# PTM: A Pervasive Trust Management Model for Dynamic Open Environments*

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

## Abstract

*Trust is considered as a fundamental aspect for inter-domain relationships. We have reviewed existing trust models and their drawbacks when they are applied to pervasive environments. We present PTM, a new 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.*

## 1. Introduction

Technological advances are bringing Weiser's 1991 vision ([20]) closer to reality. Pervasive environments comprising multiple devices with communication, computing and storage capabilities (while size is getting smaller) communicanting in open dynamic spaces. Enhanced autonomy and mobility allow these devices to engage in ad-hoc networks in order to cooperate among them, called PerNets (pervasive ad-hoc networks). PerNets must be secured for avoiding information disclosure, modification, unsuitable use of resources, and/or attacks.

Public Key Infrastructures (PKIs) are commonly used for management of security in networks. Trust models in PKI are generally hierarchical, leading in occasions to mesh of hierarchies to distribute the load among the authorities (CAs). In any case, the trusted root being very sensitive point of the scheme, the trust relationships are often configured manually by administrators and require some kind of agreement between CAs. 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, for instance, if an unknown device $u$ intends to join a PerNet $N$, would be neccesary that the CA issuer of $u's$ certificate is trustworthy and then to verify the validity of $u's$ certificate. But, what would happen if $u$'s CA is not preconfigured to be trustworthy, or if it is not reachable from $N$?. Shall $N$ prevent $u$ from joining? Is there any mechanism to get information about $u$?. So, we propose a new decentralised trust model based on public key which is independent of the security infrastructure, allowing the constitution of ad-hoc trust relationships, since interactions between highly mobile devices are very frequent in pervasive computing. Our approach tries to model trust in the real world, intending to translate these behaviours to PerNets, so that entities can exhibit autonomous, dynamic, and cooperable behaviours.

Section 2 gives a brief explanation of the different approaches about trust models. In section 3, we present our Pervasive Trust Management Model (PTM), its architecture, a recommendation protocol and the equations for trust calculation, making an analysis of the belief combination mechanisms. In section 4 we have validated our model. Finally, we summarise and mention our future research directions in section 5.

## 2. Related Work

In the last decades, trust models such as [17], [4], [5], [2], and security models based on trust such as [23] (Pretty Good Privacy, PGP), [12], and [15] have been defined. A trust model for authorisation to certain services in unknown environments is shown in [6].

* Thanks to UBISEC (IST STREP 506926) and EVERYWARE (MCyT Nº2003-08995-C02-01) projects.

Finally, an European project, SECURE ([9]), presents a trust and risk framework to secure collaboration between ubiquitous computer systems; the aim is to support collaborative tasks.

Marsh's approach [17] is focused on situational trust by using subjective variables to calculate the trust value. Beth's [4] proposes a formal method for the evaluation of trustworthiness which is used to authorisation on sensitive tasks; it is an extension of [21] and has been the basis of [2]. Blaze et al. [5] propose a decentralised trust model to determine whether particular credentials satisfy certain policies; the language used to specify trusted actions and trust relationships is complicated and not easy to understand for non-programmers ([15]). In the Abdul-Rahman's approach ([1]), trust management is performed by a distributed trust model and a recommendation protocol; just like PGP, the chains of recommendation are too large; these are used to measure the agent's trustworthiness on a specific category.

PGP [23] is used mainly to encrypt local files; key authenticity is provided through a distributed trust model based on key public cryptograpic. Although PGP was designed to work in fixed networks and robust devices, it has desirable characteristics to be used in PerNets. Jøsang ([11], [14], [12]) describes a method for computing authenticity based on certificates, on key binding, and on trust relationships. It uses an opinion model and evidence model to represent the trust; it is a powerful model, but it could be too complex to be implemented in constrained devices. Kagal et al. give a security approach based on trust for pervasive computing; a security agent (fixed device) in each domain is responsible for the trust management, authentication, and authorisation.

The above mentioned approaches present drawbacks for open pervasive environments. We have defined a pervasive trust model between autonomous entities without central servers. It minimises user intervention as much as possible. It captures the human's behaviour in order to predict potential attacks. It is simple enough to implement in the very constrained devices which have strict resource constraints. It is the basis for authentication and authorisation.

## 3. PTM: Pervasive Trust Management Model

Trust relationships are established between entities. We assume that all entities are autonomous and some of them are mobile. Entities can be persons, organizations, departments, etc., and its devices as we can see in the Fig. 1. Each entity manages its own security like

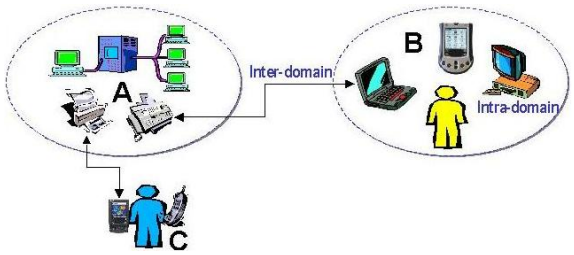


**Figure 1. PTM Scenario**

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, behaviour's information, the public key and the public key's validity. It is important to store information about untrustworthy entities because distrust is different from not to have any trust.

### 3.1. Definition of Trust

Several trust definitions have been given at psychology, economy, sociology, mathematics, etc. We have based our definition on Jøsang 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.*

From the definition above, trust is founded on particular beliefs, and this fact together with the dynamic PerNets motivates us to define PTM. We define a "general trust" (according to Marsh) because of the simplicity required since for storing trust values by each task would not be scaleable in constrained devices.

### 3.2. Properties of Trust

Trust relationships 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 a continuous function ranging from 0 to 1, being these values the extreme cases of complete distrust and complete trust respectively; in addition, we include intermediate states between the extremes, for instance, 0.5 would be used as ignorance value. We rely on fuzzy logic because it enables us more granularity than boolean logic. The fuzzy values are represented by $\alpha, \beta$ and $\gamma$ (i.e. any value between 0 and 1). We define a trust path $P(A, B, C)$ as a trust relationship between $A$ and $C$ through $B$. In addition, we use tem-

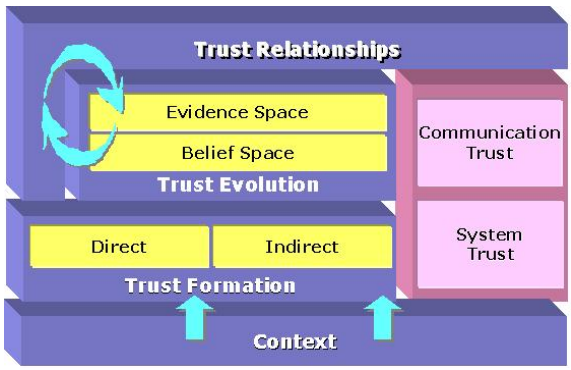


**Figure 2. PTM Architecture**

poral logic with the symbols Gx (always x), Fx (sometimes x) and xUy (is true until y is true), representing future time. Finally, we define positive actions as $a^+$ and negative actions as $a^-$. Then, the properties supported by PTM are:

- **Reflexive**. Every entity trusts itself: $\forall \mathtt{A} | \mathtt{G}(\mathtt{R}(\mathtt{A},\mathtt{A}) = 1)$.
- **Non-symmetrical**. If $A$ trusts $B$, not necessarily $B$ trusts $A$: $\exists \mathtt{A},\mathtt{B} | (\exists \mathtt{R}(\mathtt{A},\mathtt{B}) \nRightarrow \exists \mathtt{R}(\mathtt{B},\mathtt{A}))$

  If $A$ trusts $B$ and $B$ trusts $A$, then, $B$'s trust is not necessarily equal to $A$'s trust: $\exists \mathtt{R}(\mathtt{A},\mathtt{B}), \mathtt{R}(\mathtt{B},\mathtt{A}) | \mathtt{R}(\mathtt{A},\mathtt{B}) = \alpha \wedge \mathtt{R}(\mathtt{B},\mathtt{A}) = \beta \rightarrow \mathtt{F}(\alpha = \beta)$
- **Conditionally transitive**. It is only applied between unknown entities that can become known through a trust path $P$. If $A$ trusts $B$ and $B$ trusts $C$, then $A$ conditionally trusts $C$:

  $\exists \mathtt{R}(\mathtt{A},\mathtt{B}), \mathtt{R}(\mathtt{B},\mathtt{C}) \wedge \nexists \mathtt{R}(\mathtt{A},\mathtt{C}) \mathtt{U} \exists \mathtt{P}(\mathtt{A},\mathtt{B},\mathtt{C})$

  If $A$ trusts $B$ and $B$ trusts $C$, $B$ could recommend to $C$, therefore, $A$ will trust $C$. The $C$'s trust value calculated by $A$ is less or equal than the recommended value multiplied by the $B$'s trust value: $\exists \mathtt{R}(\mathtt{A},\mathtt{B}), \mathtt{R}(\mathtt{B},\mathtt{C}), \mathtt{P}(\mathtt{A},\mathtt{B},\mathtt{C}) | \mathtt{R}(\mathtt{A},\mathtt{B}) = \alpha \wedge \mathtt{R}(\mathtt{B},\mathtt{C}) = \gamma \Rightarrow \mathtt{F}(\mathtt{R}(\mathtt{A},\mathtt{C}) \leq \alpha.\gamma)$ for each $\mathtt{P}(\mathtt{A},\mathtt{B},\mathtt{C})$
- **Dynamic**. Trust changes (increase or decrease) along the time according to the actions:

$\exists \mathtt{R}(\mathtt{A},\mathtt{B}) = \alpha | \mathtt{G}(\mathtt{a}^+ \rightarrow \mathtt{R}(\mathtt{A},\mathtt{B}) \geq \alpha) \wedge \mathtt{G}(\mathtt{a}^- \rightarrow \mathtt{R}(\mathtt{A},\mathtt{B}) < \alpha)$

### 3.3. Trust Management Model Architecture

In accordance with our architecture (Fig. 2), in a specific context, the trust can be established in a *direct* or an *indirect* way (*Trust Formation*). Once the trust is formed, we obtain an initial trust value; this value is our *Belief Space*. However, the trust changes according to the entity's behaviour by providing feedback about entity's performance during the interaction, *Evidence Space*. This process is called *Trust Evolution*, which is constantly changing. Both trust formation and trust evolution allow to create trust relationships between entities; these relationships are supported by the trustworthiness of the system and the communication. It is very important the trustworthiness of the entities as well as the trustworthiness of the communication means. Our architecture includes the phases of trust relationships similar to real world as is shown in the next sections.

### 3.4. 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).

**3.4.1. Direct** The direct trust formation is given by the knowledge of the entity's nature or past interactions in the physical world, without requesting information to a trusted third party (TTP). Nevertheless, although the entities are unknown at the beginning, a trust relationship can be established (depending on the security policies). In both cases, we assign an initial value by default, which is increased by the user or with additional information when the entities are known.

**3.4.2. Indirect** When two unknown entities to each other are willing to interact, can exist a TTP by both of them. In that case, the TTP ($B$) may be able to recommend another entity ($C$) to ($A$) through either a *recommended trust value* or a *certificate*. Both mechanism are called "recommendations" and require a trust value to calculate the $C$'s trust degree, 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 $B$, we assume $R_B$ as 1, because the traditional PKI is boolean. A certificate is only issued when the entity is trustworhty, otherwise the certificate is denied.

### 3.5. Trust Calculation

$A$ calculates the $C's$ trust degree, weighing $R_B$ by our trust degree on $B$, so that $R(A,C) = R_B.R(A,B)$. This result can be derived from the third property (Section 3.2). 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 ($R(A,B_i)$):

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

Eq. (1) will not fulfil the conditionally transitive property 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.

We use the weighted average operator (WAO) because: the recommender's trust degree is important for evaluating the reliability of the sources; unlike the belief combination model of Dempster-Shafer ([19]) and the consensus operator (CO) by Jøsang [10] 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. In addition, WAO is simpler than others to calculate the trust value as being suitable for pervasive devices.

Dempster-Shafer belief model is a framework for upper and lower probability bounds. It defines a set of possible situations (states of a system) which is called the *frame of discernment*. Each state has assigned a belief mass and are called atomic because they do not contain substates, in order to calculate probabilities. The consensus operator is based on statistical inference of subjective uncertain beliefs. This work is built on Dempster-Shafer theory.

We have compared our results with the Dempster's Rule (DR), the Smet's non-normalised version of DR (NDR), and the CO, using the well known example that [22] used for criticising DR. This example is explained in [10]: Suppose that we have a murder case with three suspects (Peter, Paul, Mary) and two witnesses (W1, W2) who give highly conflicting testimonies. Table 1 gives the witnesses's belief masses and the resulting belief masses after applying DR, NDR, CO and WAO. $\Theta$ is the uncertainty value which is introduced by allocating some belief. For WAO, we assign 1 as the trust degree to both witnesses.

Table 1 shows that DR, CO, and WAO corresponds well with intuitive human judgement. NDR however indicates that new suspects must be found. In our model, uncertainty is a negative factor representing incomplete knowledge about the entity. In [13] is shown that when $\Theta$ is 0, WAO and CO produce equal results and is stated that 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.

| | W1 | W2 | DR | NDR | CO | WAO |
|---|---|---|---|---|---|---|
| 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 |
| $\Theta$ | 0.01 | 0.01 | 0.005 | 0.0001 | 0.006 | 0.010 |

**Table 1. Comparison of operators in Zadeh's example**

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

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.

### 3.6. Pervasive Recommendation Protocol (PRP)

Recommendation process includes at least three entities: 1) Requester ($A$), who requests a recommendation. 2) Target peer ($C$), unknown entity on which recommendation is requested. 3) Recommender(s) ($B$), who sends a recommendation.

Unlike other recommendation mechanisms, we only accept recommendations from TTPs avoiding large recommendation chains. In addition, this protocol allows us to exchange trust values instead of requesting information about a specific service. To exchange trust values, PRP defines three messages whose implementation (Fig. 3, 4, 5) shows the protocol flow.

**3.6.1. Recommendation Request (RRQST)** It allows the request of trust information about an unknown entity. For instance, $C$ wants to interact with $A$, but $C$ is unknown to $A$, then $A$ 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 eq. 1 (Fig. 3).

The RRQST message format is the following:

**RRQST** ::= {Rqst_ID, Rquster_ID, Target_ID, TS}

Where, *Rqst_ID* is the unique Request Identifier, this field allows the requester to know what message is being replied after. *Rquster_ID* is the Requester Identi-

fier; it could be formed by a unique name, or a certificate. *Target_ID* is the Target Peer Identifier. *TS* 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. For that, $A$ takes into account internal information ([16]), 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(Rqust_ID, Rquster_ID, Target_ID, TS));
  Total_rec = 0;
  timeout (x, EXPIRED);
  loop {
    R = listen(RRPLY(Rqust_ID, Rcder_ID, TS, TValue));
    if trusted(Rcder_ID) {
       Trust += TValue*T_recommender;
       Total_rec + +;
    }
  }
  EXPIRED:
  if (Total_rec == 0) Trust = ignorance_value;
  else Trust = 1/Total_rec *Trust;
  return Trust;
}
```

**Figure 3. RRQST Message Implementation**

**3.6.2. Recommendation Reply (RRPLY)** It is used to reply a RRQST message. The response is unicasted to the requester by those entities that know $C$:

**RRPLY** ::= {Rqust_ID, Rcder_ID, TS, Trust_Value}

Where, *Rcder_ID* is the Recommender Identifier. *TS* is the timestamp of the reply, and *Trust_value* is the degree of trust associated with the target peer.

**3.6.3. 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_ID, TS}

```
send_reply_recommendation() {
  A = listen(RRQST(Rqust_ID, Rquster_ID, Trgt_ID, TS));
  if known(Target_ID) {
     Trust_Value = search_trust_value(Target_ID);
     unicast(RRPLY(Rqust_ID, Rcder_ID, TS, Trust_Value));
  }
}
```

**Figure 4. RRPLY Message Implementation**

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

receive_alert() {
  A = listen(RALRT(Sender_ID, Target_ID, TS));
  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);
}
```

**Figure 5. RALRT Message Implementation**

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

## 3.7. How does trust evolve?

The initial trust value is not static since it changes according to the entity's behaviour (positive and negative experiences) along the time. We can see two spaces separated, similar to the Jøsang's model: the **belief space** and the **evidence space**.

**3.7.1. Belief Space** Our belief space is formed from either the previous knowledge or the evidences obtained. The belief 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 as shown in the Fig. 6. For example, *A believes that B is very trustworthy*. We use doxastic logic as a syntax for the propositions, where $B_b^a$ means "a believes b": $B_b^a[complete \mid high \mid medium \mid low][trust \mid distrust]$.

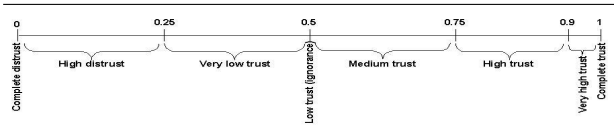


**Figure 6. Trust Clasification**

**3.7.2. Evidence Space** It is formed by past and current experiences. 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. 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 are considered right actions. An entity can make wrong actions by mistake 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, sending viruses, etc.

To calculate the *action value* $V_a$, we take 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:

$$V_a = (1 - \frac{A_N}{Total_a}).{W_a}^{(m)} \quad (3)$$

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 actions and $Total_a$ is the total number of performed actions. $W_a$ is the action's weight according to its nature (positive, wrong, and malicious). $W_a$ for positive actions is 1, for wrong actions is 0.5 and for malicious actions is 0. Finally, 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, $R(A,C)_{act}$, according to the equation 4:

$$R(A,C)_{new} = \begin{cases} V_a.\beta + R(A,C)_{act}.(1-\beta) & V_a > 0 \\ 0 & \text{else} \end{cases} \quad (4)$$

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

## 4. PTM Validation

We have validated our model according to the security level $(m)$ and the variable parameter $(\beta)$ in order to demonstrate as trust values correspond suitably with intuitive human judgement. As scenario (as shown Fig.1) we have used 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. The visitors can use the services offered in SL through the automatic configuration. When a visiting requires a specific service, then, the device (offering the service) sets up automatically the trust relationship with the visiting; without an administrator or connectivity with a remote CA.

When a visiting $C_i$ 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 $\beta$ be 0.5. $m$ 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 entity's behaviour and the influence of $m$. The table also shows how the action value is decreased as the number of negative actions increases.

On the other hand, the trust values can change according to the entity type, for instance, a conservative entity ($\beta$ equal 0) would never change the initial trust value 0.5, an entity without memories ($\beta$ equal 1) would always assign the last action value $V_a$, or an entity that maintains the equilibrium ($\beta$ equal 0.5) would have fair trust values as above values. Fig. 8 proves that neither $\beta$ equal 0 nor $\beta$ equal 1 are good options (dotted line), a better option is to maintain the equilibrium. 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

| 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 |

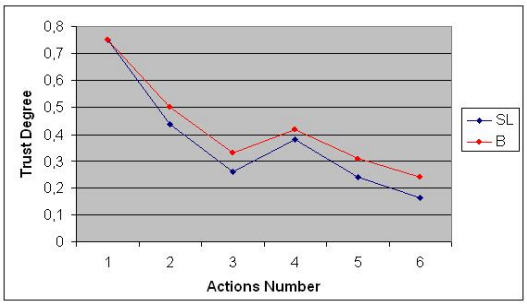


**Figure 7. Comparison of Trust Values according to the Security Level**

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.

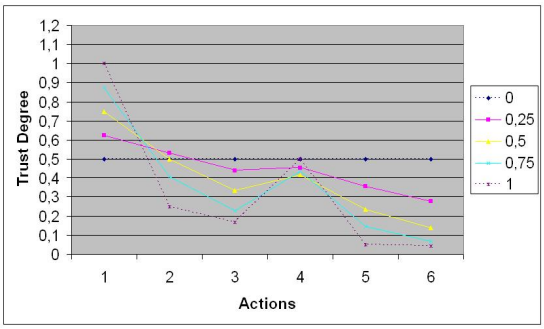


**Figure 8. Comparison of Trust Values according to $\beta$**

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. 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.

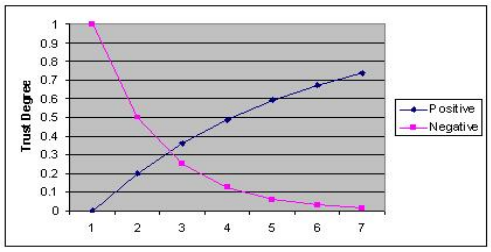


**Figure 9. Entity's Behaviour**

## 5. 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, but it is compatible with hierarchical PKIs; in PTM trust changes dynamically, according to the entity's behaviour; 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, it is simple to be executed on constrained devices.

Now, we are working in the PTM implementation using J2ME Personal Profile for constrained devices, in order to test the usefulness of our model in PerNets and to get experimental results. We have defined and implemented a trust based access control system, called TrustAC (Trust-based Access Control), using and customising XACML [18] Sun's implementation in J2SE for PC and Personal Java for constrained devices to be integrated with PTM.

We are going to simulate PRP in order to measure the time and the battery consumption required to establish a trust relationship with and without recommendations, 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) [3].

## References


[1] A. Abdul-Rahman and S. Hailes. Supporting trust in virtual communities. In *Proceedings 33th Hawaii International Conference on System Sciences*. IEEE Press, January 2000.

[2] A. Abdul-Rahman and S. Hailes. A distributed trust model. In *Proceedings of the ACM Workshop on New Se-*

*curity Paradigms*, pages 48–60, Cumbria, United Kingdom, SEP 97.

[3] F. Almenárez and C. Campo. SPDP: a secure service discovery protocol for ad-hoc networks. In *Workshop on Next Generation Networks - EUNICE 2003*, September 2003.

[4] T. Beth, M. Borcherding, and B. Klein. Valuation of trust in open networks. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS '94)*, number 875 in Lecture Notes in Computer Science, pages 3–18, Heidelberg, Germany, NOV 94.

[5] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, number 96-17, MAY 96.

[6] L. Bussard, Y. Roudier, R. Kilian-Kehr, and S. Crosta. Trust and authorization in pervasive B2E scenarios. In *Fifth International Conference on Ubiquitous Computing 2003*, October 2003.

[7] C. English, P. A. Nixon, S. Terzis, A. McGettrick, and H. Lowe. Security models for trusting network appliances. In *In Proceedings of the 5th IEEE International Workshop on Networked Appliances*, 2002.

[8] D. Gambetta. Can we trust trust? In D. Gambetta, editor, *Trust: Making and Breaking Cooperative Relations*, chapter 13, pages 213–237. Basil Blackwell, New York, NY, 1988.

[9] E. Gray, P. O'Connell, C. Jensen, S. Weber, J.-M. Seigneur, and C. Yong. Towards a framework for assessing trust-based admission control in collaborative ad hoc applications, 2002.

[10] A. Josang. The consensus operator for combinig beliefs. In *Artificial Intelligence Journal*, number 141/1-2, pages 157–170, 2002.

[11] A. Josang. The right type of trust for distributed systems. In *New Security Paradigms ´96 Workshop*, 96.

[12] A. Josang. An algebra for assessing trust in certification chains. In *Proceedings of the Network and Distributed Systems Security (NDSS'99)*, 99.

[13] A. Josang, M. Daniel, and P. Vannoorenberghe. Strategies for combining conflicting dogmatic beliefs. In *Proceedings of the 6th International Conference on Information Fusion*, July 2003.

[14] A. Josang and S. J. Knapskog. A metric for trusted systems. In *Proceedings 21st NIST-NCSC National Information Systems Security Conference*, pages 16–29, 98.

[15] L. Kagal, T. Finin, and A. Joshi. Trust-based security in pervasive computing environments. In *IEEE Computer*, volume 34, pages pp. 154–157, December 2001.

[16] N. Luhmman. *Trust.* MIT Press, Cambridge, MA, USA, 95.

[17] S. P. Marsh. *Formalising Trust as a Computational Concept.* PhD thesis, University of Stirling, APR 94.

[18] OASIS. extensible access control markup language (XACML), 2003.

[19] G. Shafer. *A mathematical Theory of Evidence.* Princeton University Press, 76.

[20] M. Weiser. The computer for the 21st century. *Scientific American*, pages 94–104, September 91.

[21] R. Yahalom, B. Klein, and T. Beth. Trust relationships in secure systems-A distributed authentication perspective. In *Proceedings of the 1993 IEEE Computer Society Symposium on Security and Privacy (SSP '93)*, pages 150–164, Washington - Brussels - Tokyo, MAY 93.

[22] L. A. Zadeh. Review of shafer's a mathematical theory of evidence. *AI Magazine*, 5:81–83, 84.

[23] P. R. Zimmermann. *The Official PGP User's Guide.* MIT Press, Cambridge, MA, USA, 95.