affects the action weight, for this reason we raise the action weight to the power of (m). The exponential really influences when the actions are wrong. We will show later in Fig. 7 how the security level affects the action values.

When a new action is performed, V_a is recalculated, reflecting the present behaviour of the entity. The new trust value will take it into account and modify the current trust value, for instance, $R(A,C)_{previous}$, according to the following formula:

$$R(A,C)_{new} = \begin{cases} V_a.\beta + R(A,C)_{previous}.(1-\beta) & V_a > 0 \\ 0 & \text{else} \end{cases}$$

Where β is a configurable parameter to give weight to the present with respect to the past. Therefore, β equal 0 means we will never change our opinion, and β equal 1 means that we do not have any memories and we are only interested in the present. Neither β equal 0 nor β equal 1 are good options, it should exist an equilibrium between the past and the present.

IV. Belief Combination Analysis

For combining recommendations in pervasive scenarios with limited devices, we consider the weighted average is a better option than others because is simpler and distinguishes among the reliability of the sources. We have compared our results with the original Dempster's rule, the Smet's non-normalised version of Dempter's rule with the well known example that Zadeh [21] used for the purpose of criticising Dempster's rule, and the consensus operator (CO). This example is explained by Jøsang in [19]. Suppose that we have a murder case with three suspects (Peter, Paul, Mary) and two witnesses (Witness1, Witness2) who give highly conflicting testimonies. Table I gives the witnesses's belief masses in Zadeh's example and the resulting belief masses after applying Dempster's rule, the non-normalised rule, the consensus operator and the weighted average. For weighted average, we assign 1 as the degree of trust to both witnesses.

	Witness1	Witness2	Dempster's rule	Non-normalized Dempster's rule	Consensus operator	Weighted average
Peter	0.98	0.00	0.490	0.0098	0.492	0.490
Paul	0.01	0.01	0.015	0.0003	0.010	0.010
Mary	0.00	0.98	0.490	0.0098	0.492	0.490
Θ	0.01	0.01	0.005	0.0001	0.006	0.010

TABLE I
COMPARISON OF OPERATORS IN ZADEH'S EXAMPLE

Θ is the uncertainty value which is introduced by allocating some belief. When the uncertainty is introduced, Dempster's rule corresponds well with intuitive human judgement. The non-normalised Dempter's rule however indicates that none of the suspects are guilty and that new suspects must be found. The consensus operator corresponds well with human judgement, too. Our weighted average gives almost the same result as consensus operator and Dempster's rule.

The main difference is the simplicity in the calculation of the trust degrees. Other difference is that in our model, uncertainty is a negative factor representing incomplete knowledge about some entity. Finally, it is shown in [22] and is stated that the weighted average operator (WAO) is not associative, but we argue that it can be computed by an algorithm that ensures its associativiness. The algorithm stores: i as the number of opinions that have been computed, and $R_{i-1}(A,C)$ as the latest result.

$$R_i(A,C) = \frac{(i-1)R_{i-1}(A,C) + R_{B_i}.R(A,B_i)}{i} \tag{2}$$

In [22] it is also shown that when the uncertainty (Θ) is 0, WAO and CO produce equal results.

V. PTM Validation

We have validated our model according to the security level and the variable parameter (β) in order to demonstrate as trust values correspond suitably with intuitive human judgement. As scenario (as shown Fig.1) we have used in a smart laboraty (SL) which has multiple embedded and fixed devices offering services such as printers, photocopiers, fax, computers, web cams, multimedia projectors, etc. These services are used by different users, that is, known users (B) such as researchers, professors, etc., and unknown users (C) such as visiting, scholarship holders, etc. Visiting users can use the services allowing the automatic configuration when they require a specific service. In addition, users form ad-hoc networks to share information.

PTM allows us to establish ad-hoc trust relationships and to automate as much as possible the dynamic certificate management and the access control without an administrator.

When a visiting entity C_i arrives, an entity detecting its presence requests information about it. In our case, nobody has information about C_i, then we grant access to C_i by assigning an initial trust value 0.5. Let β be 0.5. The security level (m) is different for SL and B because SL requires higher security level than B. B has a lower security level because it is in a friendly environment (its job place). Thus, the $B's$ security level is equal to 1; and in $SL's$ security level is equal to 2. After the sixth action, the $C_i's$ trust value in B is 0.237, higher than 0.161 obtained by SL.

The table in the Fig. 7 illustrates how trust changes with the behaviour of the entity and the influence of the security level. The table also shows how the action value is decreased as the number of negative actions increases.

Action	B		SL	
	V_a	$R(C_i)$	V_a	$R(C_i)$
Positive	1	0.750	1	0.750
Wrong	0.25	0.500	0.125	0.437
Wrong	0.166	0.333	0.083	0.260
Positive	0.5	0.416	0.5	0.380
Wrong	0.2	0.308	0.1	0.240
Wrong	0.166	0.237	0.083	0.161

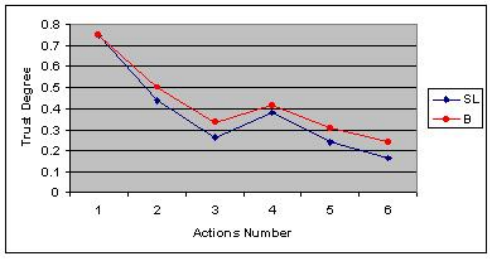

Fig. 7. Comparison of Trust Degrees according to the Security Level

On the other hand, the trust values can change according to the entity type, for instance, a conservative entity (β equal 0) would never change the initial trust value 0.5, an entity without memories (β equal 1) would always assign the last action value V_a, or an entity that maintains the equilibrium (β equal 0.5) would have fair trust values as above values. Fig. 8 proves that neither β equal 0 nor β equal 1 are good options (dotted line), a better option is to maintain the equilibrium. So, each entity calculates its own trust values which use for recommendation. The entity receiving the recommendation calculates a new trust value and will assign permissions in accordance with the new value. Hence, the fact of having variable parameters in our model does not affect the cooperation between entities. Besides, it provides autonomy to each entity to take decisions and to define policies. Finally, the storage of the historical behaviour is light since it records a summary, in this way it does not require high storage capabilities.

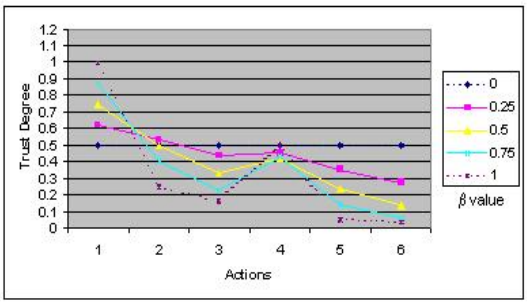

Fig. 8. Comparison of Trust Degrees according to the Parameter β

In general, the negative behaviour converges to 0 whereas the positive behaviour converges to 1. The convergence can be slower or faster according to trust policies as we can see in the Fig. 9.

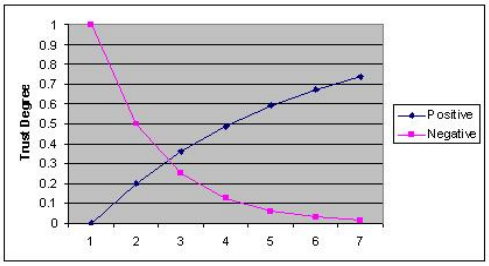

Fig. 9. Entity's Behaviour

With this information, we could also make statistics about the risk. Besides, the error could be calculated as the difference between real behaviour and calculated behaviour.

VI. Conclusions and Future Work

Trust is the basis of the inter-domain relationships in any community. In this paper, we have reviewed the existing trust models and we have shown that they present drawbacks to be applied to pervasive computing. We have presented PTM whose main contribution is the decentralised and automatic management of trust relationships for PerNets. PTM presents a number of advantages: it is decentralised because it does not require to establish hierarchical relationships between CAs and entities can be autonomous; it does not require fixed security infrastructures, but if such an infrastructure exist, PTM could make use of it as being compatible with hierarchical PKIs; in PTM trust changes dynamically, according to the entity's behaviour; the entities cooperate among them and with the environment to share trust information by evaluating the reliability of the sources of information; it minimises the human intervention since most security management functions can be performed automatically; it is used as the basis for authentication and authorisation; and finally, PTM is simple, it can be executed on limited devices.

Now, we are working in the PTM implementation using J2ME Personal Profile for limited devices, in order to test the usefulness of our model in dynamic open environments where are formed PerNets. In addition, we have defined a trust based access control system, called TrustAC (Trust-based Access Control) with its elements such as entities, degrees of trust, resources, actions and permissions. TrustAC have been implemented using and customising XACML [23] Sun's implementation in J2SE for PC and Personal Java for limited devices, in order to integrate it with PTM.

On the other hand, PTM is being used for enhancing the security architecture of Windows CE .NET-based devices. For this, we have defined three modules: *Trust Manager*, *Recommendation Protocol*, and *Action Monitoring*. *Trust Manager* calculates the trust value for new entities, recalculates the new trust value when a known entity performs an action, includes automatically new trusted certificates in the repository, and manages trust policies. *Recommendation Protocol* implements the recommendation protocol. *Action Monitoring* captures the entity behaviour; it is responsible to identify the action type (positive, negative) and calculating action value (V_a).

We are going to simulate our recommendation protocol in order to measure the time and the battery consumption required to establish a trust relationship with and without recommendations and to detect the harmful entity presence; the amount of trust relationships created, taking into account the correct relationships percentage; and the use of wireless links among others.

Finally, PTM has been proposed to provide a secure service discovery protocol (SPDP) [24]. We will study the application of PTM for agent interaction in Multi-Agent Systems (MASs) where security, trust and privacy policies are being addressed in the FIPA Security Specification [25].

Acknowledgments

This work is being developed in the Pervasive Laboratory (PerLab) Group. The authors thank UBISEC and EVERYWARE projects.

References

[1] Weiser, M.: The computer for the 21st century (1991)
[2] Weiser, M.: Ubiquitous computing (1997)
[3] Visa, MasterCard: Secure electronic transaction (SET) (1999)
[4] Kent, S.: Privacy enhancement for internet electronic mail (1993)
[5] Marsh, S.P.: Formalising Trust as a Computational Concept. PhD thesis, University of Stirling (1994)
[6] Beth, T., Borcherding, M., Klein, B.: Valuation of trust in open networks. In: Proceedings of the European Symposium on Research in Computer Security (ESORICS '94, Brighton, UK). Number 875 in Lecture Notes in Computer Science, Heidelberg, Germany, Springer-Verlag (1994) 3–18
[7] Blaze, M., Feigenbaum, J., Lacy, J.: Decentralized trust management. In: Proceedings of the IEEE Symposium on Research in Security and Privacy. Number 96-17, Oakland, CA, IEEE Computer Society, Technical Committee on Security and Privacy, IEEE Computer Society Press (1996)
[8] Abdul-Rahman, A., Hailes, S.: A distributed trust model. In: Proceedings of the ACM Workshop on New Security Paradigms, Cumbria, United Kingdom, ACM SIGSAC, ACM Press (1997) 48–60
[9] Zimmermann, P.R.: The Official PGP User's Guide. MIT Press, Cambridge, MA, USA (1995)
[10] Jøsang, A., Knapskog, S.J.: A metric for trusted systems. In: Proc. 21st NIST-NCSC National Information Systems Security Conference. (1998) 16–29
[11] Kagal, L., Finin, T., Joshi, A.: Trust-based security in pervasive computing environments. In: IEEE Computer. Volume 34. (2001) pp. 154–157
[12] Bussard, L., Roudier, Y., Kilian-Kehr, R., Crosta, S.: Trust and authorization in pervasive B2E scenarios. In: In the Fifth International Conference on Ubiquitous Computing. UbiComp 2003. (2003)
[13] Gray, E., O'Connell, P., Jensen, C., Weber, S., Seigneur, J.M., Yong, C.: Towards a framework for assessing trust-based admission control in collaborative ad hoc applications (2002)
[14] Yahalom, R., Klein, B., Beth, T.: Trust relationships in secure systems-A distributed authentication perspective. In: Proceedings of the 1993 IEEE Computer Society Symposium on Security and Privacy (SSP '93), Washington - Brussels - Tokyo, IEEE (1993) 150–164
[15] Abdul-Rahman, A., Hailes, S.: Supporting trust in virtual communities. In: Proc. 33th Hawaii International Conference on System Sciences, IEEE Press (2000)
[16] Shafer, G.: A mathematical Theory of Evidence. Princeton University Press (1976)
[17] Kagal, L., Undercoffer, J., Perich, F., Joshi, A., Finin, T.: Vigil: Enforcing security in ubiquitous environments. In: Grace Hopper Celebration of Women in Computing 2002. (2002)
[18] Jøsang, A.: An algebra for assessing trust in certification chains. In: Proceedings of the Network and Distributed Systems Security (NDSS'99) Symposium, The Internet Society. (1999)
[19] Jøsang, A.: The consensus operator for combinig beliefs. In: Artificial Intelligence Journal. Number 141/1-2 (2002) 157–170
[20] Luhmman, N.: Trust. MIT Press, Cambridge, MA, USA (1995)
[21] Zadeh, L.A.: Review of shafer's a mathematical theory of evidence. AI Magazine **5** (1984) 81–83
[22] Jøsang, A., Daniel, M., Vannoorenberghe, P.: Strategies for combining conflicting dogmatic beliefs. In: In the proceedings of the 6th International Conference on Information Fusion. (2003)

[23] OASIS: extensible access control markup language (XACML) (2003)
[24] Almenárez, F., Campo, C.: SPDP: a secure service discovery protocol for ad-hoc networks. In: In Workshop on Next Generation Networks - EUNICE 2003. (2003)
[25] FIPA: FIPA MAS security white paper (2002)
[26] English, C., Nixon, P.A., Terzis, S., McGettrick, A., Lowe, H.: Security models for trusting network appliances. In: In Proceedings of the 5th IEEE International Workshop on Networked Appliances. (2002)
[27] Chadwick, D., Young, A., Cicovic, N.: Merging and extending the pgp and pem trust models – the ice-tel trust model. IEEE Network **11** (1997) 1624
[28] Gambetta, D.: Can we trust trust? In Gambetta, D., ed.: Trust: Making and Breaking Cooperative Relations. Basil Blackwell, New York, NY (1988) 213–237