Available online at www.sciencedirect.com
SCIENCE DIRECT®
Science of Computer Programming 54 (2005) 49–71
Science of Computer Programming
www.elsevier.com/locate/scico

The Claim Tool Kit for ad hoc recognition of peer entities

Jean-Marc Seigneur[a,*], Christian Damsgaard Jensen[b]

[a] *Trinity College Dublin, Ireland*
[b] *Technical University of Denmark, Lyngby, Denmark*

Received 16 January 2004; received in revised form 5 May 2004; accepted 20 May 2004
Available online 23 July 2004

Abstract

In ubiquitous/pervasive computing environments, it is envisaged that computing elements—entities—will start interacting in an ad hoc fashion. The peer-to-peer (p2p) paradigm is appealing for such types of interaction especially with JXTA, which supports the development of reusable p2p building blocks, which facilitate implementation on any smart device. However, the inability to rely on a centralised authentication infrastructure, the openness of the environment and the absence of an administrator (it is assumed to be too expensive to have a skilled administrator at hand due to the large number of peers) challenge the use of legacy authentication mechanisms.

Supporting spontaneous interactions among previously unknown entities requires dynamic enrolment of strangers and unknown entities. Entity recognition (ER) is a process that is carried out each time an interaction happens between entities in order to dynamically recognise previously met entities.

In this paper, we present the Claim Tool Kit (CTK), a Java-based implementation of ER: entities exchange messages, called Claims, and rely on their associated clues to evaluate the level of confidence in recognition.

The CTK employs advanced features available with Java, such as JXTA and Java Cryptography and Security Architectures. We show that the CTK needs performance results on these features in order to increase the level of auto-configuration of the CTK. We describe how to obtain performance assessment for some of these new features. Finally, we explain how the CTK can be instrumented to take into account performance assessment. By analysing the evaluation results, the applicability of these advanced Java-based technologies for peer entity recognition is assessed.

* Corresponding address: 2850 route nationale, 74120 Megève, France.
E-mail addresses: Jean-Marc.Seigneur@trustcomp.org (J.-M. Seigneur), Christian.Jensen@imm.dtu.dk (C.D. Jensen).
URL: http://www.trustcomp.org (J.-M. Seigneur).

0167-6423/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.scico.2004.05.005

© 2004 Elsevier B.V. All rights reserved.

Keywords: Authentication; Ad hoc peer-to-peer; Auto-configuration; Performance

1. Introduction

Weiser's vision of ubiquitous/pervasive computing [34] will only be realised when computing capabilities are woven into the fabric of everyday life, indistinguishable from it. Major companies in the household appliance market are increasingly getting involved in smart home appliances—appliances with communication, computation and storage capabilities. An appealing candidate technology for the implementation of these smart appliances is JXTA [13], which provides reusable peer-to-peer (p2p) building blocks facilitating implementation on any smart device. The mission of an ambient intelligence (AmI) environment is to enhance the space. However, challenges remain for the fulfilment of this mission in AmI environments. Auto-configuration and autonomy, especially from a security point of view [25], form part of these challenges. Billions of entities—potentially any smart device—are expected to spread in the surrounding environment. If the enrolment of all these entities always requires human intervention, the mission to enhance the space is defeated because it makes busy householders even busier. In addition, in home environments, no skilled administrator is present (because it is assumed to be economically non-viable to have an administrator at hand at all times [25]) and often most of the users are technology-unaware people. This absence of an administrator combined with the impossibility of relying on a centralised authentication infrastructure, the openness and the large scale of the environment challenge the use of legacy authentication mechanisms. This is especially true in unstructured p2p systems [33], which do not assume centralised directories and precise control over network topology or data placement.

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 (ER) [26], rather than traditional authentication schemes. A fundamental requirement for ubiquitous computing environments is to allow for potential interaction with unknown entities [26]. In public environments, there is no so-called list of known people to be enrolled. People roam from one space to another as they wish. This introduces the requirement for smooth dynamic enrolment to get the full benefits from spontaneous interactions with previously unknown entities, that is, the door should not be closed to strangers, but instead any stranger showing up at the door might become an acquaintance.

To allow for dynamic enrolment of strangers and unknown entities, we have proposed an entity recognition (ER) process [26], which consists of four steps:

(1) Triggering of the recognition mechanism.
(2) Detective Work to recognise the entity using the available recognition scheme(s).
(3) Discriminative Retention of information relevant for possible recall or recognition.
(4) Upper-level Action based on the outcome of recognition including a level of confidence in recognition.

In the following section, after recalling requirements and specifications of ER and the usefulness of Claims, we extend the design of ER to increase the level of auto-configuration. In Section 3, we give the specifications of a Claim Tool Kit and explain how to design a Java Claim Tool Kit compliant to the ER process. Section 4 describes two types of practical results: improvements resulting from applications using an alpha version of the CTK and performance results towards the auto-configuration of the CTK run on JXTA peers. Then, we present related work and conclude.

2. Background on ER and extensions

In this section, we briefly describe our entity recognition (ER) process (more fully detailed in previous work [26]) as well as a specific scheme which illustrates the usefulness of Claims. At the end of this section we present an extension to the ER process, which is motivated by the absence of an administrator and the need for "calm technology" [34]—requiring as little user intervention as possible. This requires a model of the ER module, which can be tuned and adapt itself to the current context.

2.1. The need for dynamic enrolment

In the Resurrecting Duckling security policy model [32], ducklings (e.g., a smart TV) have to take the risk to emerge from their shell in order to find their mother (e.g., the owner's smart remote control), who will look after them. The scenario can be extended to a broad range of 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. An example (detailed in [25]) is when a new smart lock (possibly running JXTA) is unpacked in a smart home and has to find out and enrol who the legitimate household tenants are using as little manual configuration as possible.

Generally, authentication schemes start with enrolment of entities. This task is often time-consuming and requires explicit human intervention, e.g., from a system administrator. For example, registering 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 [31].

This introduces the need for a solution for smooth dynamic enrolment, that is, the door should not be closed to strangers, but instead any stranger showing up at the door might become an acquaintance. To allow for dynamic enrolment of unknown entities, we have proposed an entity recognition process (fully detailed in [26]). Table 1 compares the current authentication process (AP) with our entity recognition (ER) process. There is no initial enrolment step at the beginning of the entity recognition process but this does not mean that enrolment cannot be done. Actually, in step E.3, if the entity to be recognised has never been met before, what will be retained is going to be reused the next time this entity is going to be recognised.

Depending on the recognition scheme, it should be more or less transparent, i.e., more or less like the enrolment step in A.1. Thus, by moving down the enrolment step in the process, we emphasise that the door is still open for interaction with strangers.

Table 1
Authentication and entity recognition side-by-side

Authentication Process (AP)	Entity Recognition (ER)
A.1. Enrolment: 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 recognise the entity to be recognised using the negotiated and available recognition scheme(s)
	E.3. Discriminative Retention (optional): "preservation of the after-effects of experience and learning that makes recall or recognition possible" [19]
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. Upper-level Action (optional): the outcome of the recognition is subsequently used in some way (loop to E.1)

An authentication scheme following the authentication process, as presented in column AP, can be integrated into an ER scheme (by doing enrolment at step E.3). An example of a pure recognition scheme is *A Peer Entity Recognition* scheme (APER is proposed in [26] and extended in Section 2.2). A number of different sensing, recognition and retention strategies can be envisaged for entity recognition schemes. The Detective Work depends on which recognition scheme is used; for example, in APER 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 recognise 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 have proposed to forget about entities (this may be based on context [9,29]) that the entity has not collaborated with after a certain time.

Our investigations (see the examples in Section 4.1) show that it is necessary to extend the ER process so that the outcome of the process may be a set of entities and an associated Level of Confidence in Recognition. Instead of recognising a single entity, an ER scheme may compute a probability distribution of recognised entities. A range of methods can be used to compute this distribution (e.g., using fuzzy logic or Bayes). For example, the ER scheme based on vision techniques (e.g., face template matching), called VER [30], is proactive: a person (p) among n persons previously recognised enters a room which is equipped with a biometric ER scheme. The outcome of recognition demonstrates hesitation between two persons: p_2 and p_3 are recognised at 45% and 55% respectively. These percentages represent the Level of Confidence in Recognition (lcr). So, all other

persons are given a lcr of 0%. We have the following probability distribution of recognised entities:

$$OutcomeOfRecognition = \sum_{i=1}^{n} (lcr_i, p_i) =$$
$$(0, p_1) + (0.45, p_2) + (0.55, p_3) + \cdots + (0, p_i) + \cdots + (0, p_n).$$

2.2. APER: an ER scheme example

The APER scheme (presented in [26]) is designed to be usable for recognising peers on a network. 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. In p2p systems, the implemented propagation scheme may be used (e.g., see the JXTA propagate pipe in Section 4.2.2). 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 broadcast a digitally signed packet (a Claim) and for the Recogniser to be able to challenge the Claimant as desired or simply to recognise the peer on the basis of correctly signed 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 Claimant may 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 Level of Confidence in Recognition. In order to provide evidence that the Claim is *fresh*, and not replayed or 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 or more 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*. 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:

- APERL1: Claimants signature verified over a set of recently seen Claims.
- APERL2: Level 1 and Claimants recent Claims are fresh, based on the last-n-hashes mechanism.
- APERL3: Level 2 and the Claimant successfully responded to a challenge.

A Claim c is composed of the following fields detailed in Table 2:

$$c = \{n, [ctxt], fresh, [this, that]\}.$$

Regarding a distribution of recognised peers, an extension to APER is to say that a Claim may be signed by n different keys, which can be seen as recognition clues. It may be because both keys are indeed owned by the same peer or two peers decided to form a group and sign the Claim as one entity. The following example is when an APER Claim is signed by two keys and both signatures are valid:

$$\sum_{i=1}^{n} (lcr_i, p_i), e.g., \{(APERL1, PublicKey1), (APERL1, PublicKey2)\}.$$

Table 2
APER Claim Format

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 value
Pub	A public key claimed by C
Pri	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]}Pub
chal	A challenge to C chal=n'
Resp	A response to chal. Resp={n",hash(chal)}Pub

It is worth mentioning that, depending on the key length and the cryptographic algorithm used, the clue given by signing Claims is more or less strong.

2.3. Extension for an autonomic model of ER

Assuming that the ER module can be used in a plethora of contexts varying from one extreme context to another (e.g., due to different types of hardware), in addition to the absence of skilled administrators and requiring minimal user intervention (if a user is present at all), an autonomic model of the ER module is needed. This model is depicted in Fig. 1. The basic pattern of an "autonomic element" [2] consists of a Management Unit and a Functional Unit. When we apply this pattern to our ER process, its four steps (namely Triggering, Detective Work, Discriminative Retention and Upper-level Action) become parts of the Functional Unit. The Management Unit is in charge of monitoring and tuning the ER module.

When the Level of Attention changes (that is, a level used to specify the percentage of incoming Claims that should be processed), the Triggering of the ER Module becomes more or less sensitive. In return, more or less computation is spent for Entity Recognition. This is useful when management of computation resources is required. A greater or lower Level of Detective Work means that the ER Module spends more or less time and applies more or fewer mechanisms to recognise current entities. A greater or lower Level of Discriminative Retention means that the ER Module retains more or less Recognition Information for later recognition.

Exceptions occurring during the entity recognition process should be used to react to potential attacks at the recognition level (e.g., Denial-of-Service due to too many

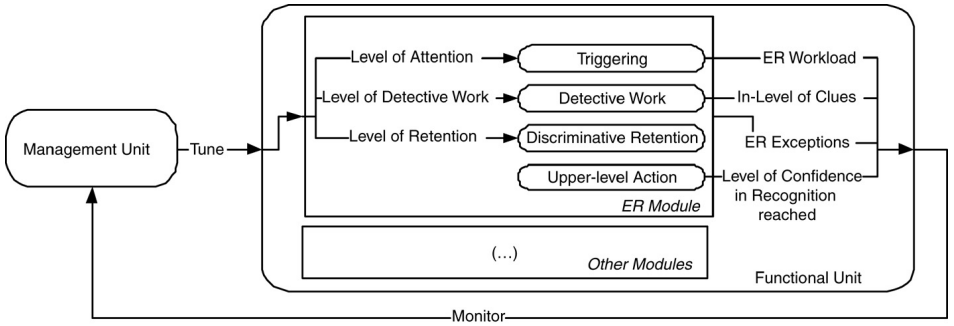

Fig. 1. The ER module in the Functional Unit.

Triggerings, Brute Force Attack over a set of possible observable attributes). Each time there is a Triggering and not enough Recognition Information clues provided, the ER module can log an optional piece of evidence summarizing this exception. Other kinds of ER Exceptions may be envisaged. This logging is represented by the loop called Monitor back to the Management Unit at the bottom of Fig. 1. Other optional useful pieces of evidence to be given back to the Management Unit consist of the Level of Confidence in Recognition reached after each recognition, the In-Level of Clues (that is, an estimation of the Level of Clues used by other Claimants) present and used for reaching these Levels of Confidence in Recognition and the ER current Workload. All these pieces of evidence can be used by the Management Unit, in addition to external information, to choose the most appropriate level inputs for the ER Module. For example, by knowing that the system is under Denial-of-Service attack, the Triggering of the ER process can be made less sensitive in order to decrease resources used for recognition.

The type of Management Unit can vary a great deal: “unlike conventional computing systems, which behave as they do simply because they are explicitly programmed that way, the management unit of an autonomic element will often have a wide range of possible strategies” [2]. The Management Unit is open to a broad panel of policies and decision-making mechanisms (e.g., from rule-based to trust-based [26] ones). This is why we focus on the Functional Unit of the CTK in the design section, which contains the generic functionalities. However, our investigations show that the Management Unit should take the context into account in order to improve decisions (see Section 4.1.1). The advantage of pervasive computing environments is that computing entities are context-aware—environmental information that is part of an application’s operating environment can be sensed by the application [4]. The access control of the autonomic element pattern can be enforced by tuning the Level of Attention: the lower the Level of Attention, the fewer the Claims processed.

3. Designing the Java CTK as an ER module

We first specify the main functionalities of a Claim Tool Kit (CTK). Then, we describe how to design and implement a CTK in Java compliant with the ER process.

3.1. Main functionalities of a CTK

In a CTK style of interaction between peers, peers claim statements by sending Claims over communication channels. An example of such a scheme is the APER protocol. Practical investigations (described in Section 4.1.1) have shown that it is possible and useful to use Claims in unicast communication channels. APER focuses on broadcast communication channels. However, a CTK goes beyond a broadcast communication channel (as assumed in APER) and the specific format of APER Claims. Generally, a Claim carries both content and clues (used for recognition during the Detective Work of the ER process). A CTK has two main responsibilities:

(1) it must allow a peer to recognise what peer made a Claim (due to its clues);
(2) it must allow a peer to make Claims over communication channels.

A CTK should allow a peer to suppose what other peers heard previous Claims. Our investigations have shown that recognition based on common knowledge through the exchange of Claims is worthwhile even without encryption (see Section 4.1.2). A peer can take two roles: Recogniser (i.e., a peer receiving a Claim) or Claimant (i.e., a peer sending a Claim). A Mastered Claimant is a Claimant which is either owned by the peer or controlled (that is, the peer can send a Claim under the name of the Claimant corresponding to the Mastered Claimant). As it should be difficult for a peer to spoof a Claimant, a peer sending a Claim is a Claimant for a Recogniser and a Mastered Claimant for itself. A CTK should allow a peer to manage several Mastered Claimants (i.e., to make Claims under the name of different Mastered Claimants), especially due to privacy protection, which is recommended (we have demonstrated an implementation of the CTK where the peer automatically sets the Mastered Claimant based on location [29]). Context-awareness and performance assessment of the CTK is optional. However, context-awareness, as well as performance assessment, is important for the Management Unit choosing the optimal configuration (see Section 4.2, which deals with performance). The outcome of recognition may be a probability distribution of Claimants. The CTK may provide the functionality to Link Claimants: to claim that one or more Claimants are indeed originated from the same peer, which owns these different Mastered Claimants (we underline the usefulness of this property in other work [28]). The functionality to Roll a Claimant to a new Claimant is related to the Link functionality in the sense that it is known that the old Claimant is linked to the new one but that it will not be used any longer (see how we used this against spam email in Section 4.1.2). An example of rolling occurs for key hygiene reasons: when a key pair is too old, it must be changed to a new one (as is provided in APER). Unicast, multicast and broadcast are required. The CTK may allow the use of different communication channels to send Claims in order to increase the likelihood of Claims propagation (in this case, special care is necessary to avoid over-counting the same Claim received on different channels). The notion of Claim made by a group of Claimants is optional. Due to the complexity of having different Mastered Claimants, in case human intervention is needed, the CTK should provide convenient user interfaces. In order to cope with scalability, the CTK should provide a mechanism to garbage collect what becomes useless (for example, resources spent to store recognition information relating to entities that are unlikely to be met again).

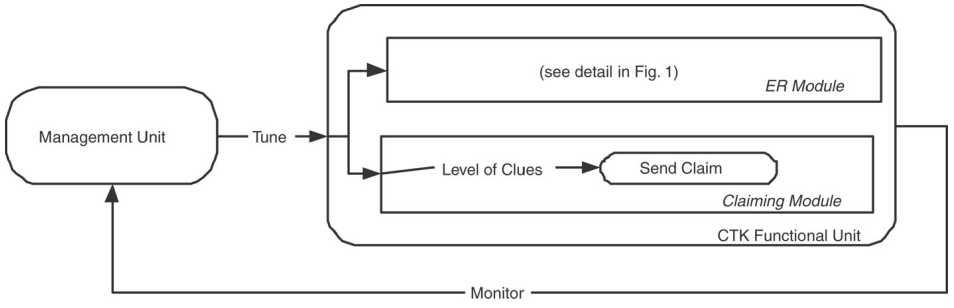

Fig. 2. The CTK Functional Unit.

3.2. A CTK in Java complying to the ER

The first step for obtaining a CTK compliant to the ER process is to add the Claiming functionality to the previous Functional Unit as depicted in Fig. 2.

The CTK has been implemented as a Java API.

We describe the main content of the core classes of the CTK and the derived classes for the APER implementation (package org.trustcomp.ctk.core and org.trustcomp.ctk.aper).

The main types in the CTK are:

- Claim: with `add/change/retrieve` content and `add/change/retrieve` recognition clues based on textual keywords.
- AssessedClaim: with a Claim (i.e., the received Claim) and `add/change/retrieve` assessment result Objects based on textual keywords (e.g., "reachedAPERLevel" is used to store and retrieve the Level of Confidence in Recognition that the Claim provided after Detective Work).
- Claimant: with a local Claimant identifier, RecognitionInformation and `roll` and `link` to other Claimant operations.
- RecognitionInformation: with `add/change/retrieve` Objects used for recognition of the Claimant (e.g., the "publicKey" used by the Claimant).
- MasteredClaimant: with `manageBuiltClaims` to keep track of what claims have been sent under this MasteredClaimant so far.

Claims are said to be heard by ClaimHearers. When Claimants send Claims, they can specify what ClaimHearer is supposed to hear (i.e., receive the Claim) as clues. When a Claim is received, the local receiver peer can associate as an assessment result what ClaimHearer is supposed to hear during the Discriminative Retention phase. The default implementation just copies the ClaimHearers, which are specified with the Claim. Other implementation may try to verify and carefully assess what ClaimHearer is to list in the final assessment result. For example, in broadcast, a special ClaimHearer, called `ClaimHearer.BROADCAST`, is always used because any Claimant is supposed to hear any Claims. This is the basic use-case of the APER scheme. However, because unicast is also needed (especially in email-based cases presented in Section 4.1.2), a ClaimHearer can specify a list of Claimants who heard the Claim.

Table 3
A CRC card example

Class Name	
Responsibility 1 of this Class	This Class collaborates with such and such other Classes to fulfil Responsibility 1
Responsibility 2 of this Class	Collaborate with … to fulfil Responsibility 2
…	…
Responsibility *n* of this Class	Collaborate with …

For Freshness (the mechanism for checking that different peers have a degree of common past Claims, e.g., the fresh field in APER), there are different possibilities for combining the past Claims. The number of past Claims to use can be specified to change the Level of Clues. The default combination consists of the calculation of the hash of each Claim and the resulting hashes in a List. Then, the Management Unit can tune when a Claim is considered to be fresh by setting how many hashes should be found in common locally and inside the new Claim. An example of an alternative combination of hash may be to use the hash of the previous hash (i.e., hash chaining).

An EROutcome object contains a list of RecognisedClaimants (pairs of Claimant/ConfidenceLevelInRecognition) and InformationForUpperLevelActions.

In order to obtain an internal maintainable CTK Java package, we spread the functionalities of the CTK (detailed in the previous subsection) in different high-level classes based on Class-Responsibility-Collaboration (CRC) cards [22]. We give an example of a CRC card in Table 3 (even though we use a textual representation thereafter).

The number of responsibilities *n* should be around 3 in order to keep a good design. A responsibility must be implemented if not specified otherwise.

The ClaimManager has the responsibilities: to manage the Claims (e.g., to retrieve/store Claims in collaboration with the AssessedClaimStore; to know what Claims are sent/received and how in collaboration with the ClaimSender); it should enforce policies concerning whether to send Claims or not (e.g., for privacy protection reasons depicted in Section 4.1.1) optionally based on context (this responsibility is closer to a Management Unit concern than the other ones).

The ClaimSender has the responsibilities: to register the possible communication channels, called ClaimSendables (e.g., a JXTA pipe or SMTP are used in Section 4), to be used; to send new Claims over a ClaimSendable; optionally to select the correct ClaimSendable based on the Level of Clues (this responsibility is closer to a Management Unit concern than the other ones).

The ClaimantManager has the responsibilities: to manage the Claimants and MasteredClaimants (e.g., to create them according to the correct cryptographic algorithms); it should enforce policies regarding the choice of the MasteredClaimant to pick/roll/link (e.g., for privacy protection reasons depicted in Section 4.1.1) optionally based on context (this responsibility is closer to a Management Unit concern than the other ones).

The ClaimantStore has the responsibilities: to store and retrieve Claimants and MasteredClaimants based on Recognition Information; it should garbage collect useless Claimants in collaboration with the ClaimantManager (for scalability reasons) optionally based on context (see an example algorithm in Section 4.1.1; this responsibility is closer to a Management Unit concern than the other ones).

The ClaimBuilder has the responsibilities: to build the Freshness to be sent with a new Claim (in collaboration with the ClaimManager) according to the specified cryptographic algorithms; to sign a new Claim according to the specified cryptographic algorithms; optionally to add more or less strong recognition clues depending on the Level of Clues given by the Management Unit.

The ClaimListener has the responsibilities: to discard/filter Claims according to the current Level of Attention; it should provide the ERWorkload (that is, information about what and how often the ER process is triggered).

The ClaimDetective has the responsibilities: to compute the Level of Confidence in Recognition from a Claim made by a Claimant (it may be a probability distribution of recognised Claimants); to do more or less Detective Work based on the current Level of Detective Work; optionally to provide the average In-Level of Clues.

The AssessedClaimStore has the responsibilities: to store and retrieve AssessedClaims based on the fact that retrieved Claims are selected if they are associated with a given set of ClaimHearer (in collaboration with the ClaimDiscriminativeRecognitionRetentor); it should garbage collect useless Claims in collaboration with the ClaimManager optionally based on context (this responsibility is closer to a Management Unit concern than the other ones).

The ClaimActioner has the responsibilities: to generate the EROutcome (in collaboration with the ClaimDetective, which gives the Level of Confidence in Recognition based on the Claim); to pass the EROutcome to registered EROutcomeListenable (the objects to be notified when new Claims come in); optionally to reply to a Challenge (e.g., it is needed for APER Challenge/Response).

The ClaimDiscriminativeRecognitionRetentor has the responsibilities: to retain more or less Recognition Information based on the current Level of Retention (in collaboration with the ClaimDetective, which gives the Level of Confidence in Recognition based on the Claim); it may carefully assess what ClaimHearers are to be associated with a specific Claim; it should react in the case of ERException (e.g., an exception is logged if a Claim with an invalid signature is received).

All these components are used to carry out the ER process. However, in order to achieve a more cohesive implementation of the logic and decrease coupling between the other objects, we applied the Mediator pattern, which uses an object to coordinate state changes between other objects instead of distributing the logic over the other objects. So, the ERProcessMediator class implements the flow between the main methods of the ERProcess, which are listed in the ERProcessable interface: `triggering`, `detectiveWork`, `discriminativeRetention`, `passForUpperLevelAction`, optional `forget` (that is, garbage collection to increase scalability).

From the core classes, we implemented the APER classes (depicted in Fig. 3) needed to run the APER scheme:

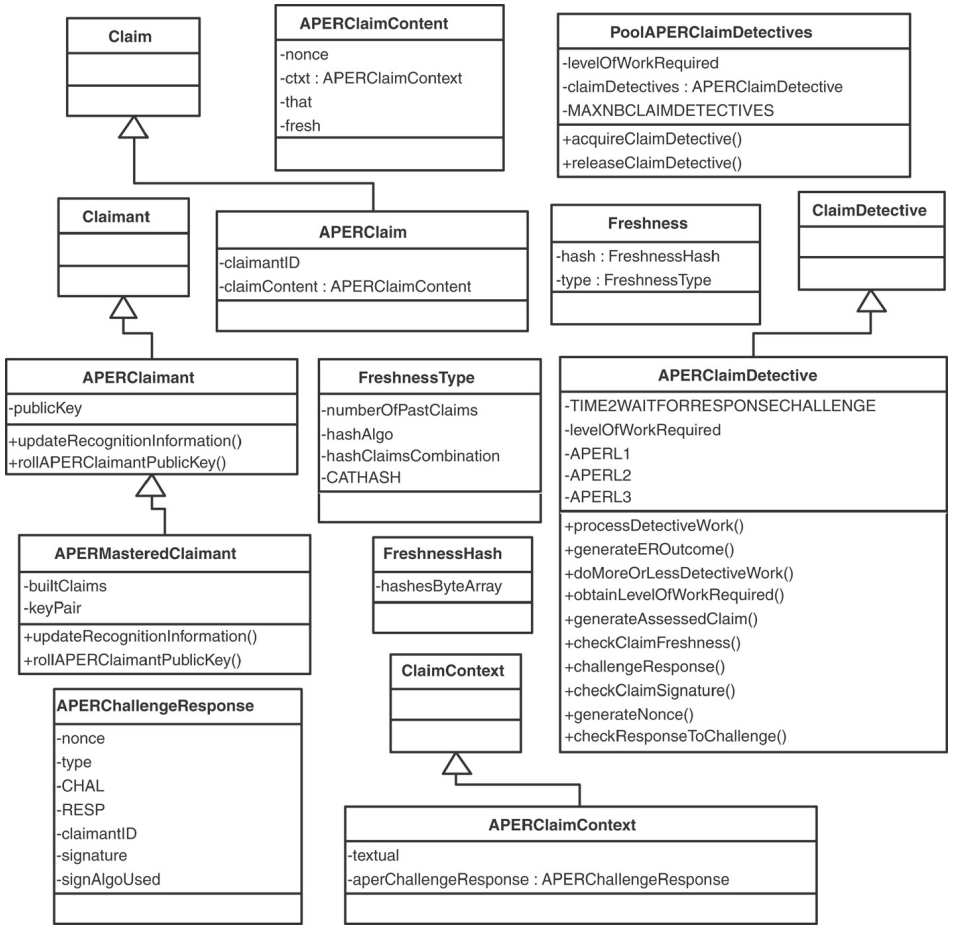

Fig. 3. Simplified main APER classes derived from the core classes.

- APERClaimant: implements Claimant and is mainly recognised by a PublicKey.
- APERMasteredClaimant: it has access to the PrivateKey associated with the PublicKey.
- APERClaim: implements Claim (the current implementation is based on a Hashtable) and the APERClaimContent contains the different fields described in Table 2.

The APERClaimDetective class is in charge of carrying out the detective work on Claims. Due to the fact that the APERL3 relies on Challenge/Response (C/R) Claims with the Claimant, the issue of no response from the Claimant demonstrates that the Detective Work can take more or less time and even never terminate. This is a general property of the Detective Work step; more Detective Work may lead to a better Level of Confidence in Recognition but the counterpart is that it also takes more time. This trade-off of Level of Confidence and time is related to the work done on performance (detailed in Section 4.2). Performance results help the Management Unit to choose the best trade-off. In order for

the CTK to handle other Claims whilst in the process of a C/R, the CTK uses a pool of APERClaimDetectives. There is also a maximum time that can be set to decide that the C/R has failed (even though no response Claim has been received).

4. The CTK in practice with Java and JXTA

There are two kinds of practical results: the first set consists of improvements resulting from the internal use of an alpha version of the CTK in different applications; the second set concerns performance results used towards the use of the CTK on JXTA peers.

4.1. Feedback from investigations

We have applied the alpha version of the CTK to two application domains: context-aware applications (e.g., based on location [29]) and email-based applications [27]. Even though the underlying API (i.e., the CTK) is not discussed in these applications (i.e., this is the purpose of this paper, to describe the CTK), building such applications helped to revise the functionalities of the CTK.

4.1.1. Usefulness of context

The CTK should take into account that a large number of peers can be present in pervasive computing environments at a time and each entity can have an open number of Mastered Claimants. We have assumed [29] that each person carries a resource-constrained device (e.g., one of the first mobile phones with Java) running a JXTA peer with a CTK, which claims statements on her/his behalf. In this case, there is a need for a solution providing scalability to an open number of Claimants: the garbage collection functionality of the CTK corresponds to the notion of forgetting in the ER process. We have proposed [9] relying on the fact that there is a social network, which forms a small-world network, in such a scenario (especially when peers are used by humans because this obviously increases the chance to obtain a social network). Newman [21] explains that in a random network of N peers knowing an average of z other peers, the number D of degrees of separation that we need to consider in order to reach all N peers in the network is $D = \log(N)/\log(z)$. In a *small-world network* (Newman [21] gives a review of models of the small world related to social networks), the average distance between pairs of peers should be comparable with the value it would have on the random graph. This is the first property of interest concerning the scalability: reaching a peer knowing another peer should not require too many hops even in a large network. For combining recommendations, circles of acquaintances must overlap. A clustering coefficient C is the average fraction of pairs of neighbours of a peer which are also neighbours of each other. In a random graph, $C = z/N$, which is too small for a large network. However, since the network is a small-world one, the value of C would make the use of recommendations feasible. In such a network, C may reach values found for real-world networks: significantly less than 1 but much greater than $O(N - 1)$[21]. In our application, location (i.e., one of the dimensions of context) is available. So, similarity to target peer in geography, it could be used in our Small-wOrld Forgetting Algorithm (SOFA) [9] to only keep Recognition Information on peers more likely to be found in the next location.

The CTK inherently enhances privacy protection because it allows persons to use different Mastered Claimants, which act as pseudonyms. In addition, the context-awareness of the CTK has shown to facilitate privacy protection. Indeed, privacy expectations vary and depend on context [29]. We have used the location of the CTK owner to disclose which Mastered Claimant makes Claims on his/her behalf [29]. The next type of application (described in Section 4.1.2) demonstrates that the CTK can also be used for privacy recovery.

4.1.2. The CTK applied in email

The CTK can be applied to a variety of communication channels. However, we picked Simple Mail Transfer Protocol (SMTP) as our first communication channel due to its actual security flaws, especially due to the fact that there is an open door for strangers to communicate through. Of course, this has created the problem of spam email. By applying the CTK to this dynamic enrolment issue, we show that a disposable email addresses can be implemented with the Roll functionality of the CTK and that this mechanism provides privacy recovery [27]. Once an email address is compromised (because spam email has been received meaning that it is in the hands of the spammers), the user can roll the email address to a new one and communicate this new email address to all supposed legitimate Claimants on this email address. Hence, there are many communication channels composed of email addresses. We called our solution Rolling Email Address Protocols on purpose: although we only presented Claimants recognised based on their email address, which is weak and easy to spoof, we have since implemented APERClaims sent over SMTP, which makes spoofing much harder. This shows that APERClaims can also be of use in non-broadcast environments.

Even if Claims are sent in the clear, knowing what peer heard previous Claims to some extent increases the level of security. In the case of email, it is easy to spoof email addresses and then send a fake email message to a peer. When keys are used, it becomes hard to spoof an email address because it requires one to either *guess* the private key associated with the public key or to get access to the private key. One may argue that if short keys are used, after a long time it becomes easy to guess the private key. Then, it starts to make sense to rely on previous email Claims, known to be shared with the other peer, because the spoofing peer must be able to eavesdrop on email messages, which requires more resources and is likely to be out of the range of remote spammers. Obviously, if spammers have access to local private keys, they can also find local hashes of previous emails. However, this requires breaking into the local storage, which can be argued to be too resource-consuming for spammers. Spam is only effective because of its bulk nature and near-zero per-message cost (please refer to [16]). Therefore we consider large-scale eavesdropping and Man-in-the-Middle attacks of the size required to circumvent our system for the number of users required for spam to be economical to be impossible. So, we also provide a Level of Confidence in Recognition checking that previous hashes of emails are correct without the use of key signatures.

4.2. Performance

In the CTK process, performance plays a key role for resource-constrained peers (as Recognisers or as Claimants) when they attempt to tune their Levels of Attention, Detective

Work, Retention and Clues in Claims. We first depict a scenario where it makes sense to know performance in order for the Management Unit to choose Levels which will optimise what can be achieved. Then, we study what needs to be done to get performance results for such an example with JXTA peers. The end of this section explains why and how to carry out performance assessment at run-time.

4.2.1. Motivation by example

We take the example of peers that can only communicate for short periods of time called "contacts" (e.g., in Delay Tolerant Networks [6]) with few of these periods, which become real communication opportunities. Further, the peers that can be encountered during these contacts are not all known and their work consists of sending Claims to specific peers. Some of the peers present at time of contact are malicious and try to spoof other peers. There are two communication channels for peers sending Claims to specific peers: a weakly secure communication channel and a more secure communication channel. The more secure communication channel takes 50% more time to send a Claim than the other one. The contact time allows each peer to send 100 Claims over the weakly secure communication channel and 50 Claims over the more secure communication channel. The peer's mission is to optimise the number of successfully sent Claims to specific peers during each contact. If the peer knew that the success rate with the weakly secure communication channel is 40% and 100% with the more secure communication channel, the Management Unit should set the Level of Clues that commands the CTK to use the more secure communication channel. The best strategy to be adopted depends on the percentage of malicious peers present. However, this information is unknown. It is up to the Management Unit to solve the problem of finding the best strategy. An example may be to test a small portion of the contact time over weakly secure communication, to check the success rate, then to test the same small portion of time over more secure communication, to compare the success rates and decide over which communication channel the remainder of the contact time should be spent. As an aside to this example, a peer entering a new network could choose the strategy without sending Claims by looking at what Level of Clues are used in Claims sent by already present peers and set its Level of Clues to the same level assuming that the majority of present peers are not malicious. The end of this section focuses on what the CTK is responsible for: to provide performance results in order for the Management Unit to make well-funded decisions.

4.2.2. Manual assessment of JXTA-Java pipes

Our previous example requires that a peer can use different communication channels with different security strengths and easily switch from one communication channel to another one. This is easily achievable for a JXTA peer thanks to one of the main abstractions in JXTA, which is the concept of pipe. A pipe describes a connection between a sending endpoint—encapsulation of the native network interfaces provided by a peer—and one or more receiving endpoints. A pipe—a kind of virtual communication channel—is used to conveniently connect peers and to send messages between them because a network transport can be accessed without interacting directly with the endpoint abstraction. Any transport capable of unidirectional asynchronous unreliable communication can be used; indeed JXTA specifications specify that the default service

pipe provides unidirectional asynchronous unreliable communication. Different endpoint transport implementations are available (e.g., TCP or HTTP). The Pipe Binding Protocol dynamically resolves the set of currently listening peers to specific pipes. As for other resources in JXTA, pipes are uniquely identified by a pipe advertisement, which is used to discover which pipes are available and to resolve the input pipe and output pipe endpoints. The pipes support two modes of communication: point-to-point, i.e., connecting exactly two pipe endpoints; propagate pipe, i.e., connecting one output pipe to multiple input pipes. JXTA implements TLS [6] to secure the communication through pipes. The following keywords are used to easily switch between the different kinds of pipes: JxtaUnicast (i.e., unicast, unreliable and unsecure pipe); JxtaUnicastSecure (i.e., unicast and secure pipe); JxtaPropagate (propagated, unreliable and unsecure pipe). Thus, in our previous example, if a peer wants to send a Claim to another peer, two pipes must be registered as ClaimSendables in the CTK: one pipe specified with the JxtaUnicast keyword and a second one with the JxtaUnicastSecure keyword. From this point, the Management Unit can specify what communication channel to use because knowing that TLS is used increases the Level of Clues.

However, we had to define how to obtain performance results about each communication channel (e.g., the type of pipe). Such results are needed for the Management Unit to choose the best strategy. The testing method (fully detailed in [24]) is based on the "performance assessment framework for distributed object architectures" [12]. Nevertheless, some changes have been made to get results more appropriate for the JXTA pipe paradigm. The JXTA model is not a distributed object model as such. Our testing method may be used to assess the performance of pipes connecting two peers separated by a context-dependent number of peers with context-dependent types of transport between them. In this paper, we focus on our example. So, we only present results for one specific configuration called "TCP only", which consists of direct physical connection between two peers using TCP.

Among the different criteria for quantitative evaluation of performance, three criteria are relevant for our case: Round Trip Time (RTT), throughput, data throughput. The definition of each criterion had to be adapted though:

- RTT—measures the time needed from the point when the Claimant peer sends a message (i.e., a Claim) to the Recogniser peer to the point when the Claimant receives the message back. Any other processing needed for sending and receiving the message on both peers is minimised as much as possible.
- Throughput—measures the number of message round trips in a given time interval. Throughput and RTT are inversely proportional. This is the reason that this paper presents only RTT results.
- Data throughput—measures the efficiency of data transfer. The data are transferred as elements in the message sent and returned. Our results are for data as strings of text.

The same Java source code was used for the different tests. The local results are for a PentiumIV 1.7 GHz with 256 MB RAM and the LAN results are for this computer and a PentiumII 450 MHz with 128 MB RAM connected to a 10 Mb Ethernet hub (the complete description of software and hardware is detailed in [24]). Time was measured with the `System.currentTimeMillis` method. The precision of this method is 1 ms under Linux

and 10 ms under Windows. Different issues arose when we assessed the accuracy of our results. The most important issue concerned the calculation of the necessary number of observations. Another issue involved the analysis of the steady state. We had to define how many observations were needed to get an appropriate level of confidence. There are two requirements on X_i, one occurrence of what is observed, to allow reliable data analysis [15]:

(1) Each X_i should be normally distributed. To satisfy this first requirement, we carried out experiments to check that the arithmetic mean of a sufficiently large batch of individual observations is close to normal. This is the reason that we measure the time of one thousand round trips.
(2) All X_i should be independent. We also carried out experiments to study the issues related to the steady state. Ideally, any data obtained from the transient period must be discarded, but practically this is not easy to detect [15]. Nevertheless, in the case of JXTA pipes, it looks like the first one thousand round trips of messages are processed slightly more slowly than the rest of the round trips. The standard deviation is also higher, showing that it is not a steady state. The slow start may be due to the initial creation of objects needed for processing the requests. However, we found that it is not really significant compared to the total number of round trips and thanks to the fact that we tripled the tests.

Since the two requirements are fulfilled according to our experiments, we can now calculate n, the number of observations necessary to obtain an accuracy of $\pm r = 1\%$ on loops of one thousand round trips of message with a confidence level of $100(1 - a) = 99\%$ for the whole range of string sizes. We ran 400 repetitions of one thousand round trips of message in order to be able to apply the following formula to find the necessary number of observations:

$$n = \left(\frac{100zs}{r\underline{X}}\right)^2 \tag{1}$$

where $\underline{X}$ is the mean, s is the standard deviation and z is the inverse of the standard normal cumulative distribution with a probability of $(1 - a)$. Hence, we validated that fifty repetitions of one thousand round trips is enough to get this level of confidence in our results for the whole range of string sizes. In all cases (see Fig. 4), the RTT looks linearly dependent on the string size. It can be approximated with a linear function in the form

$$RTT(StringSize) = k^*StringSize + Offset. \tag{2}$$

In Table 4, there is an estimation of the RTT function for each configuration calculated by linear regression analysis by using the "least squares" method to fit a line through the set of observations.

Fig. 4 also shows that to use secure communication via JXTA secure unicast pipes implies an overhead as expected. Of course, secure communication is more useful under a LAN configuration. Roughly speaking, the secure overhead grows from 125% for a string of 1 character to 300% for a string of 30000 characters.

After all this performance assessment work, we obtain that k for secure communication is approximately 4.6 times bigger than without security. This kind of assessment is clearly

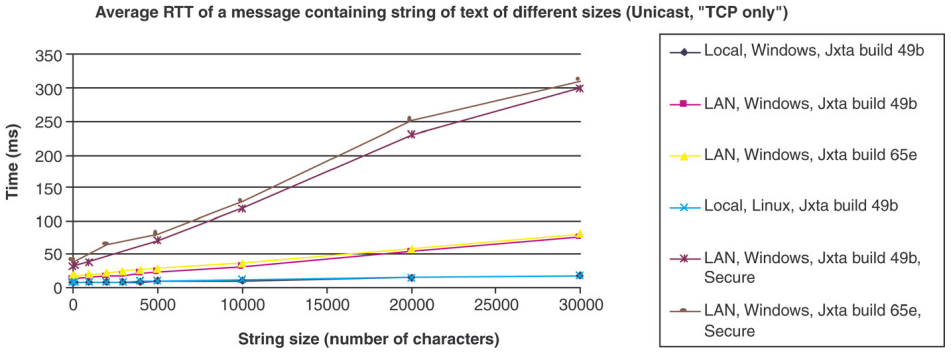

Fig. 4. Data throughput.

Table 4
RTT(Stringsize) linear functions

"TCP only"	Offset (ms)	*k* (ms/character)
Local, Windows, JXTA build 49b	7.6718	0.0003
Local, Linux, JXTA build 49b	8.1387	0.0003
LAN, Windows, JXTA build 49b	12.3042	0.0021
LAN, Windows, JXTA build 65e	17.7732	0.0020
LAN, Windows, JXTA build 49b, Secure	30.1358	0.0092
LAN, Windows, JXTA build 65e, Secure	39.8799	0.0095

time-consuming, requires a lot of human intervention, requires that the peers are not processing other time-consuming tasks during the test and demonstrates that obtaining statistically meaningful performance results requires one to carefully run resource-consuming tests. This supposes that approximation techniques may be needed to obtain dynamic performance results, which is what we aim at in the next subsection.

4.2.3. Towards performance assessment at run-time

The JXTA protocols specification is implemented in many languages (C, Java, etc.). Just with the JXTA-Java implementations (J2SE and J2ME), peers can be embedded in a broad range of devices with computational resources going from one extreme to the other (e.g., from the latest state-of-the-art server to the first generation of mobile phones embedding a JVM). Static performance results are very sensitive to change in hardware configurations and so can rarely be applied anywhere. JXTA-Java offers a "code once, run anywhere" within a virtual network. However, as yet performance results cannot be applied anywhere.

In order to allow the use of performance results in a broader range of contexts, we have started to instrument the code of the CTK to obtain results at run-time. These results are approximations but we argue they can be useful because they are generated in the real context. For example, this at least gives an idea that signing a Claim takes roughly 100 ms on a server and 2 s on a mobile phone.

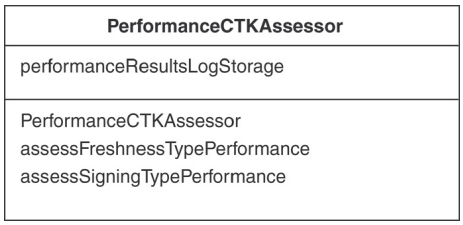

Fig. 5. The PerformanceCTKAssessor class.

For 30 batches of 1000 Claim signings	Mean time to sign 1 Claim (ms)
Sign : Freshness hash SHA-1 past Claims 4 content 10000 characters Mastered Claimant 7 key algo DSA key length 1024 sign algo SHA1	17.67421667
Sign : Freshness hash SHA-1 past Claims 4 content 10000 characters Mastered Claimant 2 key algo RSA key length 1024 sign algo SHA1	36.42108333
Sign : Freshness hash SHA-1 past Claims 4 content 10000 characters Mastered Claimant 2 key algo RSA key length 1024 sign algo MD5	36.50495
Sign : Freshness hash SHA-1 past Claims 0 content 10000 characters Mastered Claimant 2 key algo RSA key length 1024 sign algo SHA1	35.68958333
Sign : Freshness hash SHA-1 past Claims 4 content 10000 characters Mastered Claimant 1 key algo RSA key length 2048 sign algo SHA1	234.0156333

Fig. 6. A Signing Claim performance results example.

The goal is to apply the same steps as in our static performance assessment: estimation of the minimal number of individual observations that a batch of observations should contain in order to obtain a normal distribution and be confident in our results, then carrying out the tests to obtain the results.

The obvious performance results of interest in the CTK are the times for building the Freshness of Claims and for signing these Claims according to the different cryptographic algorithms. Thus, we have added a new subpackage in the CTK called org.trustcomp.ctk.dynperf, which contains the class described in Fig. 5. The current implementation logs the time taken by the operations in a file as depicted in Fig. 6 (the results are for a PentiumIV 1.7 GHz with 256 MB RAM and the CTK compiled and run with the Sun Microsystems J2SE JDK1.4.2-b28 and the JVM under Hotspot mixed mode). The minimal number of batches of the minimal number of unique observations (e.g., Claim signing or time to build the Freshness) as well as the algorithms used are specified as parameters. These methods can be called at run-time by the Management Unit in order to get an estimation of the time taken by these operations on the current peer in the current context. In the case of Fig. 6, we know that if we increase the Level of Clues from an RSA key size of 1024 bits to 2048, the number of Claims which can possibly be sent in a window of time is already approximately divided by 6.75. Also, the number of past hashes of past Claims has little impact for the signing operation (because a hash consists of few bits compared to the content of the Claim). The time taken for building the hash can be taken into account by calling `assessFreshnessTypePerformance`.

5. Related work

Following our work on the JXTA-Java pipes performance [24], Halepovic and Deters [10] propose a model for the assessment of JXTA, which encompasses pipes. As new JXTA builds may improve the performance of pipes, new results can be found

from the JXTA Benchmark project [14]. A new interesting communication channel, called CBJX, based on crypto-id (e.g., an example applied to email addresses may be: SecureHash(ClaimantPublicKey)@mailserver.something) is available in the JXTA J2SE 2.2 December 2003 release. This Message Transport allows for low-cost message authentication.

The relation between performance and auto-configuration has been studied in the case of IPv4 [1]. A Java-based infrastructure for multi-agent systems is also instrumented to obtain performance assessment at run-time in order to adapt control inputs [17].

Pseudonymity [8,18] and dependable multiple virtual identities [3] (which include federated identity management) are valuable for understanding the implications of using many Mastered Claimants. Regarding the use of keys in emails and the low adoption of other solutions based on this approach, GnuPG [7] reminds us that extra work is needed to manage these tools in order to obtain the protection that they can give. Large adoption goes with convenience of use. The new "enrol" [11] IETF Working Group aims at creating enrolment mechanisms based on keying material that are easily used by users. The CTK can be tuned to what the user wants.

More generally as regards p2p systems, the term peer-to-peer is applied to a large range of technologies. Such p2p technologies are more likely to adopt a highly distributed network-based computing style without centralised control where a critical function is performed thanks to cooperation between peers [20]. In addition to improving the performance of information discovery, content delivery and information processing, such a p2p manner of approach may also enhance the overall reliability and fault-tolerance of applications. In some cases, the cooperation between peers is the only solution for these peers for reaching a goal that they could not achieve alone. For example, in mobile ad hoc networks (MANETs), where peers have short range wireless communication and cannot reach a distant Internet gateway, the peers on the path to the gateway forward the messages of further peers. Collaboration between peers is also used to cope with uncertainty (e.g., the value measured by a group of peer sensors is calculated on the basis of a majority vote). Assuming the absence of an attacker in interdependent settings is not realistic though (as has been shown in MANETs, where the major routing algorithms had to be revised). That is why trust and reputation mechanisms have been proposed for large open environments for solving the issues due to the probable presence of malicious or faulty peers [35].

However, the results of majority votes as well as reputation and trust metrics are challenged by the easy creation of digital identities without centralised authorities due to the "Sybil attack" [5]. This attack consists of the creation of many virtual identities by one real peer, which all vote for one specific virtual peer in order to be granted an asset or to undermine the collaborative system. The CTK focuses on the recognition level and the Level of Confidence in Recognition can be used in the trust metric in order to minimise such an attack. The end-to-end trust [26] can be a function of the trust in the technical infrastructure, which includes the recognition mechanisms, and the trust in the peer based on past evidence. The main principle is to limit the end-to-end trust depending on the security strength of the technical infrastructure even if the trust in the peer is very high. Future work is required to study how to combine the outcome of recognition with computational trust engines (such as the one developed in the SECURE project [23], which provides an advanced trust/risk-based security framework including an ER module).

6. Conclusion

Dynamic enrolment is needed in pervasive computing. However, simply opening the door to any stranger leads to security and privacy issues. A current example occurs in the email system, where spammers take advantage of this opened door to send spam emails. The trend is to add strong authentication based on cryptography but dynamic enrolment of cryptographic keys cannot easily be achieved with current authentication mechanisms. We propose an entity recognition process to provide dynamic enrolment of entities, whilst keeping a degree of security due to a level of confidence in recognition. Further, our model follows the "autonomic element" pattern in order to increase the level of auto-configuration. The Claim Tool Kit is the solution based on this ER process for message-based applications.

By using the CTK in email-based systems, we show that the notion of recall (based on hashes of past messages done during Discriminative Retention) improves the level of security without any change to SMTP (i.e., the Claims are sent over SMTP). Keys can also transparently be rolled when needed. The notion of rolling sheds light on the use of rolling email addresses as a tool against spam and for privacy recovery. Other investigations highlight the importance of context information and especially performance for choosing the best configuration of the CTK. For example, care must be taken in the configuration of the Detective Work, which can offer more or less confidence but take more or less time and even not terminate (e.g., in the case of no response to a challenge). Even though performance at run-time is difficult to achieve, the CTK is being instrumented to use (approximated) performance results to improve auto-configuration. Further use of the CTK on JXTA peers and in combination with computational trust engines is under scrutiny. The functionalities of the CTK seem to be now finalised. However, from an implementation point of view, nothing is frozen. The beta version of the CTK is made to be tested by as many programmers as possible. There is still a round to go for a robust API.

Acknowledgements

We are grateful to the JXTA developers community. This work is sponsored by the European Union, which funds the IST-2001-32486 SECURE project [23].

References

[1] H. Bohnenkamp, P.v.d. Stok, H. Hermanns, F. Vaandrager, Cost-optimisation of the IPv4 Zeroconf Protocol, in: Int. Symp. on Dependable Systems and Networks, DSN 2003, IEEE CS Press, 2003, pp. 531–540, http://www.cs.kun.nl/ita/publications/papers/fvaan/IPDS03.pdf.

[2] D.M. Chess, C.C. Palmer, S.R. White, Security in an autonomic computing environment, IBM Systems Journal 42 (1) (2003).

[3] E. Damiani, S.D.C.d. Vimercati, P. Samarati, Managing multiple and dependable identities, IEEE Internet Computing 7 (6) (2003) 29–37.

[4] A.K. Dey, Understanding and using context, Personal and Ubiquitous Computing Journal 5 (1) (2001) 4–7, http://www.cc.gatech.edu/fce/ctk/pubs/PeTe5-1.pdf.

[5] J.R. Douceur, The Sybil attack, in: Proceedings of the 1st International Workshop on Peer-to-Peer Systems, 2002, http://research.microsoft.com/sn/farsite/IPTPS2002.pdf.

[6] DTN, Delay Tolerant Network, Website, Internet Research Task Force, http://www.dtnrg.org.

[7] GnuPG, The GNU Privacy Handbook, The Free Software Foundation, 1999, http://www.gnupg.org/gph/en/manual.html.
[8] I. Goldberg, A pseudonymous communications infrastructure for the Internet, Ph.D. Thesis, University of California at Berkeley, 2000, http://www.isaac.cs.berkeley.edu/~iang/thesis-final.pdf.
[9] E. Gray, J.-M. Seigneur, Y. Chen, C.D. Jensen, Trust propagation in small worlds, in: Proceedings of the First International Conference on Trust Management, 2003, http://www.cs.tcd.ie/publications/tech-reports/reports.03/TCD-CS-2003-14.pdf.
[10] E. Halepovic, R. Deters, JXTA Performance Model, Future Generation Computer Systems, Special issue on Peer-to-Peer Computing and Interaction with Grids, Elsevier (in press).
[11] IETF, Credential and provisioning (enroll), Website, http://www.ietf.org/html.charters/enroll-charter.html.
[12] B.M. Juric, T. Welzer, I. Rozman, M. Hericko, B. Brumen, T. Domanjko, A. Zivkovic, Performance assessment framework for distributed object architectures, in: ADBIS'99, September, 13–16, 1999, Lecture Notes in Computer Science, vol. 1691, Springer, 1999, pp. 349–366, http://lisa.uni-mb.si/~juric/PerfAssessmentFrw.pdf.
[13] JXTA, Project JuXTApose, Website, www.jxta.org.
[14] JXTABENCH, JXTA benchmark project, Website, http://bench.jxta.org.
[15] K. Kant, Introduction to computer system performance evaluation, ISBN 0-07-033586-9, McGraw-Hill, 1992.
[16] R. Kantola et al., Peer to peer and SPAM in the Internet, Technical Report of the Helsinki University of Technology, 2004, http://www.netlab.hut.fi/opetus/s38030/F03/Report-p2p-spam-2003.pdf.
[17] K. Kleinmann, R. Lazarus, R. Tomlinson, An infrastructure for adaptive control of multi-agent systems, in: Proceedings of KIMAS'03 the International Conference on Integration of Knowledge Intensive Multi-Agent Systems, IEEE, 2003, http://cougaar.org/docman/view.php/17/104/Kimas03KKleinmann.pdf.
[18] A. Kobsa, J. Schreck, Privacy through pseudonymity in user-adaptive systems, ACM Transactions on Internet Technology 3 (2) (2003) 149–183, http://www.ics.uci.edu/~kobsa/papers/2003-TOIT-kobsa.pdf.
[19] Merriam-Webster, Merriam-Webster's Collegiate Dictionary, Website, http://www.m-w.com/.
[20] D.S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, Z. Xu, Peer-to-peer computing, Hewlett-Packard Technical Report, HPL-2002-57, 2002, http://citeseer.nj.nec.com/milojicic02peertopeer.html.
[21] M.E.J. Newman, Models of the small world: a review, J. Stat. Phys. 101 (2000) 819–841, http://citeseer.nj.nec.com/487139.html.
[22] F. Scott, UML Distilled, Addison Wesley, 2000.
[23] SECURE, Secure Environments for Collaboration among Ubiquitous Roaming Entities, Website, http://secure.dsg.cs.tcd.ie.
[24] J.-M. Seigneur, G. Biegel, C. Damsgaard Jensen, P2p with JXTA-Java pipes, in: Proceedings of the 2nd International Conference on the Principles and Practice of Programming in Java, ACM, 2003, http://portal.acm.org/citation.cfm?id=957350.
[25] J.-M. Seigneur, C. Damsgaard Jensen, S. Farrell, E. Gray, Y. Chen, Towards security auto-configuration for smart appliances, in: Proceedings of the Smart Objects Conference 2003, 2003, http://www.grenoble-soc.com/proceedings03/Pdf/45-Seigneur.pdf.
[26] J.-M. Seigneur, S. Farrell, C.D. Jensen, E. Gray, Y. Chen, End-to-end trust starts with recognition, in: Proceedings of the First International Conference on Security in Pervasive Computing, 2003, http://www.cs.tcd.ie/publications/tech-reports/reports.03/TCD-CS-2003-05.pdf.
[27] J.-M. Seigneur, C.D. Jensen, Privacy recovery with disposable email addresses, Understanding Privacy, 1 (6), IEEE Security & Privacy, 2003 (special issue), http://www.computer.org/security/v1n6/j6sei.htm.
[28] J.-M. Seigneur, C.D. Jensen, Trading privacy for trust, in: Proceedings of iTrust'04 the Second International Conference on Trust Management, LNCS, Springer-Verlag, 2004.
[29] J.-M. Seigneur, C.D. Jensen, Trust enhanced ubiquitous payment without too much privacy loss, in: Proceedings of SAC 2004, ACM, 2004.
[30] J.-M. Seigneur, D. Solis, F. Shevlin, Ambient intelligence through image retrieval, in: Proceedings of the 3rd International Conference on Image and Video Retrieval, LNCS, Springer-Verlag, 2004.
[31] R.E. Smith, Authentication: from passwords to public keys, ISBN 0-201-61599-1, Addison Wesley, 2001.
[32] F. Stajano, R. Anderson, The resurrecting duckling: security issues for ad-hoc wireless networks, in: Proceedings of the 7th International Security Protocols Workshop, 1999, pp. 172–194, http://citeseer.nj.nec.com/stajano99resurrecting.html.

[33] B. Traversat, A. Arora, M. Abdelaziz, M. Duigou, C. Haywood, J.-C. Hugly, E. Pouyoul, B. Yeager, Project JXTA 2.0 Super-Peer Virtual Network, Sun Microsystems, 2003, http://www.jxta.org/project/www/docs/JXTA2.0protocols1.pdf.

[34] M. Weiser, The computer for the 21st century, Scientific American, 1991, http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html.

[35] W. Yao, J. Vassileva, Trust and reputation model in peer-to-peer networks, in: Proceedings of the Third International Conference on Peer-to-Peer Computing, 2003, http://csdl.computer.org/comp/proceedings/p2p/2003/2023/00/20230150abs.htm.