End-to-end Trust Starts with Recognition

Jean-Marc Seigneur, Stephen Farrell,
Christian Damsgaard Jensen, Elizabeth Gray, and Yong Chen

Distributed Systems Group, Department of Computer Science,
Trinity College Dublin, Dublin 2, Ireland
{secure-tcd}@cs.tcd.ie
http://www.dsg.cs.tcd.ie/

Abstract. Pervasive computing requires some level of trust to be established between entities. In this paper we argue for an entity recognition based approach to building this trust which differs from starting from more traditional authentication methods. We also argue for the concept of a "pluggable" recognition module which allows different recognition schemes to be used in different circumstances. Finally, we propose that the trust in the underlying infrastructure has to be taken into account when considering end-to-end trust.

1 Introduction

Weiser's vision of ubiquitous computing [33] will only become true when computing capabilities are woven into the fabric of every day life, indistinguishable from it. In ambient intelligence (AmI) environments [21], where ubiquitous computing[1], ubiquitous communication and intelligent user interfaces are combined, it has been envisaged [6] that real people would have a digital-self acting on their behalf. These digital entities are more likely to be artificial intelligence agents following the real person they are representing — kind of ubiquitous roaming entities. As in real life, these digital entities will encounter other previously unknown entities while roaming from place to place. Billions of entities — potentially any device with a digital heartbeat — are expected to spread in the surrounding environment. A fundamental question concerns the representation of entities including their naming and subsequent identification as well as their association with real-world principals. We believe that, in this context, it is more beneficial to take an approach based on entity recognition [26], rather than solely on traditional authentication schemes like PKI [9] or Kerberos [17].

Establishing the authenticated identity of the other party is not necessarily enough in pervasive computing, because identity conveys no a priori information about the likely behaviour of the other party. In many cases, it will be more useful to determine whether the other party is someone with whom one has interacted successfully in the past, whether the other party can provide a recommendation

[1] The pervasive computing magazine [8] treats ubiquitous computing and pervasive computing as synonyms; so do we.

from a trusted third party (e.g., from a personal friend) or whether the other party has a generally good reputation.

In AmI environments, the question "When should a person cooperate, and when should a person be selfish, in an ongoing interaction with another person?" [2] will be extended to digital entities collaborating with other digital entities. To tackle this question, the concept of trust in computer systems is attracting increasing attention from the research community. Trust has been formalized as a computational model [18, 32], but the term trust means different things in different research communities, for example it may relate to trust in the underlying technology [3] or to trust between entities when they have to collaborate [11, 13]. We argue that end-to-end trust includes both types of trust — trust between parties and trust in the underlying infrastructure.

Usually, authentication is the first step to ensure security in distributed computing environments. In this paper, we concentrate on the authentication level, which is one part of the technical trust in our end-to-end trust model. We start by motivating the need for dynamic enrollment and defining the notion of entity recognition, and then map this to authentication in traditional systems. We then further develop the concept of end-to-end trust as required in pervasive computing environments and describe an end-to-end trust model. Finally, this model is applied for entity recognition.

2 The Need for Dynamic Enrollment in Pervasive Computing

In this section, we motivate and describe our entity recognition process.

2.1 Benefit of Pervasive Computing Comes from Unforeseen Interactions

The current IPv4 based Internet is likely to provide the foundation of any pervasive computing environment. However, mobile ad hoc networks (MANETs) are increasingly common [35] and bring interesting properties such as spontaneity. Mobile entities will have the chance to move and join networks in an ad hoc manner. In fact, as in real life, opportunities will be offered to entities roaming from place to place. These opportunities cannot be predicted because it is not known in advance where these entities will roam. Many other aspects are still unknown: even the scale of the networks of entities cannot really be evaluated; it is known that ad hoc networks will be possible but due to their nature we can not predict when they will be formed; nor can we predict which entities will interact together; etc. We can say that no individual organization will manage the whole; self-organization will have to emerge from these unmanaged networks along with new, previously unknown, features. The benefit of self-organized systems is that they can create higher level features [12]. Thus, pervasive computing will hopefully exhibit interesting properties to solve complex issues, where the diversity of entities opens the door for new opportunities. In the Resurrecting

Duckling security policy model [29], ducklings have to take the risk to emerge from their shell in order to find their mother, who will look after them. In pervasive computing, the scenario is extended to other potentially caring entities such as friends but computational entities will not be able to identify friends without taking the risk to make friends of unknown entities. In MANETs, a node which is too far from an Internet gateway but has another node in wireless range, which is itself in range of the gateway, can only reach the gateway by forwarding packets to this node. Of course, the owner of this helping node can be unknown. In computing terms, the first risk will be to enroll unknown entities. A fundamental requirement for ubiquitous computing environments is therefore to allow for potential interaction with unknown entities.

2.2 Dynamic Enrollment

Generally, authentication schemes start with enrollment of entities. This task is often time consuming and requires explicit human intervention, e.g. from a system administrator. For example, integrating new users may involve considerable work and resources: a random initial secret may be sealed in an envelope and sent to the new user; it can be even worse for smart tokens, which can involve two separate activities — token programming and user management [27].

Usually, entities which have not been enrolled cannot interact. Collaboration is seen as a privilege that only specific entities can obtain, with the implicit assumption that it is known a priori which entities can obtain this privilege. The principle of least privilege is applied for interaction: some entities can interact; others cannot. In pervasive computing environments, collaboration should be possible between all entities; the most likely situation is that one of them has not previously enrolled. There is an inherent conflict between how enrollment is done in current computing systems and what is required for pervasive computing. This introduces the requirement for smooth dynamic enrollment, i.e. the door should not be closed to strangers, but instead any stranger showing up at the door might become an acquaintance. Since it is not known in advance which entities should get the privilege to collaborate, we argue that the principle of least privilege should not be applied for collaboration. We further argue that it is not possible to give pervasive entities exactly the privileges they really need. Instead, a practical approach, as in the real world, would be as follows: a small initial measure of trust is given to any entity so that it can be enrolled and begin collaborating, even though technically this gives permission to do things it should not be doing.

To allow for dynamic enrollment of strangers and unknown entities, we propose an entity recognition process. Table 1 compares the current authentication process (AP) with our entity recognition (ER) process.

There is no initial enrollment step at the beginning of the entity recognition process but this does not mean that enrollment cannot be done. Actually, in step E.3, if the entity to be recognized has never been met before, what will be retained is going to be reused the next time this entity is going to be recognized. Depending on the recognition scheme, it should be more or less transparent, i.e.

Table 1. Authentication and entity recognition side-by-side

Authentication Process (AP)	**Entity Recognition (ER)**
A.1. Enrollment: generally involves an administrator or human intervention	
A.2. Triggering: e.g. someone clicks on a Web link to a resource that requires authentication to be downloaded	E.1. Triggering (passive and active sense): mainly triggering (as in A.2), with the idea that the recognizing entity can trigger itself
A.3. Detective work: the main task is to verify that the principal's claimed identity is the peer's	E.2. Detective work: to recognize the entity to be recognized using the negotiated and available recognition scheme(s)
	E.3. Retention (optional): "preservation of the after effects of experience and learning that makes recall or recognition possible" [30]
A.4. Action: the identification is subsequently used in some ways. Actually, the claim of the identity may be done in steps 2 or 3 depending on the authentication solution (loop to A.2)	E.4. Action (optional): the outcome of the recognition is subsequently used in some way (loop to E.1)

more or less like the enrollment step in A.1. Thus, by moving down the enrollment step in the process, we emphasize that the door is still open for interacting with strangers and unknown entities. We can show that any authentication process can be integrated into an ER scheme (by doing enrollment at step E.3) and can also show that some ER schemes are not authentication schemes and thus that the class of authentication schemes is a proper subset of the class of entity recognition schemes. Further in this paper, we detail a "pure" recognition scheme — "A Peer Entity Recognition" scheme (APER).

A number of different sensing, recognition and retention strategies can be envisaged for entity recognition schemes. However, specifying retention strategies is the subject of ongoing work and beyond the scope of this paper. The detective work depends on which recognition scheme is used, for example, in the APER recognition scheme described in section 4.2, it may consist of sending a challenge/response.

By self-triggering (step E.1) we mean that the entity takes the initiative to start the recognition process in order to recognize potential surrounding entities, for example it may be starting the recognition scheme that involves the recogniser monitoring the network and selectively carrying out detective work on (some of) the identities that are observed. Step E.4 is optional since it is not required if the only objective is to gather recognition information. Step E.3 is also optional

but the reason is different: recognition information need not be retained — say if the entity has been seen before.

To cope with scalability, we propose to forget about entities, that we have not collaborated with, after a certain time. Actually, the tremendous number of entities expected in a pervasive computing environment raises the question of how to scale entity recognition (or authentication) to billions of entities with potentially different distinguishing characteristics. In the next subsection, we describe how entities can be recognized by different means and represent different principals.

2.3 Virtual Anonymity

Our expectation is that entities are in general virtually anonymous to the extent that identity conveys little information about likely behaviour. What is important as a prerequisite is not really "Who exactly does this entity represent?" but "Do I recognize this entity as a trustworthy collaborator?" As there is no a priori information concerning likely behaviour; identity therefore does not imply privilege. We assume virtual anonymity and therefore we do not require (but do allow) the ability to establish the identity of a given entity in absolute terms, e.g. through globally unique and meaningful X.500 "distinguished names" [10]. The nature of MANETs makes it inherently difficult to rely on centralized or online servers. As an example, consider authentication based on Kerberos, which is based on the idea of having a global hierarchy of trust, where leaf and interior nodes trust their superior. This model does not work when the superior is not reachable, e.g. in MANETs where network partitions may be common. For a web of trust like PGP [36], there is still the question of whether trust [1, 5, 13] and recommendations [4, 14] are transitive. In fact, the entity itself is responsible for its final trust decision. In the end, the control should be in the owner's hands. Rather than relying on recommendation or reputation, entities can rely on their own previous interactions and past history with another entity as soon as the recognition is possible. This is why we simply require the ability to recognize other entities, e.g. through their name, location, digital signatures or other means. Collaboration amongst virtually anonymous entities is an approach to security in the global computing infrastructure. The Resurrecting Duckling security policy model [29] is an example of entity recognition; "ducklings" know their mother is the entity who sent the imprinting key when they were "born", i.e. they must be able to recognize the entity which sent the imprinting key, no more. Stajano speaks of anonymous authentication [28], which is also seen in other authentication work [4, 7].

To us, entities are virtually anonymous: any identifier can work as long as it allows for referencing the entity involved over the required lifespan. This means that the "real" identity in absolute terms is not needed. Through collaboration, trust will be associated with these identifiers. How does this trust information relate to the inherent trust of the identifier? Khare claims that there is trust in a name [16]. The next section explains what kind of inherent trust we expect in identifiers and what we mean by end-to-end trust.

3 End-to-end Trust Implications and Model

The first part of this section makes it clear that there is a layering of trust. Then, we explain how trust in the underlying technical infrastructure is linked with trust between entities to form end-to-end trust.

3.1 Acknowledging the Presence of Layers of Trust

Differences in the strength of authentication and recognition schemes obviously raise the question of trust in the underlying infrastructure. However, recognition is not the only technical piece of the infrastructure on which a trust assessment would be useful. In fact, trust at the entity level regarding interactions is meaningless if the low level information about the collaboration is invalid. There are therefore multiple layers of trust described below, which affect end-to-end trust. The overall trust level should be chosen at the application level. We first describe trust in technologies and follow this by describing a "higher" layer of trust, which is trust between entities.

Trust in the Technical Infrastructure The trusted computing platform alliance (TCPA) [31] makes it clear that we can speak of trust in technical components. TCPA focuses on platform identity and integrity metrics to establish trust, which is defined as follows: "an entity can be trusted if it always behaves in the expected manner for the intended purpose" [30]. It is envisaged that the levels of integrity will be set either from experience or recommendation from experts and based upon either direct or indirect assessment of a platform.

We should also note that users ought not be bothered by a requirement that they frequently change security settings, since this would run counter to the requirement that ubiquitous computing should not monopolize the attention of the user [34].

Jøsang [15] gives another approach for assessing trust in underlying technologies based on a belief model, where trust is considered to be a subjective belief, a set of operators for combining beliefs and a combination of evidence such as for example security evaluation, security incidents, security advisory reports, system reputation, ISO 9000 certification or developer reputation. Again, the user should deduce from this evidence the trustworthiness of the system, which seems hard for security and technologically unaware users. Our specific focus, authentication, has even more work trying to define metrics to assess its trustworthiness [14, 22].

Trust between Entities TCPA helps to deduce the trustworthiness in the underlying technology, partly thanks to collaboration between different on-line parties, e.g. a so-called "privacy CA". In return, as any collaboration implies, these different parties have to put more or less trust in their partners to allow for collaboration. Even if this trust assessment takes place at a higher level of abstraction — trust between entities — this highlights the fact that effectively

a chain of trust is present in the background. Axelrod's question on real world cooperation [2] is transferred in pervasive computing as follows: when should an entity cooperate, and when should an entity be selfish, in an ongoing interaction with another entity? This time Marsh's model of trust [18] is a more relevant approach for this layer of trust. The SECURE project [24] aims at building secure environments for collaboration among ubiquitous roaming entities thanks to mechanisms based on the human notion of trust. A model for trust management is expected from this project. One may also mention Jøsang's work [13] or the use of a virtual trust currency, such as the trusto [25].

As mentioned above, dynamic enrollment as required in pervasive computing environments leaves the door open for any encountered entities to become an acquaintance. The question is now how to evaluate trust in such open and dynamic environments?

3.2 End-to-end Trust Model

We have identified layers of trust which can be divided into two main categories: trust in the underlying technology and trust between entities. The point is that these layers form an end-to-end trust, a chain of layers of trust. The overall trust is the result of how much trust is found at each level. Whether the final level of trust is acceptable or not is a separate issue. Some benefits of pervasive computing applications make it worth relying on not-so-trustworthy underlying technologies; there is a trade-off between what can be obtained and what can be lost. This trade-off has to be acknowledged and made clear. Our view is to set up a threshold at the end of the trust chain, most likely the application layer, which would indicate how much trust is required. This threshold may be represented by a value in the range [0,1], 1 for full trust and 0 for no need of trust. A similar approach has been proposed in different authentication metrics [22]. In some way, it is equivalent to setting the trust level in an ASP.NET Web applications [20], even though in that case the trust "granularity" is coarser — consisting of only four levels: full, high, low, none. More complex metrics, such as Jøsang's subjective ones [14, 15], may be implemented at a later stage. To reach this threshold, trust at each layer has to be taken into account. This requires being able to evaluate the end-to-end trust, which is the result of trust in technology and in other entities.

For trust in the underlying technology, we could use metrics, dynamically calculated or statically defined by a group of experts, as detailed above in this paper.

For trust in entities, trust would be calculated based on the human notion of trust, probably thanks to direct observations, past history, and careful use of recommendation and reputation.

Again, as a starting point, we think of converting these two trust values on a scale between 0 and 1, where trust may be interpreted as the probability that an entity behaves in the expected manner for the intended purpose. Since both of these types of trust have no inheritance, we assume they are independent. This suggests the use of the following formula.

End-to-end trust = f(Trust in infrastructure, Trust in entities)

which can be as simple as:

*End-to-end trust = (Trust in infrastructure) * (Trust in entities)*

This formula acknowledges the idea that in pervasive computing the security properties of the underlying infrastructure are more or less strong, e.g. MANETs may come easily but they can also go easily. Indeed, to get the full potential of pervasive computing, the risks of using not-really-trustworthy environments have to be considered explicitly. Nevertheless, the above model helps to keep in mind that a risk has been taken. The next section shows how this can be applied for authentication in pervasive computing.

4 End-to-end Trust Model Applied at the Source — the Authentication Level

One of the foundations of security is authentication. Stajano [28] emphasized that without being sure with whom an entity interacts, the three fundamentals properties - confidentiality, integrity and availability - can be trivially violated. As explained at the beginning of this paper, in some cases, absolute security will not matter and weak recognition schemes may be used in order to still be able to collaborate, this is a choice to be made at the application level. This section explains how this will be applied for recognition by following the model described at the end of the previous section. Recognition is only one piece of the underlying technical infrastructure. However, other mechanisms have to be trusted after recognition, e.g. secure communication over networks after authentication. We consider the problem of implementing our approach for other parts of the underlying infrastructure as beyond the scope of this paper and we only explain how trust in the recognition scheme is reported into the end-to-end trust. We therefore assume secure communication channels for any collaboration that follows recognition.

First, we give an overview of the difference in trust for entity recognition schemes in pervasive computing. Then, we present a "pure" entity recognition scheme (APER). Finally, we give a coarse-grained description of our design and expected implementation.

4.1 Recognition Schemes with Different Technical Trust

Since any authentication scheme can follow the entity recognition process explained above, we already support a considerable set of "legacy" entity recognition schemes: symmetric and asymmetric keys, biometrics... Moreover, the openness required for enrollment suggests many more schemes to come, e.g. the APER scheme described in the next subsection.

On the other hand, each recognition solution will have to be assessed concerning its trustworthiness. Actually, we think of setting up a static value between

0 and 1 for each recognition scheme implemented. This value would be the consensus between different security experts concerning individual technologies, but other means may be used such as integrity metrics. The short list below gives an example of which value may be used for different recognition schemes based on the average attack space (AAS) [27]. These values are neither definitive nor reviewed but they are helpful to demonstrate the idea.

In 1999, the company RSA Security recommended to use public keys (PK) of at least 768 bits, which has an AAS of 76 bits but that can be attacked off-line. Well-designed biometrics, those that can only be attacked interactively, are considered strong when the false acceptance rate (FAR) is around 1 in 1000000 [27]. So, we consider that schemes respecting at least the latter criterion would get near to 1. Biometrics with higher FAR would get a value in proportion with the criteria for strong biometrics (e.g. a FAR of 1/100000 would get 0.1). With higher FAR, enrollment can be achieved more dynamically because learning phase is simpler. Imprinting with strong key (e.g. 128-bit AES, which gives an AAS of 127 bits [27]) in the Resurrecting Duckling scheme [29] would get near to 1 because it respects our criteria when off-line attacks are possible.

4.2 A Peer Entity Recognition Scheme (APER)

The APER scheme (pronounced "ape-er", based on the word "aping", meaning imitating) is designed to be usable to recognise peers on a network. It is explicitly not designed to associate an identity with the recognised peer, other than in terms of the cryptographic artefacts involved in the protocol. However, it does not prevent higher-layer code from associating an identity with the recognised peer (which is where identity can more usefully be handled).

APER assumes that the network supports some form of "broadcast" or "multicast" messaging, for example using IP broadcast or multicast addresses, or adopting an application layer broadcast approach.

APER has not (yet) undergone peer review for its security properties, therefore we only indicate the properties we assume it to have, which is fine for current purposes since the scheme is really a proof-of-concept for the "recognition is enough" argument. It is certainly clear that other schemes with similar properties can be developed.

There are two roles distinguished in APER, the recogniser and claimant (though any party can take on any role). The basic approach is for the claimant to occasionally (according to its own schedule) broadcast a digitally signed packet (a "claim") and for the recogniser to be able to challenge the claimant as desired. However, in contrast to most such schemes, it is also considered useful and, though weaker, often sufficient for the recogniser not to issue challenges, but simply to recognise the peer on the basis of correctly signed (and, perhaps, co-dependent) claims.

When a challenge is issued, producing a correct response to the challenge requires the claimant to possess the private key used to sign some previous claims. The recogniser can then much more safely (re-)associate the public key with the claimant's network address or other context information. The claimant

may optionally include some context information (e.g. time, network-address, application-layer naming) in claims.

There is one further "trick" used in order to increase the recogniser's confidence that the claimant was responsible for previous claims. In order to provide evidence that the claim is "fresh", and not, e.g. copied from some other broadcast network, the claimant is required to (where possible) include within its claim the hashes over the last "n" claims which were seen on the network (by the claimant). If the recogniser has also seen one of these (the recogniser is assumed to record its own set of recently received claim hashes) then the recogniser can treat the claim as being "fresh".

So, we end up with three levels of recognition, any of which can be sufficient, depending on the end-to-end trust level set. Each level will have some associated parameters (e.g. the number of claims seen), which may also impact on how the recognition is treated. The levels are:

- Level 1: Claimants signature verified over a set of recently seen claims
- Level 2: Level 1 and claimants recent claims are "fresh", based on the "last-n-hashes" mechanism
- Level 3: Level 2 and the claimant successfully responded to a challenge

We describe the scheme in an algebraic notation in Table 2.

Table 2. APER scheme

Item	**Description**
"{x}y"	A digitally signed form of "x", verifiable using (and containing) public key "y"
"a,b"	The comma is used for catenation. "a,b" is the catenation of a and b
[x]	An optional field is enclosed in square brackets
C	Claimant
R	Recogniser
n, n',n"	Nonces, i.e. a long (say 128 bits) random values
PuC	A public key claimed by C
PrC	A private key that ought to be C's
Ctxt	(optional) Context information, e.g. time, network address, application scope
fresh	A value which provides evidence that the claim is "fresh", in this case, this contains the last-n-hashes value (during a bootstrapping sequence this may be empty)
this, that	Identifiers for claims used when binding claims together
c	A claim, c={n,[ctxt],fresh,[this,that]}PuC
chal	A challenge to C chal=n'
Resp	A response to chal. Resp={n",hash(chal)}PuC

For key hygiene and other reasons a claimant may own as many key pairs as it wishes. However, for application and storage limitation reasons it will be beneficial for a claimant to reveal to a recogniser that it "owns" two keys. This is done by creating a new pair of claims in which the "that" field of one contains the value of the "this" field of the other and vice-versa. The recogniser can then treat the pair of recognised entities as one. If the application context requires, the recogniser can ask the claimant to do this by sending along a pair of "this" values which it would like to see bound. Clearly, this key binding can be extended to arbitrary sets of keys, and we term such sets key-chains. A useful way to ensure this binding is for the "this" and "that" values to be hashes of the "pointed-at" public keys or claims.

It may be argued that by associating context information (e.g. a network address) with the public key, we are really authenticating that information and that therefore the context information is taking the part of an identity. That, however, is not the case, as can be seen if we consider a case where a network address is included in a claim in the presence of a network address translation (NAT) box, where the claimant and recogniser will "see" different network addresses. (Note: we are not recommending either inclusion of addresses, nor NAT, but just using those to show that recognition can work where authentication is problematic, at best!)

That brings us to considerations of the overall strength and security of APER. As we said we are not providing proofs at this stage (since they are not needed for the main argument of this paper), but we claim that APER:

1. Provides a strong recognition scheme when challenges are issued.
2. Requires an attacker to compromise a claimant's private key to succeed in a useful attack against a real claimant.
3. Provides a useful recognition scheme even for recognisers who do not issue challenges. In particular, if a recogniser associates a quality of recognition with each claimant for which claims have been seen (using an algorithm which is yet to be developed, but probably based on the time-span over which claims with this key-chain have been seen), then the scheme is such that spoofing an application layer identity can be made arbitrarily hard by the recogniser.

5 Pluggable Recognition Module — High-Level Design

Ubiquitous computing environments mostly imply MANETs but also include all other kinds of networks as well, most obviously the current IPv4 Internet for which many authentication protocols have been developed to verify the identity claimed by a principal. Most of these authentication schemes assume a managed network. Open and dynamic environments with heterogeneous parties do not have network managers in the sense assumed, or even proper infrastructure in the case of MANETs. Adaptability to an entity's capabilities but also to the legacy authentication schemes is therefore required. For this reason, we are investigating an entity recognition module into which different recognition schemes

can be plugged. The design of that pluggable recognition module (PRM) will be based on extending the Pluggable Authentication Module (PAM) [23]. PAM allows for the use of different legacy authentication schemes: Kerberos, smart cards, etc. In order to get dynamic enrollments, policies — regarding which authentication scheme or combination of authentication schemes should be used — cannot require an administrator to be effective. For pervasive computing, the degree of auto-configuration has to be increased. To achieve this, the appropriate recognition scheme must be negotiated either explicitly or implicitly (e.g. perhaps selected by an application). This negotiation should make use of the degree of trust needed which is set by the application. Choosing a weak recognition scheme, maybe one allowing for highly dynamic enrollment, will be possible but this choice will impact the end-to-end trust; the highest level of trust possible will be as high as the level of trust in the underlying technology. In doing so, privacy of the to-be-recognized entity can be taken into account during the negotiation. Certainly, if an authentication mechanism is not trustworthy, data exchanged cannot be really considered "private". In addition, the security of the recognizing entity is somehow protected because the end-to-end trust takes into account the technical infrastructure. Our approach implicitly takes into account that recognition impacts trust. Dynamic enrollment is possible because technical trust is taken into account. Therefore, as part of any recognition scheme, meta-data should be included to achieve a sufficient level of confidence in the recognition. We will have to create this feature in our PRM because PAM does not provide it.

6 Conclusion

Pervasive computing needs smooth dynamic enrollment to get the full benefits from spontaneous interactions with previously unknown entities. This unconditional enrollment creates a risk that should be made explicit, so that it can be taken into account. Under certain circumstances, due to limited capabilities of entities or the ad hoc nature of collaboration — the technical infrastructure available may be limited as well, and especially, the security of the underlying infrastructure can be very low. In fact, even seamlessly integrated in pervasive computing environments, the intermediate technical infrastructure is still a concern of trust. However this is only one layer of the end-to-end trust. Trust is also envisaged to facilitate opportunistic collaboration among ubiquitous digital roaming entities. Through continued collaboration, trust between such entities would be built and evolve. This notion takes place at the level of entities where technical components are abstracted. Trust in the underlying infrastructure and trust between entities form the end-to-end trust. Each layer has to be taken into account to assess the whole.

The foundation of computer security has been built upon authentication. With this view, for initial collaboration to be possible, it is first necessary to know with whom the interaction occurs. However, we argue that it is sometimes sufficient to find out whether previous collaboration has ended successfully rather

than to get a precise identity without information about the likely behaviour of the entity. Therefore, entity recognition provides a more general basis for dealing with end-to-end trust. Pervasive computing requires a pluggable recognition module into which more or less secure/dynamic schemes can be plugged. However, it is essential to acknowledge how far each scheme should be trusted by reflecting their implicit impact on the end-to-end trust.

There are remaining issues in this approach such as the correlation of trust associated with identifiers based on different recognition schemes but pointing to one unique entity. Although combining a sequence of more or less trusted recognition schemes can aid against spoofing, the ease of building Byzantine attacks in such open environments is another problem. Weak recognition schemes make it really hard to ensure accountability. Another open issue is the question of how to trigger recognition, especially in case of self-triggering? This requires a trade-off between computing power spent for recognition and for other relevant processing.

This work is sponsored by the European Union which funds the IST-2001-32486 SECURE project.

References

1. Abdul-Rahman, A. and Hailes, S.: A Distributed Trust Model. In: Proceedings of the 1997 New Security Paradigms Workshop, pp. 48-60, ACM, (1997).
2. Axelrod, R.: The Evolution of Cooperation. ISBN 0-465-02122-0, Basic Books Publishers, (1984).
3. Blaze, M., Feigenbaum, J., and Keromytis, A. D.: Keynote: Trust Management for Public-Key Infrastructures. In: Proceedings of the Cambridge 1998 Security Protocols International Workshop, pp. 59-63, Cambridge, England, (1998).
4. Blaze, M., Feigenbaum, J., and Lacy, J.: Decentralized Trust Management. In: Proceedings of the 17th IEEE Symp. on Security and Privacy, pp. 164-173, IEEE Computer Society, (1996).
5. Christianson, B. and Harbison, W. S.: Why Isn't Trust Transitive?. In: Proceedings of the Security Protocols International Workshop, University of Cambridge, (1996).
6. Ducatel, K., Bogdanowicz, M., Scapolo, F., Leitjen, J., and Burgelman, J.-C.: That's what friends are for. Ambient Intelligence (AmI) and the IS in 2010. In: the congress of Innovations for an e-Society, Challenges for Technology Assessment Berlin, Deutschland, 17 - 19 Oktober 2001, (2001).
7. Ellison, C.: The Trust Shell Game. In: Bruce Christianson, Bruno Crispo, William S. Harbison and Michael Roe (Eds), Proceedings of the 6th International Workshop on Security Protocols, Lecture Notes in Computer Science, ISBN 3-540-65663-4, pp. 36-40, Springer, (1998).
8. IEEE: Pervasive computing. IEEE Magazine, http://www.computer.org/pervasive/.
9. IETF: Public-Key Infrastructure (X.509). http://www.ietf.org/html.charters/pkix-charter.html.
10. ITU: The Directory: Overview of Concepts, Models and Service. ITU-T Rec. X.500, Information Technology - Open Systems Interconnection, (1993), http://www.itu.int/home/index.html.
11. Jensen, C. D.: Secure Collaboration in Global Computing Systems. In: ERCIM News, vol. 49, (2002).

12. Johnson, S.: Emergence. ISBN 0-140-287-752, (2001).
13. Jøsang, A.: The right type of trust for distributed systems. In: Proceedings of the 1996 New Security Paradigms Workshop, ACM, (1996).
14. Jøsang, A.: A Subjective Metric of Authentication. In: J.- J. Quisquater, Y. Deswarte, C. Meadows, and D. Gollmann, editors, ESORICS'98. Louvain-la-Neuve, Belgium, Springer-Verlag, (1998).
15. Jøsang, A. and Knapskog, S. J.: A Metric for Trusted Systems. In: Proceedings of the 21st NIST-NCSC National Information Systems Security Conference, (1998).
16. Khare, R.: Whats in a Name? Trust. 4K Associates, (1999), http://www.4k-associates.com/IEEE-L7-names-trust.html.
17. Kohl, J. and Neuman, B. C.: The Kerberos Network Authentication Service (Version 5). Internet Request for Comments RFC-1510, (1993).
18. Marsh, S.: Formalising Trust as a Computational Concept. PhD Thesis, Department of Mathematics and Computer Science, University of Stirling, (1994), http://citeseer.nj.nec.com/marsh94formalising.html.
19. Merriam-Webster: Merriam-Webster's Collegiate Dictionary. Website, http://www.m-w.com/.
20. Microsoft: .NET Framework General Reference: trust Element. Website, (2001), http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpgenref/html/gngrftrustsection.asp.
21. Philips: Philips Ambient Intelligence. Website, http://www.research.philips.com/InformationCenter/Global/FArticleSummary.asp?lNodeId=712.
22. Reiter, M. K. and Stubblebine, S. G.: Authentication Metric Analysis and Design. In: ACM Transactions on Information and System Security, vol. 2(2), pp. 138–158, (1999).
23. Samar, V. and Lai, C.: Making Login Services Independent of Authentication Technologies. Sun Microsystems, (1995), http://java.sun.com/security/jaas/doc/pam.html.
24. SECURE: Secure Environments for Collaboration among Ubiquitous Roaming Entities. Website, http://secure.dsg.cs.tcd.ie.
25. Seigneur, J. M., Abendroth, J., and Jensen, C. D.: Bank Accounting and Ubiquitous Brokering of Trustos. In: 7th Cabernet Radicals Workshop, (2002), http://citeseer.nj.nec.com/seigneur02bank.html.
26. Seigneur, J.-M., Farrell, S., and Jensen, C. D.: Secure ubiquitous computing based on entity recognition. In: Ubicomp'02 Security Workshop, Göteborg, (2002), http://www.cs.tcd.ie/Jean-Marc.Seigneur/publications/secureubicomper.pdf
27. Smith, R. E.: Authentication: from passwords to public keys. ISBN 0-201-61599-1, Addison Wesley, (2001).
28. Stajano, F.: Security for Ubiquitous Computing. ISBN 0470844930, John Wiley & Sons, (2002).
29. Stajano, F. and Anderson, R.: The Resurrecting Duckling: Security Issues for Adhoc Wireless Networks. In: Proceedings of the 7th International Security Protocols Workshop, pp. 172-194, (1999).
30. TCPA: TCPA Design Philosophies and Concepts Version 1.0. White paper, Trusted Computing Platform Alliance, (2000), http://www.trustedcomputing.org/docs/designv1_0final.pdf.
31. TCPA: Trusted Computing Platform Alliance. Website, http://www.trustedcomputing.org/.
32. Weeks, S.: Understanding Trust Management Systems. In: IEEE Symposium on Security and Privacy, Oakland, (2001).

33. Weiser, M.: The Computer for the 21st Century. Scientific American, (1991), http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html.
34. Weiser, M. and Brown, J. S.: Designing Calm Technology. In: PowerGrid Journal, v 1.01, (1996).
35. Wexler, J.: Wi-fi world. Network World, (2002), http://www.nwfusion.com/wifi/2002/main.html.
36. Zimmermann, P. R.: The Official PGP User's Guide. ISBN 0-262-74017-6, MIT Press, (1995).