

Verifying Interoperability Requirements in Pervasive Systems

A research project at Birmingham, Liverpool and Glasgow

Abstract

This document arises from the proposal submitted for funding this project. We have extracted the parts relevant for the applicants to the PhD position in Birmingham. In particular, the candidates can refer to WP9 in page 8. In this document, the applicants will find an introduction to the project (its context and its aims), some brief explanations about the considered case studies, and a detailed research programme (section D). With regard to this research programme, candidates are invited to write up to two pages on their understanding of, and their possible contributions to the project.

A. Introduction

The success of pervasive computing depends crucially on the ability to build, maintain and augment *interoperable systems*: components from different manufacturers built at different times that are required to interact to achieve the user's overall goals. The aim of the project is to develop viable and appropriate techniques for verifying interoperability requirements in pervasive systems. This work will build upon existing techniques and expertise in the areas of formal modelling and reasoning, and will be motivated, validated and challenged by case study scenarios from pervasive systems, which will run throughout the project. Our approach to verifying these properties is to identify interoperability requirements for the interaction between the devices and their environment. These requirements also introduce an important layer of abstraction allowing modularity in the verification process: it suffices to show that each mobile device or fixed component meets the interoperability requirements, and that the interoperability requirements entail the desired high-level properties. To give a flavour of the sort of questions we have in mind, consider these examples:

1. **service and device interaction** – *will my cardio-care service (heart monitoring, drug delivery) still work while my PDA is getting data updates? Can granny's activity sensors also be used to detect burglars?*
2. **security and privacy** – *are the low-level access control configurations adequate to ensure management-level authorisation policies? Are visitors properly prevented from accessing confidential information on our wireless network? What are the consequences for my treatment plan of restricting data to only my doctor, excluding the consultant in London?*
3. **performance** – *will I get adequate video throughput when on the train? Will voice data be prioritised? How reliable is an remote assessment of a patient in a homecare setting when 30% sensors are down or known to lose data?*

We don't expect to answer all these questions about real systems by the end of the project, but we will be able to tackle some of them. Our focus on interoperability requirements makes it possible to adapt and extend techniques (such as model checking and process calculi) which have traditionally been used for verifying properties of small homogeneous systems, to large heterogeneous systems. To support this thesis, we will develop techniques to verify properties concerning important aspects of heterogeneous systems' *security*, *individual* and *collective* behaviour, *performance* and *privacy*. We will use the formal techniques to verify the consequent interoperability requirements, and evaluate their effectiveness through case studies.

Our focus is on the verification of *designs*; in particular we focus on the design of basic component behaviours and the protocols which dictate access to them and interaction between them. It is important to note our intention is not to develop pervasive computing systems as such, but rather to draw motivation from, and test our ideas in, a number of planned and existing systems.

The project brings together qualitative techniques, including deductive methods, model checking, and abstraction methods, with quantitative techniques, including probabilistic and performance analysis, in order to tackle the problem of verifying pervasive systems. We will be investigating problems which are both new and challenging (hence new techniques and methods will be required), but are still sufficiently close to existing work that our established work provides a solid foundation for solving them. The outcomes will directly benefit system designers, and indirectly, end users.

B. Background

Current state of the art formal methods appear 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 (such as the questions 1–3 above). In order to tackle this challenge, the project aims to leverage the power of established techniques, notably

model checking: a logic-based approach to analysing properties of state-based systems. There has been work (some of which was carried out by the investigators) on extensions such as *parametrised model checking*, *infinite state model checking* and *probabilistic model checking*, and this will be developed further within the project.

using deduction and abstraction: two closely linked, approaches that can be used either to reduce the verification problem to a scale suitable for model checking, or to tackle the larger problem directly.

process calculi: high-level descriptions of interaction, communication and synchronisation.

Part of our effort will involve pushing each technique further, but the majority of it will be on *pushing the combination*, i.e. bending and synthesising the techniques to make them give meaningful results in our case studies.

C. Case studies

Several case studies are planned; two of which are with external collaborators. We will draw inspiration from these case studies motivating our techniques as we develop them. We will then test our ideas on these case studies. From such an application-driven research we will gain 'real life' experience of applying our techniques in the field.

The case studies will be selected to exhibit the challenges in verification of pervasive systems in our chosen themes including verification of security protocols (RFID and TPM case studies) and access control systems (MATCH and Scatterbox case studies), or analysis of configuration change impact (MATCH and Scatterbox case studies).

The case studies will be drawn from three layers typical within pervasive systems: application (MATCH and Scatterbox case studies), infrastructure and network (TPM and RFID case studies).

The TPM The Trusted Platform Module (TPM) [?] is a hardware chip designed to enable commodity computers to achieve greater levels of security than was previously possible. There are 100 million TPMs currently in existence [?], mostly in high-end laptops made by HP, Dell, Sony, Lenovo, Toshiba, and others. The TPM stores cryptographic keys and other sensitive data in its shielded memory, and provides ways for platform software to use those keys to achieve security goals. Application software such as Microsoft's BitLocker and HP's ProtectTools use the TPM in order to guarantee security properties.

TPMs are manufactured by chip producers, including Atmel, Broadcom, Infineon, Sinosun, STMicroelectronics, and Winbond. It is specified by the Trusted Computing Group (TCG) industry consortium, that includes Intel, HP, Microsoft, AMD, IBM, Sun, Lenovo, and about 130 other members, in three documents [?] totalling about 800 pages. This introduction is intended to give a flavour and overview of its operation, and is specifically about TPM version 1.2. Some details are inevitably missing and some issues may be oversimplified here. The TPM specification is, of course, the authoritative source and overrides anything written here.

The TPM offers three kinds of functionality:

- **Secure storage.** User processes can store content that is encrypted by keys only available to the TPM. To store data using a TPM, one creates TPM keys and uses them to encrypt the data. TPM keys are arranged in a tree structure. To each TPM key is associated a 160-bit string called *authdata*, which is analogous to a password that authorises use of the key.
- **Platform measurement and reporting.** A platform can create reports of its integrity and configuration state that can be relied on by a remote verifier. The TPM contains a number of 160-bit registers called *platform*

configuration registers (PCRs) intended to enable a relying party to obtain unforgeable information about the platform state. A component can “measure” another component (compute its hash) and insert that measurement into a PCR. A TPM signing key can be used to sign the values of the PCRs. In this way, application software can send assurance about the state of the platform to a third party. Additionally, PCR values can be used to ensure that certain data is accessible only to authorised software.

- Platform authentication. A platform can obtain keys by which it can authenticate itself reliably. For platform authentication, one may create signing keys known as *application identity keys* (AIKs). These may be used to sign (appropriately tagged) application-specific data, and to sign PCR values. For such signatures to be useful, AIKs need to be certified as belonging to a TPM.

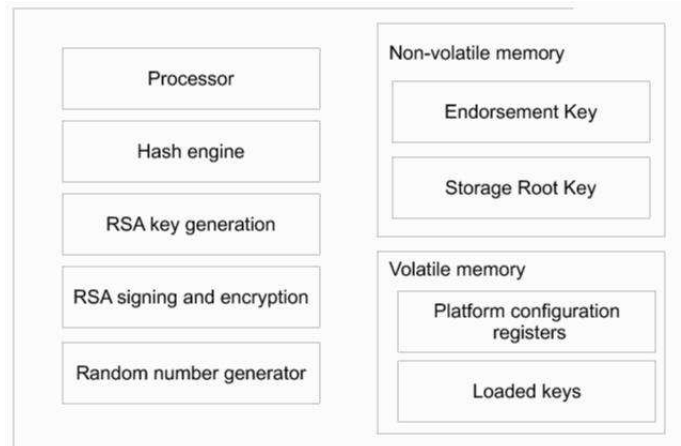


Figure 1: TPM architecture

RFID Infrastructure. Radio frequency (RF) technology is used in many applications for automated identification of objects or people. A RF system consists of two main components: RF chips and RF readers. RF chips are small microchips supporting wireless data transmission. Data is stored (remotely or not) in the RF chip and can remotely be retrieved by a RF reader. RF chips can be incorporated into products, animals, or persons for the purpose of identification and tracking using radio waves. RF readers query these chips for some identifying information about the objects to which chips are attached. Current and emerging applications using this technology include amongst others electronic toll collection, documents such as electronic passports and visas, and RF passes for public transportation. We now detail two of these application of RFID technology.

Electronic toll collection (ETC) consists in determining whether a given car driving through a given toll gate is enrolled in the programme. If yes, then it electronically debits the account of the car’s owner without requiring him to stop. If not the car owner should be stopped at the gate. Current ETC systems rely on RF technology: a RF reader placed at the toll gate communicates with a RF chip on the vehicle in order to identify it. A car should cross a toll gate if and only if it is enrolled in the program and its owner account has been debited with the corresponding amount of money.

An electronic passport is a combined paper and electronic identity document used to verify the identity of travelers. The passport’s critical information is stored on a RF chip embedded within its booklet. A typical session (*i.e.* interaction) between a reader and the passport’s chip is described below:

1. The reader optically acquires some information (passport number, name of the holder amongst others) from the passport;
2. The reader then needs to go through basic access control (BAC), *i.e.* to prove to the passport’s chip that it knows the information it should have acquired at the previous step. This is a challenge-response protocol aiming at remotely establishing some session keys between the reader and the passport’s chip.
3. If the previous step succeeds then the reader can remotely access information on the RF chip. All data exchanged at this phase is encrypted with the keys established during BAC.

This technology presents verification challenges in our chosen themes including verifying protocols that provide seamless communication properties for mobile devices in connected/disconnected states, implementations of and extensions to gossip protocols, APIs for communication with mobile devices, and protocols for persistent and secure data caching.

Context-Aware Message Management The *Scatterbox* system [?, ?] has been designed to serve as a test bed for context-aware computing in a pervasive computing environment. It provides a content-filtering service to its users by forwarding relevant messages to their mobile phone. The user’s context is derived by tracking his/her location and monitoring his/her daily schedule. These context data are accessed through Construct, and situations are identified that indicate the user’s interruptibility. As messages arrive, Scatterbox forwards them to subscribed users provided the user’s available context suggests they are in a situation where they may be interrupted. The system architecture is shown in Figure ??.

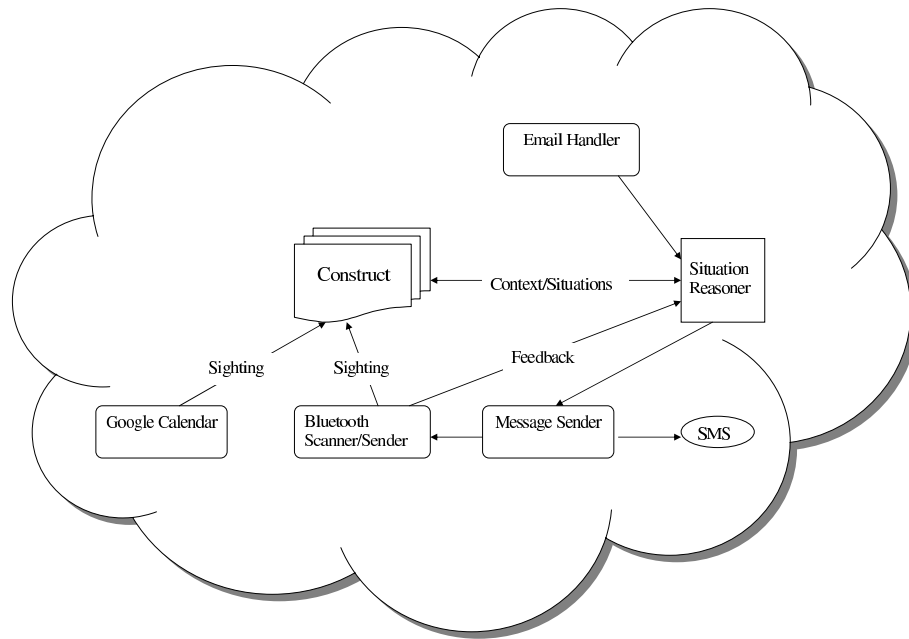


Figure 2: Scatterbox system architecture.

Consider a reminder email from a calendar application processed and sent to a Scatterbox user, Bruce. Bruce has a meeting scheduled after lunch with one of his students in his office. Bruce is currently upstairs in the coffee area having lunch with one of his colleagues. Bruce’s calendar application sends him an email to remind him of this appointment. In this case an email reminder will be of no use to Bruce, as he is not currently at his computer. Scatterbox should forward messages that normally arrive in a user’s email inbox to their mobile device, if they are away from their computer and the message is deemed to be contextually relevant by the system. Scatterbox’s contextual inputs at this point are: the current time of day, schedule information from Bruce’s calendar application (which includes both the time it takes place and the room it takes place in, as well as who else is invited), Bruce’s current position in the building, the details of other participating users that are currently in the room with Bruce, (in this case, his colleague’s details). Scatterbox should decide if this message is contextually relevant and whether or not the message will be pushed to his mobile phone. Variants of this situation that would not lead to a message being sent would be: had Bruce been in his office at the time the meeting was to start; had the student that Bruce was to meet also been up in the coffee area at the time the meeting was to start. In this case a message will not be sent, as Bruce’s social context overrides the location information, and we infer that they have decided to change the venue of their meeting.

Homecare application. In collaboration with the MATCH project and Kelvin Connect. Verification issues include configuration change impact analysis, authorisation policy enforcement by access control systems, and reliance and timing properties. These will become more complex (giving rise to more interoperability issues)

as the applications are scaled up to include more devices (e.g. movement sensors, heat detectors, door sensors) and a variety of communication mechanisms. We will have a unique opportunity to be involved in the design process as the new applications are developed and assessed.

From a modelling perspective, MATCH is an event driven system. An event is a requirement that when a given condition is met the system takes appropriate action. Typical examples of events include: “if the front door is left open and nobody is downstairs, then send a message to a stakeholder”, “if the lights are left on and nobody is in the house, then turn the lights off”, “If Bill is laid down but not in bed, then contact Gill”, etc.

The MATCH system consists of a set of components and a set of users. The set of components can be split into 4 main categories: sensors, alert triggers, tasks and outputs.

Sensors are hardware or software components that update a set of variables unique to that sensor. A sensor will usually be hardware which monitors real world events, but it may be a software component that monitors other components within the MATCH system. Examples of hardware sensors include: thermostats, pressure pads, motion detectors, etc. Software sensors can be used to combine hardware sensors. For example, if there were a sensor on every window and door to a house, which detects if it is closed and locked or not, a binary software sensor “house secure” would return the value true iff all the door and window sensors return true. Other software sensors may record how often a system component is used, spare capacity in storage devices, the latency of a given system component, etc.

Alert triggers are software components that monitor sensor variables and trigger an alert under a given condition. When an alert is triggered, one or more messages, of a given type, will be placed in one or more channels. These components are used to check for conditions such as: the front door is unlocked and nobody is downstairs, the oven is on and nobody is in the kitchen, the TV volume is above a certain value and it is later than 11pm etc.

Tasks are internal components, normally software, but can be hardware. They poll one or more channels for messages. When a message or set of messages of a given type are received, then some action is taken and one or more messages of a given type are placed in one or more output channels. Typical tasks include: system message to natural language conversion, string to voice XML, voice XML to wav file, etc. If these tasks are software components then it is reasonable to assume that we can create them as and when needed, meaning that two or more instances of the same component can be used in parallel without conflict. Were as hardware components may be restricted by physical capacity constraints.

Outputs are components that affect the physical world, usually hardware but can be software. They poll one or more channels for messages. When a message or set of messages of a given type are received, then an action is performed by the output component. Output components include: speakers, GUIs, lights, mobile phone, tactile devices etc

An event path is an ordered subset of connected system components which can fulfil the requirements of a given event. The event path must start with an alert trigger which monitors an appropriate subset of the sensors to determine if the event condition is met or not. The path will terminate on one or more output devices. The path may contain loops such as feedback loops or iterative improvement cycles. Each internal component of the path must honour the type requirements of the components adjacent to it in the path.

A configuration is a snapshot of the system consisting of all components, both software and hardware, and the channels connecting them, reflecting one or more event paths that have been set within the system.

The policies are rules that both define undesirable/illegal behaviour and provide guidance on how non-determinism is resolved within the system. Policies can be used to express stakeholders preferences for things such as output content, frequency and devices. They can also be used to define acceptable/desirable/annoying use of output devices.

D. The Research Programme in Birmingham

Research Objectives.

1. To develop frameworks for modelling interoperability requirements in pervasive systems (specifically, interaction requirements, performance and security).
2. To develop verification techniques that are tailored to analysing the requirements in models of pervasive systems.

3. To evaluate the techniques on significant case studies in a realistic application domain of distributed systems.

The **first phase** of the project consists of an exploration of pervasive computing case studies for the chosen themes of access control, security assurance and performance. This phase will last four months.

WP1. Case Study Exploration

(Month 1 to Month 4)

[ALL (4PM EACH)]

OVERVIEW AND METHODOLOGY We will exhibit interoperability issues in the case studies. In order to expose a wide variety of verification needs, the case studies are drawn from three layers: application, infrastructure and network.

TASKS

- 1.1. Identify the pervasive computing applications and interoperability requirements.
- 1.2. Analyse the applications in terms of the themes of access control, performance and security assurance.

In **phase two** of the project, which lasts one year, we formalise the requirements identified in the first phase. We formalise both individual component behaviour as well as protocols between individual components. We expect the case studies to raise important access control problems which we also formalise.

WP3. Protocols Between Individual Components

(Month 5 to Month 16)

[LIV (6PM), BIR (6PM)]

OVERVIEW The aim of the work package is to analyse the case studies from WP1 in order to study the interaction between components of pervasive computing systems, and model the interoperability requirements.

TASKS

- 3.1. Develop formalisations of the protocols using existing notations at a level suitable for analysis.
- 3.2. Provide a basis for the analysis of co-operation through (probabilistic/non-probabilistic) temporal properties.

METHODOLOGY We will consult with infrastructure and component developers including our industrial partners to obtain the actual protocols used in our case studies. This will include application-level protocols specific to the case studies (for example, protocols related to the chemotherapy sensors and mobile equipment), as well as low-level protocols of authentication and data distribution. We will formalise the expected properties of the protocols for later analysis using a suitable temporal logic-based language. The development of such a language will heavily use ideas from WP1. It is important to note that, as the proposed protocols address new problem domains we cannot simply re-use existing industry standards or formalisms (e.g., [?]).

WP4. Information Access

(Month 5 to Month 16)

[BIR (6PM), GLA (6PM)]

OVERVIEW Describe and develop the information access problems which arise in the case studies of WP1.

TASKS

- 4.1. Define access control scenarios which specify and enforce dynamic access control policies for mobile code.
- 4.2. Develop access control policies and policy verification.

METHODOLOGY The case studies of WP1 will identify issues and problems of access control (typically what component or group of components has the right to access a certain resource in a pervasive computing infrastructure) and privacy. In this workpackage we will develop and refine models of access control systems, and techniques for proving properties about them. It is clear that access control systems which assume a root (super-user) agent are not suitable for pervasive computing systems due to the de-centralised, ad-hoc nature

of networking pervasive infrastructures are based on. Pervasive computing therefore requires access control models with the authority to change permissions devolved to appropriate users. Sensitive data must be provided with privacy-related meta-data which devices are able to interpret and enforce [?]. *Role-based* languages are widely used here [?], as well as languages for modelling access control which have emerged in recent years, such as SPKI/SDSI [?] and XACML [?]. We will evaluate these, by using them to describe access control scenarios from WP1. In collaboration with our partners, we will identify the deficiencies of the languages and their ability to scale up for pervasive computing. We will develop verification methods, based on our early work [?].

After completion of the second phase of the project we will have formalised the requirements for the pervasive systems we consider. In the **third phase**, lasting one year, we develop suitable technologies to verify these requirements formally. Properties specified using temporal logic can be verified efficiently via model checking. The extensions identified in WP2–4 to current formalisms require suitable extensions to model checking techniques. In addition, deductive techniques like abstraction complement model checking very well. For the verification of the properties of the communication protocols we use process algebra.

WP5. Process Algebras

(Month 17 to Month 28)

[GLA (6PM),BIR (6PM)]

OVERVIEW We will apply process algebra formalisms to verify the properties concerning communication and synchronisation formalised in WP2–4. For example, we will use the pi-calculus (and variants and extensions) to model communication protocols involving mobility. Such models are obtained by translating the pseudo code and other notations obtained in WP3 to the pi-calculus. We will also use stochastic process algebras such as PEPA [?] for quantitative performance analysis of system components. We will also tackle foundational issues such as the relationship between continuous and discrete, and deterministic and stochastic dynamics, based on our work on modelling chemical dynamics (this in turn derives from work on modelling telecommunications, so there will be an interesting synthesis of research effort).

TASKS

- 5.1. Model protocols of WP3 in appropriate variants and extensions of pi calculus. Use and extend tools to verify the intended properties.
- 5.2. Develop stochastic process algebras models of relevant pervasive scenarios and use them to verify the properties previously determined. Extend/re-define the frameworks as necessary.
- 5.3. Develop formal relationships between stochastic and deterministic models (both discrete and continuous).

METHODOLOGY Process algebras have proved to be very useful verification formalisms in a variety of different contexts, including modelling and verifying component behaviour, protocol analysis, and information access. We will employ and extend formalisms including pi calculus and process algebras to work on the specific high-level problems arising from WP2-4. In particular, we will start by examining PRISM [?], extending as required.

WP7. Deductive Techniques

(Month 17 to Month 28)

[BIR (6PM),LIV (6PM)]

OVERVIEW To complement and support model checking, we will work on adapting and further developing deductive techniques for the verification of the interoperability properties identified earlier. Specifically, we will focus on abstraction and scalability.

TASKS

- 7.1. Identification of opportunities for abstraction arising in WP5 and WP6. We will use deductive techniques to prove the correctness of the abstractions.
- 7.2. Develop temporal deductive techniques for parametrised systems and infinite state systems, in order to analyse the problems developed in WP3,4.

METHODOLOGY For behavioural and performance analysis under the deductive approach, varieties of temporal logic have been shown to be important. Thus, in investigating deductive techniques for these problems, we will involve our previous work on deductive temporal verification [?]. While the link between deductive techniques and abstraction has been commented upon, there has been little work so far on linking specific deductive techniques with abstraction. Thus, we will utilise our work on abstractions [?] and deduction [?] in order to implement abstractions using deductive techniques and to establish the validity of abstractions using deductive techniques. We will also use standard compositional reasoning techniques (such as rely/guarantee methods). Typically, these techniques will be used to reduce highly complex scenarios to a more manageable size, or to directly tackle problems with a potentially infinite state space. In particular pervasive systems often admit an unbounded number of components, hence techniques to reduce a possible infinite state space to a finite one are necessary (important, e.g., in protocol verification).

In the **final phase** of the project we will revisit the case studies in the light of the new technologies developed.

WP8. Case Study Analysis and Synthesis

(Month 29 to Month 36)

[ALL (8PM EACH)]

OVERVIEW Develop to conclusion all the case studies, draw and exchange experiences and lessons, feedback results to industrial collaborators, write up results and final project report. Most of these activities will have been carried out throughout the project, here we will ensure that all results are brought together in a comprehensive way and generic guidelines are extracted.

TASKS

- 8.1. Refine and comprehensively review all technical and case study results.
- 8.2. Evaluate hypothesis in light of results obtained.

METHODOLOGY Review the verification needs uncovered during the project, and how the new (qualitative and quantitative) techniques we have developed for modelling and for expressing properties, and the new combinations of these (and other) techniques can meet the verification needs.

WP9. PhD Projects

(Month 13 to Month 48)

[ALL (36PM EACH)]

OVERVIEW Research Fellows will lead the work on the pervasive computing applications (years 1-3). This work will identify directions for specific, longer-term extensions of the verification techniques. To investigate these directions and to bring to train a new generation of researchers, 3 PhD students, one at each site, will work on extending the verification technology in this way (years 2-4).

METHODOLOGY The staggered approach allows the project to be managed more efficiently (e.g. not everyone starting at the same time). The identification of PhD topics are more integrated into the project as it unfolds, and provide the PhD student with a year after the project to write up. The Research Fellows and PhD students are each appointed for three years, but the PhD students start in Year 2.

The topics will be identified precisely during the first year of the project. We have a number of topics in mind right now, but we expect these to be refined as we delve into this research and in particular, gain further experience of the case studies. Some example topics include: *algorithms for model checking lower/upper bounds for subclasses of parametrised probabilistic systems; deductive verification of security in pervasive systems; modelling and relating individual and group behaviour in RFID systems using stochastic process algebra; BDI model checking of autonomous pervasive components; verifying fluid organisations; and efficient temporal verification.*

The PhD studentships (starting in 2nd year) will allow exploration of topics uncovered during the first year of the project, especially technical challenges related to the case studies. There, combined with space restrictions, means that we deliberately give no detailed workplan for the work for the PhD-students in the proposal. The precise nature of the work for them will only become clear at the end of the first year, when it possible to identify precise topics. We are confident that there is plenty of scope for the PhD projects.

References

- [1] R. Bordini, M. Fisher, W. Visser, and M. Wooldridge. Verifying Multi-Agent Programs by Model Checking. *Journal of Autonomous Agents and Multi-Agent Systems* 12(2), 2006.
- [2] M. Calder and A. Miller. Feature Interaction Detection by Pairwise Analysis of LTL Properties. *Formal Methods in System Design* 28:213–26, Springer, 2006.
- [3] M. Calder and A. Miller. An automatic abstraction technique for verifying featured, parameterised systems. *Theoretical Computer Science*, Elsevier, to appear, 2007.
- [4] D. Clarke, J.-E. Elien, C. Ellison, M. Fredette, A. Morcos, and R. L. Rivest. Certificate chain discovery in SPKI/SDSI. *Journal of Computer Security* 9(4), 2001.
- [5] S. Creese, M. Goldsmith, W. Roscoe and I. Zakiuddin Authentication for pervasive computing. In *Proc. 1st Int. Conf. on Security in Pervasive Computing*, 2003.
- [6] A. Degtyarev, M. Fisher, and B. Konev. Monodic Temporal Resolution. *ACM Trans. Comput. Logic* 7(1), 2006.
- [7] M. Fisher, C. Dixon, and M. Peim. Clausal Temporal Resolution. *ACM Trans. Comput. Logic* 2(1), 2001.
- [8] X. Jiang and J.A. Landay. Modeling privacy control in context-aware systems. *IEEE Pervasive computing*, 2002.
- [9] OASIS (Organization for the Advancement of Structured Information Standards). *XACML Specification*, Feb. 2004. www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.
- [10] J. Rutten, M. Kwiatkowska, G. Norman and D. Parker. Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems. CRM Monograph Series, vol. 23, AMS March 2004.
- [11] W. Yao, K. Moody, and J. Bacon. A model of OASIS role-based access control and its support of active security. *ACM Trans. Information and System Security* 5(4), 2002.
- [12] N. Zhang, M. D. Ryan and D. Guelev, Evaluating Access Control Policies Through Model Checking. In *Proc. 8th Information Security Conference. LNCS*, pp. 446-460, Springer, 2005.
- [13] S.Knox and R.Shannon and L.Coyle and A.Clear and S.Dobson and A.Quigley and P.Nixon, Scatterbox: Context-Aware Message Management, In *Revue d'Intelligence Artificielle*, 2008.
- [14] S. Knox and A. K. Clear and R. Shannon and L. Coyle and S. Dobson and A. Quigley and P. Nixon, Towards Scatterbox: a Context-Aware Message Forwarding Platform. In *Fourth International Workshop Modeling and Reasoning in Context (MRC 2007)*, pp. 13-24, 2007.
- [15] The TPM, <ftp://ftp.cs.bham.ac.uk/pub/authors/M.D.Ryan/08-intro-TPM.pdf>.
- [16] Quote from Brian Berger in TCG press release by Wave Systems.
www.trustedcomputinggroup.org/news/press/member_releases/WAVETCGPROMOTIONMW5_31_FINAL_.pdf
See also “TCG timeline”, revised April 2008.
www.trustedcomputinggroup.org/about/corporate_documents/TCG_timeline_rev_april_2008.pdf
- [17] Trusted Computing Group. TPM Specification version 1.2. Parts 1–3. www.trustedcomputinggroup.org/specs/TPM/
- [18] E. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. In B. Pfitzmann and P. Liu (Eds.), *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS 2004)*, pp. 132-145, ACM press, 2004.
- [19] B. Smyth, M. Ryan, and L. Chen. Direct Anonymous Attestation (DAA): Ensuring privacy with corrupt administrators. In *proceedings of the Fourth European Workshop on Security and Privacy in Ad hoc and Sensor Networks (ESAS'07)*. Lecture Notes in Computer Science (LNCS), volume 4572, pp. 218-231, Springer-Verlag.
- [20] M. Backes, M. Maffei, and D. Unruh. Zero-Knowledge in the Applied Pi-calculus and Automated Verification of the Direct Anonymous Attestation Protocol. In *Proceedings of 29th IEEE Symposium on Security and Privacy*, May 2008.