Electronic Communications of the EASST
Volume 22 (2009)

Proceedings of the Third International Workshop on Formal Methods for Interactive Systems (FMIS 2009)

Towards the Verification of Pervasive Systems

Myrto Arapinis, Muffy Calder, Louise Denis, Michael Fisher, Philip Gray, Savas Konur, Alice Miller, Eike Ritter, Mark Ryan, Sven Schewe, Chris Unsworth, Rehana Yasmin,

15 pages

Guest Editors: Michael Harrison, Mieke Massink
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Towards the Verification of Pervasive Systems

Myrto Arapinis[a], Muffy Calder[b], Louise Denis[c], Michael Fisher[d], Philip Gray[e], Savas Konur[f], Alice Miller[g], Eike Ritter[h], Mark Ryan[i], Sven Schewe[j], Chris Unsworth[k], Rehana Yasmin[l],

[a] m.d.arapinis@cs.bham.ac.uk [h] e.ritter@cs.bham.ac.uk
[i] m.d.ryan@cs.bham.ac.uk [l] r.yasmin@cs.bham.ac.uk
School of Computer Science, University of Birmingham

[c] l.a.denis@liverpool.ac.uk [d] mfisher@liverpool.ac.uk
[f] s.konur@liverpool.ac.uk [j] sven.schewe@liverpool.ac.uk
Department of Computer Science, University of Liverpool

[b] muffy@dcs.gla.ac.uk [e] pdg@dcs.gla.ac.uk
[g] alice@dcs.gla.ac.uk [k] chrisu@dcs.gla.ac.uk
Department of Computer Science, University of Glasgow

Abstract: Pervasive systems, that is roughly speaking systems that can interact with their environment, are increasingly common. In such systems, there are many dimensions to assess: security and reliability, safety and liveness, real-time response, etc. So far modelling and formalizing attempts have been very piecemeal approaches. This paper describes our analysis of a pervasive case study (MATCH, a homecare application) and our proposal for formal (particularly verification) approaches. Our goal is to see to what extent current state of the art formal methods are capable of coping with the verification demand introduced by pervasive systems, and to point out their limitations.

Keywords: pervasive systems, modelling, formalizing, verification

1 Introduction

Pervasive systems (often also termed *ubiquitous systems*) are increasingly common. But what are they? One of the many definitions is that

> *Pervasive Computing refers to a general class of mobile systems that can sense their physical environment, i.e., their context of use, and adapt their behaviour accordingly [PSHC08].*

While there are very many forms of pervasive system, they are often:

- mobile and autonomous;
- distributed and concurrent;
- interacting and interactive;

- reactive and potentially non-terminating; and
- composed of humans, agents and artifacts interacting together.

Existing pervasive systems provide services to the inhabitants of a home, the workers of an office or the drivers in a car park. We know that requirements for current and future pervasive systems involve great diversity in terms of their types of services, such as multimedia, communication or automation services.

Typical Example. *As we walk into a shopping area, our intelligent clothing interacts wirelessly with shops in the area and then with our mobile phone to let us know that our shoes are wearing out and that the best deals nearby are at shops 'X', 'Y' and 'Z'.*

Our PDA, which holds our shopping list, also interacts with our phone to suggest the optimum route to include shoes in our shopping.

At the same time, our PDA interacts with the shopping area's network and finds that one of our friends is also shopping – a text message is sent to the friend's mobile/PDA to coordinate shopping plans and schedule a meeting for coffee at a close location in 15 minutes.

Even in this simple example the components at least need capabilities to carry out:

- plan synchronisation;
- spatial reasoning and context-awareness;
- planning and scheduling;
- mobility and communication, etc.

The above is *an* example but there are *many* more, often more complex examples[1]. What is of concern to us is that pervasive computing is increasingly used in (safety, business, or mission) critical areas. This, of course, leads us to the potential use of formal methods in this area. Again, even considering the simple example above there are many dimensions to assess: security and reliability; safety and liveness; real-time response, etc.

Although there have been some attempts at modelling, e.g. [CFJ03, WZGP04, HI04, SB05, Sim07], formalising and even verifying aspects of pervasive and ubiquitous systems, e.g. [DKNP06], these have been very piecemeal approaches. We wish to be able to analyse the varied behaviours of a pervasive system from a number of viewpoints.

But how might this be achieved? This paper describes our analysis of a pervasive case study, MATCH [CM07], and our proposals for formal (particularly verification) approaches.

As we have seen, pervasive systems have some quite complex specification aspects. This explains, in part, why the verification of such systems is difficult. Essentially, most pervasive systems involve many dimensions that we must address (formalise and verify) simultaneously:

[1] See *Personal and Ubiquitous Computing* journal (Springer); *Pervasive and Mobile Computing* journal (Elsevier); *Journal of Ubiquitous Computing and Intelligence* (American Scientific Publishers); *International Journal of Ad Hoc and Ubiquitous Computing* (Inderscience Publishers); and *Journal of Ubiquitous Computing and Communication* (UBICC publishers).

- autonomous behaviour of agents;
- uncertainty in communications;
- teamwork, collaboration and coordination;
- organisations, norms, societal interactions;
- uncertainty in sensing;
- real-time aspects;
- *etc*...

In addition, such systems often involve humans *within* the system which directly affect the system's behaviour.

Current state of the art formal methods appear incapable of coping with the verification demand introduced by pervasive systems, primarily because reasoning about such systems requires considering quantitative, continuous and stochastic behaviour. We also require to prove interaction properties which are quite subtle to express.

2 MATCH overview

MATCH (Mobilising Advanced Technology for Care at Home) is a collaborative research project focused on technologies for care at home. The overall aim of MATCH is to develop a research base for advanced technologies in support of social and health care at home. The client users for MATCH will include older people and people with disabilities of all ages. The goal is to enable people to manage their health and way of life so that they can continue to live independently in their own homes for longer.

The main aim of the research in the MATCH project is to integrate a number of home-care technologies into one system that can be installed into a user's home. This system will provide support and assistance where needed by realising a number of goals. Goals may include "Warn the user if they are doing something dangerous", "Call for help if the user has an accident and needs assistance", "Inform a medical professional if the user's health deteriorates".

This could be implemented as a rule based, event driven pervasive system in which a set of input components such as: motion detectors, RFID readers, temperature sensors, heart monitors, and microphones, and enable the detection of significant events. A set of output components such as: speakers, GUIs, mobile devices, TVs, lights, and tactile devices, allow the system to interact with the environment. A set of rules will determine under what circumstances the system should take action and what action should be taken.

A significant factor in whether such a system would be adopted within a home would be the issue of trust. Therefore, it would be advantageous if properties of the MATCH system could be verified. This would then provide support to the claims made by the system developers. Properties of interest can be categorised as (but not limited to) Security, Safety and/or Usability.

2.1 Security

Security properties relate to the integrity of the system and its ability to withstand the efforts of malicious agents. As an example the property "Food delivery staff cannot see mental health records" is concerned with the user's confidentiality. It would be advantageous if the food delivery staff could have limited access to the homecare system. This could be used to find relevant information about the user, such as dietary requirements. However, it would not be reasonable for all of the user's confidential medical records to be divulged to anyone who interacts with the system. Another example, "Medical records are consistent across the system" relates to system integrity. Because of the distributed nature of a pervasive system, there maybe several different devices used to monitor and record the medical condition of the user. A number of these devices may hold similar or overlapping data. It should be the case that all such data is consistent. "Correspondence between the system view of events and the events taking place on portable devices" is another integrity property. It is unlikely that portable devices will have complete knowledge of the state of the system. However, they should act in a manner that corresponds to the current state of the system as a whole. "No unauthorised tracking" is a property which is concerned with confidentiality. It should not be possible for a malicious agent to use system outputs to allow them to track the user.

2.2 Safety

Safety properties relate to situations in which the system may cause harm, either by action or inaction. For example, "Sensors are never offline when a patient is in danger". The system should always have sufficient sensor coverage to detect all potentially dangerous situations that it can be reasonably expected to recognise. If the system does detect that a patient is in danger then appropriate action should be taken. The action taken should have an expected response time that is appropriate for the situation. For example, if the patient is having a heart attack then an ambulance should be called. Other forms of notification such as e-mail are unlikely to be received in time. This could be represented by the property "If a patient is in danger, assistance should arrive within a given time". Properties such as "No component will perform an action that it believes will endanger the patient" can relate the hardware devices used within the system. A more traditional property, "Urgent actions related to the patient's safety will always take precedence over all other actions" ensures that important actions are prioritised. It is also important to consider potentially damaging actions a user may attempt, "Users have no reasonable strategy for cooperating to falsify records or events". This property aims to prevent users from being able to manipulate the system so as to deceive a health professional.

2.3 Usability

Usability properties refer to aspects that directly affect the user's experience and interaction with the system. Example properties include, "Notifications occur only at appropriate intervals and in appropriate circumstances". If a user is bombarded with too many insignificant messages or is interrupted while busy, they are likely to find the system obtrusive and will probably reject it. Similarly "Requests to the users must be relevant to them" and "There should not be to many pending requests on a user/patient" also relate to the users' acceptance of the system.

3 Some sample properties

In this section we will recall the earlier sample properties of the MATCH system, and formulate them using formal languages. In particular, we consider formal languages for which a model checking or verification tool is available.

Below we consider an informal and formal account of the relevant properties.

3.1 Security

3.1.1 Formalisation

Food delivery staff cannot see mental health records. In order to formalise this property we could use the access control language RW of [GRS04]. In that system, we assume predicates such as

$$\begin{array}{lcl} fd(X) & : & X \text{ is a member of food delivery staff} \\ mhr(X,Y) & : & X \text{ is the mental health record of } Y \\ td(X,Y) & : & X \text{ is a treating doctor for } Y \\ ha(X) & : & X \text{ is a health administrator} \end{array}$$

Predicates *read* and *write* indicate read and write permissions respectively. The following formulae show how such permissions are granted:

$$\begin{array}{lcl} read(mhr(X,Y),Z) & \Leftrightarrow & Y = Z \ \vee \ td(Z,Y) \\ write(td(X,Y),Z) & \Leftrightarrow & Z = Y \ \vee \ ha(Z) \\ & \vdots & \end{array}$$

RW can calculate whether a user can manipulate the access control system in order to give himself the necessary permissions to achieve a goal. In this example, a member of food delivery staff might be able to read mental health records if he can promote himself to treating doctor. We can verify that this is not possible by evaluating the query

$$\forall a,b \in Agent.\ r \in Record.\ fd(a) \ \rightarrow \ \neg[a : mhr(r,b)]$$

This query evaluates whether there is a (perhaps roundabout) sequence of reads and writes resulting in a food delivery staff member a being able to read the health records of some patient b.

Correspondence between the system view of events and the events taking place on portable devices. This property is known as injective agreement in Lowe's hierarchy of authentication [Low97]. Intuitively, this property states that each time the MATCH system stores some value v in the record of a patient p, then p's doctor d has submitted this value v. In the same way, each time doctor d submits some value v for p's record to the system, then the MATCH system should update its state accordingly. This can be expressed in ProVerif's query language [Bla01] as follows:

$$\begin{array}{l} query\ evinj:\ ev_{syst}(d,p,v) \ ==> \ evinj:\ ev_{patient}(d,p,v) \\ query\ evinj:\ ev_{patient}(d,p,v) \ ==> \ evinj:\ ev_{syst}(d,p,v) \end{array}$$

The first (*resp.* second) query is true when, for each executed event $ev_{syst}(d,p,v)$ (*resp.* $ev_{patient}(d,p,v)$), there exists a distinct executed event $ev_{patient}(d,p,v)$ (*resp.* $ev_{syst}(d,p,v)$); and $ev_{syst}(d,p,v)$ is executed before $ev_{patient}(d,p,v)$ (*resp.* $ev_{patient}(d,p,v)$ is executed before $ev_{syst}(d,p,v)$).

***Non-Authorised tracking* (e.g. *by strangers outside the house*).** MATCH's implementation will rely on Radio Frequency IDentification (RFID) tags attached to patients, in order for the system to be able to detect if a particular patient is in the house. However, one wouldn't want someone outside the house with an RF reader to be able to determine patient's position. In the RFID literature [GJP05, Jue06, WSRE03], this security requirement is known as untraceability. Intuitively, this property states that an attacker cannot link two different identifications to the system of the same tag (and thus of the same patient). In other words, all tags that identify themselves to the system look different to an outsider.

This property can be specified in the applied pi calculus formalism. The applied pi-calculus [AF01] is a language for describing concurrent processes and their interactions. It is based on the pi-calculus, but adds equations which make it possible to model a range of cryptographic primitives.

We consider RFID protocols that can be expressed in the applied pi calculus as a closed plain process P

$$P \equiv \nu\tilde{n}.\ (DB \mid !R \mid !T)$$

where

$$T \equiv \nu\tilde{m}.\ init.\ !main$$

for some processes *init* and *main*[2]. Intuitively, T is the process modeling one tag, and having T under a replication in P corresponds to considering a system with an unbounded number of tags. Each tag initialises itself (this includes registering at the database DB and is modelled by *init* in T) and then may execute itself an unbounded number of times. Thus *main* models one session of the tag's protocol. R corresponds to one session of the reader's protocol, and DB to the database. We consider an unbounded number of readers, thus R is under a replication in P.

Many properties of security protocols (including untraceability) are formalised in terms of *observational equivalence* ($\approx$) between two processes. Intuitively, processes which are *observationally equivalent* cannot be distinguished by an outside observer, no matter what sort of test he makes. This is formalised by saying that the processes are indistinguishable under any context, *i.e.* no matter in what environment they are executed.

Let P be an RFID protocol as defined above, P is said to satisfy untraceability if $P \approx P'$ where

$$P' \equiv \nu\tilde{n}.\ (DB \mid !R \mid !T')$$

and

$$T' \equiv \nu\tilde{m}.\ init.\ main$$

The intuitive idea behind this definition is as follows: each session of P should look to the intruder as initiated by a different tag. In other words, an *ideal* version of the protocol, *w.r.t.* untraceability, would allow tags to execute themselves at most once. The intruder should then not be able to tell the difference between the protocol P and the ideal version of P.

[2] $\nu\tilde{n}$ models a sequence of names $n_1,\ldots,n_k$ restricted to $(Db \mid !R \mid !T)$. In the same way $\nu\tilde{m}$ denotes a sequence of names $m_1,\ldots,m_\ell$ restricted to $(init.\ !main)$.

3.1.2 Verification - at present

Several tools are available for verifying the properties defined above. Considering the first property, RW is supported by the AcPeg tool [Zha06] which accepts descriptions of access control models and evaluates queries. It treats the case that the sets of agents and other resources are finite (this case is decidable). It is currently not able to deal with non-bounded sets of resources, a problem known to be undecidable [HRU76].

DynPAL [Bec09] also analyses access control systems of a dynamic nature, similar to RW, and is more able to treat unbounded systems, but does not handle queries about "read" capabilities.

We expressed the second property in ProVerif's query language and the third using observational equivalence. This, in some cases, allows a user to automatically check that a protocol satisfies these requirements using the tool ProVerif [Bla01]. However, the verification of these properties is an undecidable problem [DLMS99], and ProVerif isn't always able to give an answer. It may even introduce false attacks.

For the second property, which is a correspondence property one could use other tools like Avispa [ABB+05] or Casper [Low98]. In order for these tools to be able to give some results, only bounded systems are considered. Indeed, when the number of executions of a protocol is bounded, the verification of many correspondence properties becomes decidable. To the best of our knowledge, ProVerif is the only tool able in some cases to prove that processes are observationally equivalent.

3.2 Safety

3.2.1 Formalisation

Sensors are never offline when a patient is in danger. We can, for example, formulate such a statement using standard linear-time temporal logic (LTL) [Eme90, MP92, Fis07]. We first capture the notions of sensors being "offline" and patients being "in danger".

Assume s_i is a sensor ($i \in \{1,..,m\}$). We define propositions[3] $\mathit{fail}(s_i)$, $\mathit{switch\_off}(s_i)$ and $\mathit{offline}(s_i)$ as follows: $\mathit{fail}(s_i)$ denotes that the sensor s_i fails; $\mathit{switch\_off}(s_i)$ denotes that s_i is switched off; and $\mathit{offline}(s_i)$ denotes that s_i is offline. Now, we assume that if either the sensor fails or it has been switched off, then it is offline

$$\Box[(\mathit{fail}(s_i) \vee \mathit{switch\_off}(s_i)) \Rightarrow \mathit{offline}(s_i)], \quad \forall i \in \{1,..,m\}$$

future time points", '$\Diamond$' means "at *some* present or future time point", and '$\bigcirc$' means "at the *next* time point".) We now consider the cases where patients are in danger. Assume p_j is a patient ($j \in \{1,..,n\}$). We define the following propositions: $\mathit{in\_danger}(p_j)$ denotes that the patient p_j is in danger; $\mathit{high\_heart\_rate}(p_j)$ denotes that the heart rate monitor of p_j has detected p_j's heart rate to be higher than normal; $\mathit{low\_activity}(p_j)$ denotes that the activity monitors of the patient p_j have detected that p_j has moved very little for a period of time; $\mathit{motionless}(p_j)$ denotes that sensors have detected that p_j has not moved at all for a period of time. If we assume that a patient

[3] While we use a fragment of first-order language, the finiteness of the domain in question ensures that this is essentially *propositional* temporal logic.

is in danger only if either his/her heart rate is higher than normal, he/she has moved very little with a period of time, or he/she has been inactive (but not in bed) for a while, then

$$\Box[(high\_heart\_rate(p_j) \vee low\_activity(p_j) \vee motionless(p_j)) \Rightarrow in\_danger(p_j)]$$

The truth of predicates such as '*low_activity*' can also be defined in terms of the values of sensors, together with some real-time and probabilistic constraints.

We can now specify the property "if a patient is in danger sensors never go offline until the patient is no longer in danger" as follows:

$$\Box[in\_danger(p_j) \Rightarrow (\neg \bigvee_{i\in\{1,..,m\}} offline(s_i)) \mathscr{U} (\neg in\_danger(p_j))], \quad \forall j \in \{1,..,n\}$$

Of course, this simple version assumes that the dangerous situation will eventually be resolved.

Let us now consider the second property.

Urgent actions related to the patients' safety will always take place before other actions Assume A is a set of *urgent actions*, and B is a set of *non-urgent* actions such that $A \cap B = \emptyset$ (again, this notion of urgency might well be defined in terms of some priority measure.) We use the proposition $action(a)$ to denote that the action 'a' takes place. Property (ii) states that if a patient is in danger, a non-urgent action should not take place before an urgent action. This can again be expressed in LTL as follows:

$$\Box[in\_danger(p_j) \Rightarrow (\neg(\bigvee_{b\in B} action(b) \ \mathscr{U} \bigvee_{a\in A} action(a)) \mathscr{U} \neg in\_danger(p_j)], \quad \forall j \in \{1,..,n\}$$

This, of course, can be made more detailed and complex, for example if we delve into the properties of actions.

If a patient is in danger, assistance should arrive within a given time We define the proposition $assistance(p_j)$ as denoting that assistance arrives for p_j. If the specification were "*if* a patient is in danger, assistance should arrive eventually", then we could formulate this in LTL as follows:

$$\Box[in\_danger(p_j) \Rightarrow \Diamond assistance(p_j)], \quad \forall j \in \{1,..,n\}.$$

assistance must be available. Since this is a real-time property, we must extend the logic to capture this specification. Alternatively, we might use a logic such as TCTL [ACD90]. TCTL is a branching-time temporal logic, specifically a real-time extension of the logic CTL [CE82], which can express real-time properties. Thus, (ii) can be expressed in TCTL as follows:

$$\forall\Box[in\_danger(p_j) \Rightarrow \forall\Diamond_{\leq t} assistance(p_j)], \quad \forall j \in \{1,..,n\}$$

This formula states that if a patient is in danger, it is guaranteed that some assistance will arrive within time t. (Note that '$\forall$' here refers to *all* possible paths through the branching futures.)

If a patient is in danger, assistance should arrive within a given time with a probability of 95% This property has both real-time and probabilistic aspects. Therefore, TCTL can no longer be used. In order to express this specification we can use the logic PCTL [HJ94], which can express quantitative bounds on the probability of system evolutions. Thus, (iv) can be specified in PCTL as follows:

$$P_{\geq 0.95}[in\_danger(p_j)\ \mathscr{U}^{\leq t}\ assistance(p_j)], \quad \forall j \in \{1,..,n\}$$

No component will take an action that it believes will endanger the patient The property (v) now includes a situation where each component's beliefs must be taken into account. This suggests the use of a modal logic which uses possible worlds to capture the beliefs (or knowledge) that the component/agent has. Temporal Logics of Knowledge or Belief[FHMV96, HMV04] can be used to express the property (v) as they combine the temporal aspects with a modal logic of knowledge (or belief). Assume c_i ($i \in \{1,..,C\}$) is a component agent, and A is the set of all urgent and non-urgent actions. (v) might be expressed in a temporal logic of belief as:

$$\Box K_{c_i}[\neg(\bigvee_{a\in A} action(a) \Rightarrow \Diamond \bigvee_{j\in\{1,..,n\}} in\_danger(p_j))], \quad \forall i \in \{1,..,C\}$$

3.2.2 Verification Approaches

There are various verification and model checking tools for the logics we introduced in the previous section. The properties *(i)-(v)* can be verified using a suitable model checker or verification tool described below.

For the logic LTL some well-known tools are NuSMV [CCGR99], SPIN [Hol03], VIS [Gro96], TRP++ [HK03]. NuSMV is an extension of the model checking tool SMV [McM93], which is a software tool for the formal verification of finite state systems. Unlike SMV, NuSMV provides facility for LTL model checking. Spin is an LTL model checking system, supporting all correctness requirements expressible in LTL, but it can also be used as an efficient on-the-fly verifier for more basic safety and liveness properties. VIS is a symbolic model checker supporting LTL. VIS is able to synthesise finite state systems and/or verify properties of such systems, which have been specified hierarchically as a collection of interacting finite state machines. TRP++ is a resolution based theorem prover for LTL. TRP++ is based on resolution method for LTL [FDP01].

The best-known tools for the logic TCTL are UPPAAL [BLL+95] and KRONOS [BDM+98]. UPPAAL is an integrated tool environment for modelling, validation and verification of real-time systems modelled as networks of timed automata, extended with data types. The tool can be used for the automatic verification of safety and bounded liveness properties of real-time systems. KRONOS is a tool developed with the aim to verify complex real-time systems. In KRONOS, components of real-time systems are modelled by timed automata and the correctness requirements are expressed in the real-time temporal logic TCTL.

PRISM [HKNP06] and APMC [HLMP04] are tools which can be used to model check PCTL formulae. PRISM is a probabilistic model checker, a tool for formal modelling and analysis of systems which exhibit random or probabilistic behaviour. It supports three types of probabilistic models: discrete-time Markov chains (DTMCs), continuous-time Markov chains (CTMCs) and Markov decision processes (MDPs), plus extensions of these models with costs and rewards. The

property specification language is mainly PCTL; however, the tool also supports the temporal logics CSL and LTL. The "Approximate Probabilistic Model Checker" (APMC) [HLMP04] is an approximate distributed model checker for fully probabilistic systems. APMC uses a randomised algorithm to approximate the probability that a temporal formula is true, by using sampling of execution paths of the system.

Finally, there are tools for tackling the verification of specifications in combined modal and temporal logics, such as temporal logics of belief or temporal logics of knowledge. Although less well developed that some of the tools above, these range from deductive approaches [JHS+96, DFB02] to model checking for such logics [KLP04, GM04, BDFF08].

4 Discussion

4.1 Limitations

In Section 3, we give some example properties of different dimensions, e.g. security and safety, and describe various tools that can be used in the verification and model checking of these properties. There are several limitations of these tools and techniques.

As discussed in Section 3.2.1 and Section 3.2.2, different formal frameworks and tools are used to specify and verify different safety properties. For example, we should consider different temporal logics and model checking tools for real-time aspects, probabilistic aspects, belief aspects, etc. Unfortunately, there is no standard methods which can be used for all aspects we are interested in. This makes it difficult to use a certain formal framework for all safety properties. Another limitation is that each tool requires a different presentation of the model of the system.

As we discussed in Section 3.1.2, there exist several tools that allow us to verify systems *w.r.t.* security. However, some security properties of interest in pervasive systems are not supported well or even at all by these tools. Indeed, while ProVerif (to name only one) is very efficient for verifying correspondence and reachability properties like injective agreement, it seems to reach its limits when used for verifying observational equivalence, and thus properties like untraceability. This is due to the fact that ProVerif considers a stronger relation than observational equivalence introducing in practise many false attacks.

All the existing tools for verifying systems *w.r.t.* security abstract away from real time, focusing only on the sequencing of events. Although this has many advantages, it is a serious limitation for reasoning about protocols such as distance bounding protocols, which rely on real time considerations. These protocols are often implemented in RFID-based systems, and thus in many pervasive applications, in order to prevent relay attacks. Modelling time will thus be necessary for reasoning about pervasive systems.

In the same way, some features of many protocols designed to achieve the above mentioned requirements are not efficiently handled by existing tools. In particular, ProVerif is also inefficient for systems with local non-monotonic mutable states. But, protocols which aim to enforce untraceability often rely on such states. More precisely, ProVerif handles all states as monotonic introducing again false attacks.

As far as tools for verifying access control systems are concerned, the biggest limitation would be, as we already mentioned in section 3.1.2 that they usually cannot deal with unbounded access control models. Moreover, it is difficult to express and model integrity constraints in these tools.

Finally, we have introduced three existing theories for formally specifying security properties, namely RW, ProVerif's query language, and observational equivalence. It is important to note that they involve different levels of abstraction. Indeed, one can reason at the access control policy level, or at some symbolic model for protocols level, or even at the implementation level. It will be very important, to link these levels in order to transfer results from more abstract levels to more concrete ones.

4.2 Our Approach

As mentioned, a pervasive system might have quite complex specification aspects. Most pervasive systems involve many dimensions that must be formalised and verified simultaneously. Some of the dimensions that are common in pervasive systems are *real-time aspects*, *uncertainty in sensing and communication*, *teamwork, collaboration and coordination*, *organizations, norms and social interactions*, *autonomous behaviour of agents*, *etc.*

Current state of the art of formal methods appears incapable of coping with the verification demand introduced by pervasive systems, because reasoning about such systems requires combinations of multiple dimensions such as quantitative, continuous and stochastic behaviour to be considered, and requires proving properties which are quite subtle to express. For example, [Zah09, KZF09] show that some simple properties of a typical pervasive system can be verified using a single verification tool; but a single verification approach cannot be used in verification of more complex properties involving different dimensions.

Generally speaking, while the formal description of pervasive systems is essentially multi-dimensional, we generally do not have verification tools for all the appropriate combinations. It is very clear that developing a framework covering all these dimensions is almost impossible, or verification over very complex frameworks is very challenging.

In order to tackle the challenge of pervasive system verification, we aim to combine the power of established verification techniques, notably *model checking*, *deduction*, *abstraction*, etc. In particular, we are currently working on a generic framework for the model-checking of combined logics, including logics of knowledge, logics of context, real-time temporal logics, probabilistic temporal logics, etc. This work is based on the work of [FMD04], where a framework is given for the model-checking of combined modal temporal logics. This framework does not capture complex logics, which are quite essential in specifying different dimensions of pervasive systems. We therefore will extend this approach to provide a coherent framework for the formal analysis of pervasive systems. We will then apply the new technique to the verification of combined properties of a sample pervasive system, e.g. MATCH.

5 Conclusion

This paper describes our analysis of a pervasive case study, MATCH. As an initial step, we formally specify some safety and security properties of the MATCH system. We discuss to what extent current state of the art formal methods are capable of coping with the verification demand introduced by pervasive systems, and we point out their limitations. We also give an account of our proposal for formal verification of pervasive systems.

Bibliography

[ABB+05] A. Armando, D. A. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cullar, P. H. Drielsma, P.-C. Ham, O. Kouchnarenko, J. Mantovani, S. Mdersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Vigan, L. Vigneron. The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications. In Etessami and Rajamani (eds.), *CAV*. Lecture Notes in Computer Science 3576, pp. 281–285. Springer, 2005.

[ACD90] R. Alur, C. Courcoubetis, D. Dill. Model-Checking for Real-Time Systems. In *Proc. Logic in Computer Science (LICS)*. Pp. 414–425. 1990.

[AF01] M. Abadi, C. Fournet. Mobile values, new names, and secure communication. *SIGPLAN Not.* 36(3):104–115, 2001.

[BDFF08] R. H. Bordini, L. A. Dennis, B. Farwer, M. Fisher. Automated Verification of Multi-Agent Programs. In *Proc. 23rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. Pp. 69–78. 2008.

[BDM+98] M. Bozga, C. Daws, O. Maler, A. Olivero, StavrosTripakis, S. Yovine. Kronos: A Model-Checking Tool for Real-Time Systems. In *CAV '98: Proceedings of the 10th International Conference on Computer Aided Verification*. Pp. 546–550. Springer Verlag, 1998.

[Bec09] M. Y. Becker. Specification and Analysis of Dynamic Authorisation Policies. In *CSF '09: Proceedings of the 2009 22nd IEEE Computer Security Foundations Symposium*. Pp. 203–217. IEEE Computer Society, 2009.

[Bla01] B. Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *CSFW '01: Proceedings of the 14th IEEE workshop on Computer Security Foundations*. P. 82. IEEE Computer Society, Washington, DC, USA, 2001.

[BLL+95] J. Bengtsson, K. G. Larsen, F. Larsson, P. Pettersson, W. Yi. UPPAAL — a Tool Suite for Automatic Verification of Real–Time Systems. In *Proceedings of Workshop on Verification and Control of Hybrid Systems III*. Lecture Notes in Computer Science 1066, pp. 232–243. Springer Verlag, 1995.

[CCGR99] A. Cimatti, E. Clarke, F. Giunchiglia, M. Roveri. NuSMV: A New Symbolic Model Verifier. In *Proceedings of International Conference on Computer-Aided Verification (CAV'99)*. Pp. 495–499. 1999.

[CE82] E. M. Clarke, E. A. Emerson. Using Branching Time Temporal Logic to Synthesise Synchronisation Skeletons. *Science of Computer Programming* 2:241–266, 1982.

[CFJ03] H. Chen, T. Finin, A. Joshi. An ontology for context-aware pervasive computing environments. *Knowl. Eng. Rev.* 18(3):197–207, 2003.

[CM07] J. S. Clark, M. R. McGee-Lennon. MATCH: Mobilising Advanced Technologies for Care at Home. 2007. Poster at *Delivering Healthcare for the 21st Century*, Glasgow.

[DFB02] C. Dixon, M. Fisher, A. Bolotov. Resolution in a Logic of Rational Agency. *Artificial Intelligence* 139(1):47–89, July 2002.

[DKNP06] M. Duflot, M. Z. Kwiatkowska, G. Norman, D. Parker. A Formal Analysis of Bluetooth Device Discovery. *STTT* 8(6):621–632, 2006.

[DLMS99] N. A. Durgin, P. D. Lincoln, J. C. Mitchell, A. Scedrov. Undecidability of bounded security protocols. In *Proceedings of the Workshop on Formal Methods and Security Protocols-FMSP*. 1999.

[Eme90] E. A. Emerson. Temporal and Modal Logic. In Leeuwen (ed.), *Handbook of Theoretical Computer Science*. Pp. 996–1072. Elsevier, 1990.

[FDP01] M. Fisher, C. Dixon, M. Peim. Clausal Temporal Resolution. *ACM Transactions on Computational Logic* 2(1):12–56, Jan. 2001.

[FHMV96] R. Fagin, J. Halpern, Y. Moses, M. Vardi. *Reasoning About Knowledge*. MIT Press, 1996.

[Fis07] M. Fisher. Temporal Representation and Reasoning. In van Harmelen et al. (eds.), *Handbook of Knowledge Representation*. Elsevier Press, 2007.

[FMD04] M. Franceschet, A. Montanari, M. De Rijke. Model Checking for Combined Logics with an Application to Mobile Systems. *Automated Software Engg.* 11(3):289–321, 2004.

[GJP05] S. L. Garfinkel, A. Juels, R. Pappu. RFID Privacy: An Overview of Problems and Proposed Solutions. *IEEE Security and Privacy* 3(3):34–43, 2005.

[GM04] P. Gammie, R. van der Meyden. MCK: Model Checking the Logic of Knowledge. In *Proc. 16th International Conference on Computer Aided Verification (CAV)*. Lecture Notes in Computer Science 3114, pp. 479–483. Springer, 2004.

[Gro96] T. V. Group. VIS: A System for Verification and Synthesis. In *Proceedings of the 8th International Conference on Computer Aided Verification*. Pp. 428–432. 1996.

[GRS04] D. P. Guelev, M. Ryan, P. Y. Schobbens. Model-checking Access Control Policies. 3225:219–230, 2004.

[HI04] K. Henricksen, J. Indulska. A Software Engineering Framework for Context-aware Pervasive Computing. In *Proceedings 2nd IEEE Conf. on Pervasive Computing and Communications*. Pp. 77–86. 2004.

[HJ94] H. Hansson, B. Jonsson. A Logic for Reasoning about Time and Reliability. *Formal Aspects of Computing* 6:102–111, 1994.

[HK03] U. Hustadt, B. Konev. TRP++ 2.0: A Temporal Resolution Prover. In *Proceedings of Conference on Automated Deduction (CADE-19)*. Pp. 274–278. 2003.

[HKNP06] A. Hinton, M. Kwiatkowska, G. Norman, D. Parker. PRISM: A Tool for Automatic Verification of Probabilistic Systems. In *Proc. 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06)*. Lecture Notes in Computer Science 3920, pp. 441–444. Springer, 2006.

[HLMP04] T. Hérault, R. Lassaigne, F. Magniette, S. Peyronnet. Approximate Probabilistic Model Checking. In *Proc. 5th International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI'04)*. Lecture Notes in Computer Science 2937, pp. 307–329. Springer, 2004.

[HMV04] J. Y. Halpern, R. van der Meyden, M. Y. Vardi. Complete Axiomatizations for Reasoning about Knowledge and Time. *SIAM J. Comput.* 33(3):674–703, 2004.

[Hol03] G. J. Holzmann. *The Spin Model Checker*. Addison-Wesley, 2003.

[HRU76] M. A. Harrison, W. L. Ruzzo, J. D. Ullman. Protection in Operating Systems. *Commun. ACM* 19(8):461–471, 1976.

[JHS+96] G. Jaeger, A. Heuerding, S. Schwendimann, F. Achermann, P. Balsiger, P. Brambilla, H. Zimmermann, M. Bianchi, K. Guggisberg, W. Heinle. LWB–The Logics Workbench 1.0. http://lwbwww.unibe.ch:8080/LWBinfo.html, 1996. University of Berne, Switzerland.

[Jue06] A. Juels. RFID security and privacy: a research survey. *IEEE Journal on Selected Areas in Communications* 24(2):381–394, 2006.

[KLP04] M. Kacprzak, A. Lomuscio, W. Penczek. From Bounded to Unbounded Model Checking for Temporal Epistemic Logic. *Fundam. Inform.* 63(2-3):221–240, 2004.

[KZF09] S. Konur, A. A. Zahrani, M. Fisher. Verification of a Message Forwarding System using PRISM. In *PreProc. of Ninth International Workshop on Automated Verification of Critical Systems*. Pp. 237–239. Technical Report, University of Swansea, 2009.

[Low97] G. Lowe. A Hierarchy of Authentication Specifications. In *CSFW '97: Proceedings of the 10th IEEE Workshop on Computer Security Foundations*. Pp. 31–43. IEEE Computer Society, Washington, DC, USA, 1997.

[Low98] G. Lowe. Casper: A Compiler for the Analysis of Security Protocols. *Journal of Computer Security* 6(1-2):53–84, 1998.

[McM93] K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishing, 1993.

[MP92] Z. Manna, A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag, New York, 1992.

[PSHC08] E. K. Paik, M.-K. Shin, J. Hwang, J. Choi. Design Goals and General Requirements for Future Network. N13490, Korea Technology Center, 2008.

[SB05] Q. Z. Sheng, B. Benatallah. ContextUML: A UML-Based Modeling Language for Model-Driven Development of Context-Aware Web Services Development. Volume 0, pp. 206–212. IEEE Computer Society, 2005.

[Sim07] C. Simons. CMP: A UML Context Modeling Profile for Mobile Distributed Systems. Volume 0, p. 289b. IEEE Computer Society, 2007.

[WSRE03] S. A. Weis, S. E. Sarma, R. L. Rivest, D. W. Engels. Security and Privacy Aspects of Low-Cost Radio. In *Hutter, D., Müller, G., Stephan, W., Ullman, M., eds.: International Conference on Security in Pervasive Computing - SPC 2003, volume 2802 of LNCS, Boppard, Germany*. Pp. 454–469. Springer-Verlag, march 2003.

[WZGP04] X. H. Wang, D. Q. Zhang, T. Gu, H. K. Pung. Ontology-based Context Modeling and Reasoning Using Owl. In *Context Modeling and Reasoning Workshop at PerCom 04*. Pp. 18–22. 2004.

[Zah09] A. A. Zahrani. Formal Analysis of a Message Forwarding System using PRISM. Master's thesis, Department of Computer Science, University of Liverpool, 2009.

[Zha06] N. Zhang. AcPeg, The Access Control Policy Evaluator and Generator. July 2006. The tool can be obtained from www.cs.bham.ac.uk/ nxz or www.cs.bham.ac.uk/ mdr/research/projects/05-AccessControl.