CASPEr: Containment-Aware Security for Pervasive Computing Environments

Boris Dragovic

St John's College

University of Cambridge

A thesis submitted for the degree of

Doctor of Philosophy

March, 2006

I dedicate this thesis to all those out there who are able and deserving but never get a chance.

Acknowledgements

Without the unreserved love and support from my family I would never be where I am now...

There are no words that can express my gratitude to my supervisor, Prof Jon Crowcroft, for all his support provided during the PhD and for admitting me in the first place. I am also thankful for all those blitz idea exchanges which regularly left me more confused than I used to be previously. I am just about now starting to appreciate them...

From a different perspective, I thank Dr Mark Nicholls, my College tutor who always endeavoured to fulfil all my personal needs. He contributed greatly to the feeling that the College cared about myself to a great extent.

A number of people, to whom I am indebted, have invested considerable time in reviewing this thesis in various stages of its creation. These are: Calicrates Policroniades, Martin Vechev, Rashid Mehmood, Julian Chesterfield, Anil Madhavapeddy, Evangelos Kotsovinos, Russel Ross and Meng How Lim. During my PhD I have collaborated with a number of people from the Computer Lab and wider — I am thankful to them for sharing ideas, knowledge and experience with me. Special thanks for thesis related idea exchanges to Calicrates, Pablo Vidales, Javier Baliosian and David Richerby. Last but not the least, I thank Petya Blumbach for all her efforts to contribute some style to my written English.

The time spent in Cambridge would have been substantially less pleasurable had there not been sincere friends around. Their spirit and support was invaluable. Thank you Cali and Pablo for getting me so badly hooked on salsa, teaching me to love hot food, demonstrating me the intricacies of tequila drinking, all those moments in between and, mostly, for putting up with me all this time — it must have been a nightmare! To Martin, who I have met slightly later, I thank for all those inspiring chats and chalga gym sessions. Also, thank you amigos for numerous other things I should really not mention here... for the lack of space, of course.

I am grateful to Jovana and Dusanka for being there when I needed them the most and when it mattered the most.

Last but not the least, I would have never arrived at this point in the PhD path without generous financial support from St John's College Benefactors' Scholarship, Overseas Research Scholarship (ORS), Cambridge Overseas Trust and Cambridge Philosophical Society.

Summary

Over a decade after Mark Weiser defined and inaugurated the study of Ubiquitous Computing, its elusive shapes are starting to appear through the proliferation of mobile, embedded computing and wireless communications technologies. The paradigm shift away from classical distributed computing not only gives rise to novel security challenges, but also causes qualitative shifts in the attributes upon which many existing security models rely. The fluidity of data movement, for example, leads to the dissolution of the notion of the secure perimeter. With it goes the ability to predict the range contexts in which a piece of sensitive information may exist over time and, thus, the threat models that it may experience.

In this thesis, we identify a class of information confidentiality threats that arise as a side-effect of otherwise legitimate information management and handling procedures as employed in a particular context. They occur unwittingly on the part of the information custodian. We refer to them as *information exposure* threats and present their characterisation and systematisation. Current security engineering practises, when applied to the threats, result in a static solution to a dynamic problem, and often in an indiscriminate manner. Thus, they turn out to have an overly oppressive effect on information availability, as well as on wider system usability, most of the time. This seriously undermines some of the prime qualities of the ubiquitous computing vision and represents a hurdle on the path to its full embracement.

We offer a novel paradigm, CASPEr, that aims at continuous and adaptive provision of adequate protection in the face of unpredictably changing information exposure threat models whilst making an active effort to minimise its intrusiveness. As a basis for CASPEr, we develop a data-centric model of the world, structured around the concept of a *container* as a protective enclosure. The model allows for fine-grained reasoning about threat effects and facilitates the placement of protection toward the threatened assets (data) — confining its impact. We introduce the concept of a *level of exposure* and develop a dynamic programming algorithm which, together, provide for an optimal threat mitigation strategy discovery. CASPEr champions the utilisation of standard information handling and management procedures for threat mitigation, adding another dimension to the ability to sustain information availability and system usability. We also show how the full spatial and temporal continuity in data protection is achieved. The contribution encompasses μCASPEr, a policy-based specialisation of CASPEr and an adaptation of its architecture for highly-constrained platforms.

The feasibility of CASPEr deployment is discussed theoretically, at the architectural and the conceptual levels of abstraction. The overall flexibility of the CASPEr model is shown to account for the envisaged heterogeneity of the target platforms. CASPEr facilitates incremental deployment, with graceful service degradation in constrained environments. Moreover, both CASPEr and μCASPEr have been designed to fully support operational autonomy of the hosting devices. We show, by means of a qualitative comparison, how CASPEr is complementary to the major information security paradigms, filling a gap in the big picture not previously addressed in a systematic and adequate manner. With the μCASPEr evaluation we demonstrate the applicability of CASPEr concepts even on the thinnest of the ubiquitous computing platforms.

Declaration

I hereby declare that this dissertation is not substantially the same as any I have submitted for a degree or diploma or any other qualification at any other university. Further, no part of the dissertation has already been or is being concurrently submitted for any such degree, diploma or other qualification.

In Chapter 5 we present a policy based system that instantiates a number of concepts presented in the thesis. The policy model utilised originates from [BS04]. The manner in which the model is deployed, including the policy structuring, to address information exposure threats within the constraints of the target setting and with the resulting characteristics is completely my own contribution. The system architecture presented in Section 5.3 resembles that presented in [VBS+05]. The resemblance, however, is to the extent required by the common policy model solely, as indicated explicitly in the chapter text. Otherwise, the identified architecture components, their individual roles as well as the specification of the related processes are fully my own contribution. This is further supported and acknowledged as presented in [DBVC06] [1] — a publication resulting from collaboration among the authors.

Other chapters are solely my own work even if they share ideas developed jointly with my supervisor. The section "Author Publications" lists all publications co-authored by myself during the course of this PhD course, sharing ideas with this thesis and otherwise, along with the full authorships.

This dissertation does not exceed sixty thousand words, including tables and footnotes, and it does not contain more than one hundred and fifty figures.

Boris Dragovic, March 2006.

[1] The same paper was initially submitted to ESORICS 2005, with the submission deadline of 01/04/05, but was rejected on the grounds of being outside the target scope.

List of Publications

The following list contains references to all the publications that I co-authored during the course of this PhD course.

Referred Publications

- B. Dragovic and C. Policroniades. Information SeeSaw: Availability vs. Security Management in the UbiComp World. In *Proceedings of the 2nd VLDB Workshop on Secure Data Management (SDM '05)*, *Lecture Notes in Computer Science*, vol. 3674, pp. 200–216. Springer-Verlag, Berlin Heidelberg, Germany, 2005.
- B. Dragovic and J. Crowcroft. Containment: from Context Awareness to Contextual Effects Awareness. Presented at the *2nd ICPS International Workshop on Software Aspects of Context (IWSAC '05)*, Santorini, Greece, July 2004.
- J. Scott and B. Dragovic. Audio Location: Accurate Low-Cost Location Sensing. In *Proceedings of the 3rd International Conference on Pervasive Computing (Pervasive '05)*, *Lecture Notes in Computer Science*, vol. 3468, pp. 1–18. Springer-Verlag, Berlin Heidelberg, Germany, 2005.
- B. Dragovic and J. Crowcroft. Information Exposure Control through Data Manipulation for Ubiquitous Computing. In *Proceedings of the 2004 Workshop on New Security Paradigms (NSPW '04)*, pp. 57–64. ACM Press, New York, NY, USA, 2005.
- B. Dragovic and J. Crowcroft. Context-Adaptive Information Security for UbiComp Environments. Presented at the *2nd UK-UbiNet Workshop: Security, Trust, Privacy and Theory for Ubiquitous Computing*, Cambridge, UK, May 2004.
- A. Fernandes, E. Kotsovinos, S. Ostring and B. Dragovic. Pinocchio: Incentives for Honest Participation in Distributed Trust Management. In *Proceedings of the 2nd International Conference on Trust Management (iTrust '04)*, *Lecture Notes in Computer Science*, vol. 2995, pp. 63–77. Springer-Verlag, Berlin Heidelberd, Germany, 2004.

- B. Dragovic. Knowing Your Place: Containing Ubiquitous Systems. Presented at *1st UbiComp Workshop on Location Aware Computing*, UbiComp 2003, Seattle, WA, USA, October 2003.
- B. Dragovic and J. Crowcroft. Containment: Knowing Your Ubiquitous Systems' Limitations. Poster and extended abstract in *Adjunct Proceedings of the 4th International Conference on Ubiquitous Computing (UbiComp 2003)*, Seattle, WA, USA, October 2003.
- R. Chakravorty, P. Vidales, B. Dragovic, C. Policroniades, and L. Patanapongpibul. Ubiquity in Diversity – A Network-Centric Approach. Poster and extended abstract in *Adjunct Proceedings of the 4th International Conference on Ubiquitous Computing (UbiComp 2003)*, Seattle, WA, USA, October 2003.
- P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebar, I. Pratt and A. Warfield. Xen and the Art of Virtualization. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP '03)*, pp. 164–177. ACM Press, New York, NY, USA, 2003.
- B.Dragovic, E. Kotsovinos, S.Hand and P. Pietzuch. XenoTrust: Event-based Distributed Trust Management. In *Proceedings of the 2nd IEEE International Workshop on Trust and Privacy in Digital Business (DEXA TrustBuss)*, pp. 410–414.
- B. Dragovic, S. Hand, T. Harris, E. Kotsovinos and A. Twigg. Managing Trust and Reputation in the XenoServer Open Platform. In *Proceedings of the 1st International Conference on Trust Management (iTrust '03)*, *Lecture Notes in Computer Science*, vol. 2692, pp. 59–74. Springer-Verlag, Berlin Heidelberd, Germany, 2003.

Technical Reports

- P.R. Barham, B. Dragovic, K.A. Fraser, S.M. Hand, T.L. Harris, A.C. Ho, E. Kotsovinos, A.V.S. Madhavapeddy, R. Neugebauer, I.A. Pratt and A.K. Warfield. Xen 2002. *Technical Report 553*, University of Cambridge Computer Laboratory, January 2003.

Submitted for Publication

- B. Dragovic, J. Baliosian, P. Vidales and J. Crowcroft. Autonomic System for Context Adaptive Security in Ubiquitous Computing Environments. Submitted to *Elsevier Journal on Pervasive and Mobile Computing*, April 2006.

Glossary

ACL	Access Control List
AES	Advanced Encryption Standard
CASPEr	Containment Aware Security for Pervasive Computing Environments
CC	Common Criteria
CCL	Client Connectivity Layer
CML	Context Modelling Layer
CMM	Containment Model Manager
CR	Conflict Resolver
CS	Control Set
DAC	Discretionary Access Control
DCON	Information Dissemination Control
DRM	Digital Rights Management
DT	Deterministic Transducer
DVD-CSS	Digital Video Disc Content Scrambling System
ECA	Event-Condition-Action
EM	Enforcement Manager
ER	External Repository
FSA	Finite State Automata
FSM	Finite State Machine
FST	Finite State Transducer
GUI	Graphical User Interface
HCI	Human Computer Interaction
IT	Information Technology
LoE	Level of Exposure
MAC	Mandatory Access Control
MLS	Multi-Level Secure
MP	Message Push
OCD	Optimal Cover Discovery
PBT	Payment-Based Type
PC	Personal Computer
PCIM	Policy Core Information Model

PDA	Personal Digital Assistant
PDM	Policy Deployment Module
PDP	Policy Decision Point
PEM	Policy Evaluation Master
PFT	Payment-Free Type
PM	Policy Manager
PSI	Policy Specification Interface
RBAC	Role-Based Access Control
TCB	Trusted Computing Base
TF	Tautness Function
TFFST	Finite State Transducer with Tautness Functions and Identities
UCON	Information Usage Control
VM	Virtual Machine
XML	Extensible Markup Language
ZIA	Zero Interaction Authentication

Contents

List of Figures

Chapter 1

Introduction

The most profound technologies are those that disappear. They wave themselves into the fabric of everyday life until they are indistinguishable from it. This is how, in his seminal paper [Wei91], Mark Weiser referred to what is nowadays a common term — Ubiquitous Computing. We are irrevocably marching toward it, and possibly more so than we realise. Nowadays even non-technical people interact on a regular basis with dozens of microprocessors, blissfully ignorant of it. Tasks being as mundane as conducting business, driving, communicating, shopping, doing household activities and entertaining ourselves to mention only a few.

On one estimate by IDC[1], people in United States interact with about 150 embedded systems every day, whether aware of it or not. Less than 2% of more than 3 billion 8-bit microprocessors sold on the market yearly (as of 2005) are destined for conventional Personal Computers (PCs), the vast majority of the rest are deployed in embedded systems[2]. Computing is inevitably entering every aspect of our lives. Perhaps most notably, this is in the areas of mobile communications and mobile computing — ubiquitous computing forerunners. The world appetite for mobile phones has exceeded even the most optimistic expectations, according to Gartner[3] analysts. They estimate 2.6 billion units in use by the end of 2009. The fastest growing category of the device being, so called, smart-phones — transforming an otherwise purpose specific platform into a more general computing platform. On the purely mobile computing front, Gartner estimates the market to grow by 18% in 2005[4] reaching 55.4 million units — while the overall PC market will grow at less than 8%.

The boom goes hand-in-hand with the development and proliferation of wireless communications technologies and services. If one considers public places such as airport lounges, coffee shops or even streets, it is becoming increasingly difficult to imagine them without network access of one form or another. This is even more prominent in the business and

[1] `http://www.idc.com`

[2] According to the Semiconductor Industry Association, `http://www.sia-online.org`.

[3] `http://www.gartner.com`

[4] The exact figures for 2005 were not available at the time of writing of this thesis.

domestic environments. Computing power and interconnection technologies are everywhere, representing the basic substrate for Mark Weiser's vision.

To realise the holistic vision of ubiquitous computing we are, however, missing the global synergy of the parts: "bringing together all these hidden processing nodes into a coherent entity worthy of being called a system" [SC03]. Creating the coherent architecture involves solving a number of hard problems. Some challenges are purely technical but some, perhaps more importantly for actualising the vision, are socio-technical. Security is, by and large, one of the latter. It is also a paradigmatic example of a property of a system as a whole. If the holistic vision of ubiquitous computing is ever to set off truly, the security research community has to come up with *ubiquitous security*. This is as opposed to the patchy, incoherent, add-on and, more often than not, highly intrusive approach to security in current systems.

The issues in the domain of ubiquitous computing security can be, broadly speaking, divided into two categories: i) those solvable by the tools we have developed for the more traditional computing settings, and ii) the novel ones requiring original solutions. The latter stem from the manner in which the vision of ubiquitous computing redefines the notions of computation and human computer interaction as well as the concepts of data input and output, its persistence and ownership. With respect to the abundance of mobile devices itself, the threat it poses is generally recognised as twofold: firstly, mobile computing devices can be used to introduce large volumes of unauthorised data or programs to corporate computing infrastructures, but perhaps more worryingly, they are regularly used to store, access and otherwise leverage extensive amounts of sensitive data in a variety of contexts — characterised by different and unpredictably varying threat models. We refer to the quality of ubiquitous systems that prompts the latter threat as *information omnipresence*. It denotes information being available *where*, *when*, in the *form* and through the *means* as needed by a user for completion of a task. Information omnipresence is seen as one of the prime qualities and promises of ubiquitous computing. Along with its benefits, of course, come a number of security challenges.

In this thesis we identify and systematise a novel class of information security threats, that we name *information exposure* threats, brought into forefront by the advent of ubiquitous computing, in general, and through its promise of information omnipresence and the manners in which it is exploited, in particular. The thesis presents a novel paradigm [DC04] that we baptise CASPEr — ***C****ontainment* ***A****ware* ***S****ecurity for* ***P****ervasive Computing* ***E****nvi****r****onments*, aimed specifically at addressing information exposure threats.

1.1 Motivating Scenarios

To create an intuitive feel for information exposure threats and illustrate the desired behaviour of CASPEr we offer a small set of example scenarios. The scenarios are also intended to hint at the CASPEr scope. Let Alice[5] assume a job that regularly exposes her to a large

[5] A character well known to most computer security researchers, thus requiring no further introduction.

amount of sensitive information. Alice spends most of the working hours outside the secure perimeter where she relies on a number of mobile computing devices — a laptop, Personal Digital Assistant (PDA) and mobile phone, to provide her with the corporate, sensitive information wherever and whenever needed.

- Alice is leaving her office. As she steps across the physically secure perimeter, data classified as *Top Secret*, contained on her laptop and PDA get securely destroyed. *Secret* data gets safely encrypted on the laptop but erased from both the PDA and mobile phone to account for the respective likelihoods of physical abduction of the devices themselves (mainly due to their physical size and mobility characteristics).

- Alice is giving a presentation to a wide auditorium. The slides contain corporate *Confidential* information and comments. However, the sensitive pieces of information are displayed only on the laptop screen — observable solely by Alice, while the information accessible to the auditorium gets automatically pruned in an adequate fashion.

- Alice is in an airport lounge, she wishes to read a sensitive document while she waits. To avoid information exposure, the document is displayed in a Graphical User Interface (GUI) window of a strictly constrained size. She also has an option of using the PDA display for the purpose. The measures ensure that no discreet, over-the-shoulder, looks are possible.

- To open the document, in the setting of the previous use case, Alice needs to undergo an additional level of password authentication. Rather than using the widely observable laptop keyboard, PDA touchscreen and a specialised interface which reduces the likelihood of correlating physical and the resulting input sequences are automatically activated for the password entry. Alice is suitably notified of the decision and the reasons leading to it.

- Alice sees a friend nearby and goes to greet her. As the distance between Alice and her laptop increases the window displaying the document shrinks and subsequently gets minimised. Prolonged absence causes high sensitivity data to be fully erased from the laptop storage as well as from all the intermediate buffers.

- Actually, rather than being destroyed, the sensitive documents are migrated to Alice's PDA which suffers less likelihood of abduction as it is attached to her belt. The transfer is accomplished over a secure (encrypted) link. Upon return to her laptop, Alice wishes to continue the work and the documents must be reinstated on the laptop. A spatially and directionally localised Infra-Red (IrDA) link is used instead of other available technologies. Non sensitive transfers proceed over a higher capacity, but less secure, channel.

- Back in the office, Alice wishes to listen to an audio memo from a closed meeting. She puts it up on a set of speakers. The volume is automatically limited so that the audio cannot be heard in neighbouring offices. As a colleague from a different department approaches the door the volume goes down. As he opens the door and enters the audio replay is switched to Alice's headphones.

Thinking about it, we can notice that analogous adaptive behaviour is typical for humans in their everyday lives, albeit to an extent limited by their cognitive (in)abilities. However, it is unrealistic to expect humans to be able to grasp the complexities leading to security issues in highly heterogeneous and dynamic pervasive computing environments. The average user of mobile computing technologies nowadays is fully technologically illiterate and should not be relied on, under any circumstances, to be able to reason about such threats at all. From the cognitive point of view, it is well known [AS99] [YBAG00] [YBAG04] that even the simplest of the password schemes pose unsurmountable usability problems for the average user in general. Along with the identification of the existence of information exposure threats themselves, as well as the recognition of their potential severity in the highly dynamic ubiquitous computing environments, this is precisely where the seed of our motivation for designing CASPEr originates.

1.2 Information Exposure — What to Watch Out for

Information exposure threats are a subset of the broader class of information leakage threats. Information leakage itself can be described as a consequence of an event that causes information to become available to unauthorised parties from what was designed to be a "closed"[6] system. An information leakage threat is a prospect of such an event occurring.

To be able to characterise information leakage, and thus distinguish information exposure, we firstly introduce the notion of information[7] *custodian*. An information custodian is a person in a legitimate[8] possession of information. A custodian is not necessarily an information owner or its user. Being the owner does not imply being in custody of the "owned" data at any point in time. Being in custody of a piece of information does not, on the other hand, imply any reference to the information — as assumed by the notion of a data user. For example, a postal courier can be seen as in custody of the mail they are to deliver.

Based on the information custodian's level of conscious awareness of and their involvement in information confidentiality compromise we can distinguish between:

- Intentional, and

[6] A system where an explicit distinction exists between authorised and unauthorised users, specified by a security policy [And01] and established through a set of dedicated mechanisms.

[7] Information may exist in a digital system only in the form of its particular representation, i.e. encoding, referred to as data. Thus, we use the term data to denote its actual information content and its characteristics.

[8] As determined by a set of authentication and authorisation mechanisms and defined by a security policy.

- Unintentional information leakage.

Intentional information leakage arises in either of the following two situations: i) an information custodian makes an active effort to leak the information, or ii) an information custodian fails to make an active effort to prevent information leakage they are aware of. As opposed to the intentional, *unintentional* information leakage assumes information custodian's complete ignorance of the leakage process or the respective threat. Information custody, thus, denotes solely the capacity of a principal handling the information to cause intentional or unintentional information leakage. For example, a secure courier who is in custody of diplomatic mail is neither its owner or user but is still in the capacity to relay the sensitive documents to unauthorised parties.

Information Exposure as Information Leakage

The work on CASPEr, as presented in this thesis, was motivated by the prospect of *information exposure* or, more precisely, *information exposure threats*. Information exposure represents unintentional information leakage into the environment occurring as a side-effect of an otherwise legitimate set of data management and handling procedures as employed to ensure information omnipresence in the particular context. We say that the context determines the threat model. A fundamental characteristic of information exposure channels is that they are shared with or indivisible from legitimate information flow and storage channels — they co-occur.

The term information flow traditionally applies to flows of information between objects in a system [DD77]. We extend the concept to also encompass flows of information among an object in a system and a principal in the environment. For example, a person reading a document of a computer screen implies an information flow. The term information management and handling procedure, as used in the definition of information exposure, in this case refers to the particular manner in which information is presented on the screen (GUI window size, font typeface and size, rendering method etc.). Thus, when we say that information is exposed it means that that it becomes available to unauthorised third parties as a side effect of an otherwise authorised, legitimate, information flow or storage channel due to its characteristics. In other words, information exposure channels are tightly, indivisibly, bound to the legitimate information flow and storage channels they are a direct consequence of. Moreover, information exposure and the corresponding legitimate information flow often share a single channel, as illustrated by the motivating scenarios and also in the following examples. The most notable examples of information exposure are due to:

- Physical abduction of the information containing device — where the information exists in a form that facilitates its subsequent recovery by the thief.

- Information being displayed in a form and on a screen visually accessible by a third party, either directly [TC03] or indirectly through e.g. analysing optical emanations [KA98] [Kuh05].

- Data entry on an input device which, due to its physical characteristics, allows key strokes to be captured visually or reconstructed otherwise [ZZT05] [Zal05].
- Audio information replay through speakers at a volume which facilitates, in the particular context, the information to be overheard by unauthorised parties.
- Transmission in plain-text over an unprotected wireless link whose signal penetrates into a publicly accessible area.

The only "tangible" and, thus, the instance of information exposure threats that arouses the most attention nowadays is that incurred by the physical loss or theft of a mobile computing device. Frequent high-profile incidents that make it to the media headlines regularly exercise our awareness. The magnitude of the threat is well illustrated by the results of an original survey[9] that shows that a staggering 63,135 mobile phones, 5,838 pocket PCs and 4,973 laptops have been left in licensed taxi cabs by their London customers over a period of 6 months only. Furthermore, as reported in ISBS-2004[10], 45% of large UK[11] businesses have suffered physical theft of computer equipment in 2004. The damage caused by the incidents does not come from the cost of the hardware lost and stolen but from the value of information that they contained.

A rough feel for the amount of proprietary, corporate sensitive, information potentially affected by information exposure is obtained when estimates on the size of the mobile workforce[12] is considered. IDC expects the mobile worker population to increase from 650 million in 2004 to 850 million in 2009. The US had the highest percentage of mobile workforce in 2004 — a staggering 70%, while Asia Pacific accounted for the largest quantity. The numbers speak volumes when considering that the mobile workforce implies availability of sensitive data on mobile devices and access to it in highly dynamic, unpredictably changing, contexts in a myriad fashions.

The Trust Model

The overall trust model implied by information exposure threats and assumed in this thesis is that of a *cooperating* user — a non-malicious user who needs help in protecting the data it is in custody of due to the sheer complexity of the issues involved in reasoning about its security in a particular setting. CASPEr is intended as an aid to users in protecting information in their custody as opposed to aiming at representing a non-circumventable

[9] Available freely from `http://www.pointsec.com` on request.

[10] Information Security Breaches Survey 2004, commissioned by the UK Department of Trade and done by PriceWaterhouseCoopers (PwC).

[11] Similar results have been obtained for the US by the 2004 CSI/FBI Computer Crime and Security Survey, available from `http://www.gocsi.com`.

[12] Workers leveraging mobile computing devices and interconnection technologies to accomplish business related tasks outside the confines of traditional office space.

information security mechanism as is the case with, for example, Digital Rights Management solutions and Trusted Computing platforms (Chapter 2).

1.3 The Challenges

The vision of ubiquitous computing represents a move toward purely data centric environments. The importance and semantics of hardware and perimeters fades deeply into the background and what comes into the forefront is the fluidity of data movement. The notion of secure perimeter, one of the concepts that shape the way we think about security in traditional systems, becomes dispersed and obliterated. The ability to predict threat models or control the movement of data is lost. The same applies to the means by which and the situations in which information is leveraged by users — bringing into the spotlight the risk of information exposure. Thus, proactive information security protection solutions are needed.

As widely recognised, security is a complex socio-technical problem [And93] [San03] [Sas03] [BDGS04] [Yee04]. The social aspect gains increasingly more weight as we move toward the vision of ubiquitous computing. The focus of attention on data makes users increasingly aware of any enforced restrictions and emphasises their intrusiveness and obstructiveness. For instance, the principle of the secure perimeter approach to security engineering does not sit well with the fluidity of data movement as envisaged for the novel setting. Generalising, the unnecessary adverse effects on information omnipresence and wider system usability incurred by following the traditional methodology of pushing the protection toward threat source, and away from the threatened entities (data), gains substantial importance in ubiquitous computing environments. This is perhaps most notable in the field of mobile computing security where the application of the traditional secure perimeter approaches to security engineering result in overly oppressive policies and mechanisms having an overwhelmingly adverse affect on the functionality and inherent benefits of mobile computing. Thus, we believe that the shift in paradigms toward the placement of protection as close to data as possible is required.

Stomp severely on users' usability and functionality expectations and the result is active effort to circumvent any "offending" security mechanisms — "security is not a goal most users strive for; rather, it is seen to get in the way of their production tasks." [Sas03]. This is increasingly true with the advent of pervasive computing. Sanctioning does not work, non-circumventable sanctioning is next to non-existent. In ubiquitous computing environments we need good-enough security [San03], explicitly balancing information omnipresence, together with the more general system usability and availability, with the level of protection provided. This proves to be an insurmountable challenge for classical security models due to their lack of flexibility and support for adaptive behaviour in dynamically and unpredictably changing environments. The issue is even more stressed in the context of information exposure threats due to their shared nature with legitimate information flow and storage channels. Any constraints placed on an information exposure channel are automatically reflected on the associated legitimate information flow channel — adversely affecting the information availability.

Throughout its lifetime in a ubiquitous computing system, a piece of information is subjected to highly unpredictable changes in its surrounding context and, thus, experienced information exposure threat models. Information exposure of different kinds may occur whether information is at rest or actively used, whether it is on a storage device, on a display, being transmitted or otherwise. This calls for temporal and spatial continuity in protection — which, in conjunction with addressing the above challenges, is not provided by the current solutions. Temporally, active[13] information security mechanisms are often associated with the points of explicit request, such as access, or the duration of explicit information usage. Thus, they often fail to provide protection at the point of actual threat model change which may occur throughout information lifetime. Spatially, the vast majority of temporally continuous mechanisms provide solely for information protection on storage devices — for data at rest, or within communications channels — for data in transit, typically in a data-wise indiscriminate manner. This leaves data vulnerable at a number of its other potential whereabouts within a system, such as e.g. displays.

The tightly bound, often shared, nature of the information exposure and legitimate information flow channels makes the exposure process itself, as well as its prospect, rather elusive. This is especially true when we consider that the average user of mobile computing devices nowadays is fully technology illiterate. Current security models entrust users with handling data in a security savvy manner once the identification, authentication and authorisation phases are successfully completed. In the context of information exposure, this assumes users' ability to reason about dynamically changing threat models and the implications various information management and handling procedures have on them. As previously argued, this is highly infeasible.

Potential manifestations of information exposure observable to the affected parties are typically temporally distant from the actual incident occurrence. In other words, the cause and effect of an information exposure are temporally distant and their relationship is difficult to establish. This makes the task of linking the two hard, often impossible, in practise. Consequently, threat analysis and correlation as well as evaluation of effectiveness of relevant protection mechanisms and models must, by and large, rely on non empirical methods such as historical evidence, expert opinion, etc. This is even further stressed when we consider the inherently sensitive nature of the incidents themselves which significantly limits the ability to conduct realistic experiments for evaluation purposes.

[13] As opposed to passive protection of data at rest, such as encryption.

1.4 Thesis Contribution

In this dissertation we make the following contributions, as part of our quest to address the above challenges:

1. We identify and systematise a novel set of threats to information confidentiality that we name *information exposure threats*, which are brought into the spotlight and the severity of which is emphasised by the vision of ubiquitous computing. [In Chapters 1 and 3.]

2. We develop CASPEr — a theoretical framework for fine-grained, information-centric, spatially and temporally continuous information exposure threat protection in an autonomic fashion. CASPEr aims at addressing the challenges stated in the previous section at the conceptual level of abstraction. This overall, broadly stated, contribution is comprised of the following major (sub-)contributions:

   - We introduce a novel method for structured, information-centric and fine-grained threat analysis. The method is founded on an original approach to modelling the world, physical as well as virtual, in a data object centric, fully distributed and highly flexible manner, based on the novel concept of a *container* as a protective enclosure. The model addresses the relevant requirements of the ubiquitous computing vision and honours the operational autonomy of target deployment platforms. (In Chapter 3)
   - We present an approach to strictly localised threat mitigation that maximises information omnipresence while providing for adequate protection based on threat model severity assessment. The contribution encompasses the utilisation of standard information handling and management procedures for threat mitigation, leveraging their side-effects. At the heart of the approach lies a dynamic programming algorithm that we developed together with the supporting reasoning model. (In Chapter 4)

3. We develop a policy based system, μCASPEr, that instantiates the general CASPEr concepts. The system is targeted at highly constrained target platforms and fully supports their operational autonomy. The contribution of μCASPEr is within the system architecture, based on the more general CASPEr architecture, and the manner in which the particular policy model is applied to achieve the stated goals.

Overall, CASPEr moves the state-of-art barriers in ubiquitous computing information security through filling a gap in the information security protection big picture brought into the spotlight by the vision of ubiquitous computing and not systematically addressed by the current security paradigms and mechanisms. Furthermore, CASPEr is complementary to all major information security paradigms. This is detailed in Chapters 2 and 6.

CASPEr does not, *per se*, represent a novel security mechanism that aims at addressing any individual information security threat in particular — such as wireless security or location privacy, for example. Rather, it represents a theoretical framework that allows for leveraging existing security mechanisms as well as standard information management and handling procedures in an original manner to address the holistic class of information exposure threats while providing additional benefits — making it particularly appealing for deployment in the target environment. The targeted class of information exposure threats does, indeed, encompass a number of individual, specific, threats tackled previously in isolation. With respect to addressing these, CASPEr offers additional flexibility, adaptability and management of the usability and information omnipresence tradeoffs.

1.5 Thesis Outline

The rest of the thesis is structured as follows.

Chapter 2 focuses on the outline of the major information security models and paradigms that create the big picture in which CASPEr fits and helps put CASPEr into perspective. We start of by an overview of the concepts of context and context-aware computing as used in ubiquitous computing; we describe the principles of autonomic computing, which guide some of the design decisions of CASPEr; and we present the major security challenges brought about by the vision of ubiquitous computing, as recognised in the literature. Since one of the major contributions of CASPEr is the approach to modeling the world based on the concept of *container* (Chapter 3), we also provide reference to the major related findings of the mobility theory and geo-information systems.

Chapter 3 is devoted to the presentation of a data centric approach to modelling the world based on the concepts of container and containment — the foundation stone for information exposure threat reasoning and mitigation in CASPEr. To set the scene, we firstly provide a systematisation of information exposure threats and discuss the benefits of explicit threat reasoning more generally. We proceed to formally define the model itself and detail its enabling role in spatially and temporally continuous, data item grained, information exposure reasoning. We also show how the model can be leveraged for localised information exposure threat mitigation in an autonomic fashion. The chapter concludes by formally stating properties of the model that are built on in the following chapters.

Chapter 4 presents tools and techniques, as employed by CASPEr, for dynamically balancing the *information utility* vs. level of protection tradeoff. The concept of information utility is defined as the criterion for information omnipresence characterisation. We introduce the *Optimal Cover Determination* algorithm as a dynamic programming solution for discovery of the most optimal, information utility-wise, protection strategy in the face of a set of information exposure threats as present in a context. In its decision making process, the algorithm is supported by the, so called, *Levels of Exposure* model — used for matching the perceived exposure severity to adequate mitigation operations, also introduced in the chapter.

Chapter 5 describes the architecture and the operation of μCASPEr, a specialisation of CASPEr targeted at highly constrained ubiquitous computing devices. μCASPEr rests on a policy model based on a variant of Finite State Automata. We outline the policy model and show how it is leveraged in μCASPEr to meet the design objectives. The chapter opens with a high-level overview of system architecture to support CASPEr, which is subsequently built upon. We conclude with a brief description of the requirements and an outline architecture of a data management model for CASPEr.

Chapter 6 provides a discussion of CASPEr concepts as well as a theoretical and example-driven evaluation of both CASPEr and μCASPEr. The goal of the evaluation being the definitive deployment feasibility argument. μCASPEr is evaluated in terms of the complexities incurred by the employed policy model. Termination and complexity of the Optimal Cover Discovery algorithm is assessed and its potential generalisations are discussed. A substantial part of the chapter is devoted to the qualitative comparison of CASPEr to the major information security paradigms as overviewed in Chapter 2. The criteria used were chosen to provide for a clear placement of CASPEr within the big picture, thus crystallising the overall, conceptual, contribution of this dissertation.

Chapter 7 concludes the thesis with a brief summary of the work presented and an outline of several directions in which we envisage CASPEr being taken in the future.

Chapter 2

Background

2.1 Chapter Overview

The material that we present in this chapter has a two-fold aim: firstly it describes the wider setting of this thesis and points at the constraints and requirements implied by it and, secondly, it paints the information security big picture to which CASPEr contributes.

We set off with a brief outline of the concept of context and context-aware computing (Section 2.2), relevant from the point of view of the information exposure threat modelling. Next, we present the principles of IBM's vision of autonomic computing (Section 2.3) and explain how they influence the design of CASPEr. We proceed to present the general challenges ubiquitous computing poses for security (Section 2.4), as established and agreed upon by the literature in the area. For each of the challenges mentioned, we briefly state how CASPEr relates to it conceptually. To paint the big picture to which CASPEr contributes we describe the four major information security paradigms: Information Flow Control (Section 2.5), Information Dissemination Control (Section 2.6), Access Control (Section 2.7) and Information Usage Control (Section 2.8). Particular attention is drawn to the types of threats addressed by each of the paradigms — which later helps to demonstrate the contribution of this thesis. Direct comparison of CASPEr with the outlined paradigms is postponed to Chapter 6, after all the contributing concepts have been detailed. For completeness purposes, going below the level of abstraction at which CASPEr is presented, in Section 2.9 we briefly outline the main trends in data protection on mobile computing devices. These, however can be seen as encompassed by CASPEr. Finally, in Section 2.10, we relate the approach to modeling the world based on the notions of *container* and *containment*, as presented in Chapter 3, with the research conducted in the field of mobility theory and geo-information systems.

The chapter does not pretend to be exhaustive but only tries to present the major, and the most influential, contributions in the relevant research areas, setting the scene for the rest of the thesis.

Terminology: Ubiquitous vs. Pervasive Computing

Prior to delving into the chapter we firstly need to clarify a piece of terminology. Ubiquitous computing, the term coined by Mark Weiser, is often also referred to as *pervasive computing* in the literature. However, the letter was defined by Lyytinen and Yoo [LY02] to assume embedded but static computing infrastructure. Thus, the concept of ubiquitous computing [Wei91] could be seen as actually encompassing that of pervasive computing, along with the support for mobile hosts and mobile code. To the contrary, the editor in chief of IEEE Pervasive Computing Magazine declares, in the inaugural issue [Sat02], that the two terms are synonymous. In this thesis we embrace the latter definition and use the two terms interchangeably. In addition, we bind the concept of ubiquitous computing tightly with that of autonomic computing [IBM01] [KC03] and proactive computing [Ten00].

2.2 Context and Context-Aware Computing

Mark Weiser describes ubiquitous computing as inherently *calm* [Wei91]. The computation fades into the background and from there it actively supports user tasks in a non-intrusive fashion. To meet the challenge and provide for effective decision making in face of high degrees of heterogeneity and dynamism, a pervasive computing system has to be *context-aware*. A user's context can be rather rich, comprised of attributes describing state of its physical surroundings (such as location, presence or activity), user's physiological and emotional states, mobility profile, historical behaviour patterns and many more [Sat02].

The notion of context has intuitive connotation in human reasoning. If a human, as opposed to a digital, assistant were given details about the user's context they would use them to make proactive decisions, anticipating the user's needs. In making these decisions, the human assistant would strive to minimise disturbance to the user — otherwise its own role would become meaningless. One of the fundamental questions motivating research in context-awareness is whether and how a pervasive computing system may emulate such a human assistant [Sat02].

The notion of context is far from being exclusive to pervasive computing or tied to a user centric view of the world. It is also leveraged where more general, wider-scale, adaptation to a dynamic environment is sought, such as in e.g. distributed systems or ad-hoc networking. In these cases, the term context is interpreted from the point of view of entities affected by the state of their environment, such as system processes, applications or pieces of information. For example, a context of a computer process (such as a mobile agent) can, in addition to accounting for the state of physical surrounding of a platform it executes on, incorporate any relevant state of its computing environment, such as resource loads, service availability or authorisations of respective users etc.

2.2.1 Defining Context

The first step in outlining the semantics of context-aware computing is defining the notion of context itself. Broadly speaking, the concept of "context" has a long history in literature, philosophy, artificial intelligence and linguistics [Mos04]. In the field of pervasive computing the notion of context is often interpreted in an application specific manner. Due to its heavy exploitation in different domains the term context enjoys myriad definitions. Consequently, it remains a general word with a vague meaning.

Context is usually defined in terms of examples, such as used above, or by leveraging recursive terms such as *state of environment*, *surrounding* or *situation*. While the former are hard to generalise, the latter offer no insight into the relevant contextual elements. For long, researchers have been struggling to provide a precise, unique, definition of context [Sch95a] [SAW94] [SBG99] [Pas98] [Dey01] [ADB$^+$99]. However, the efforts have resulted in little success. It is the general lack of precise, mathematical formalisms, that cause the wide confusion of the meaning of the term itself. On the other hand, they are next to impossible to define due to the exploitation of the concept across a wide body of research areas and application scenarios, even confined solely to pervasive computing, with differing semantics. To pinpoint the meaning of context as used in this thesis, we briefly outline a selection of relevant definitions in a chronological order.

Schilit et al. [SAW94] define context by dividing it into three aspects and referring to each through a set of examples. The aspects are: *where you are*, *who you are with* and *what resources are nearby*. Schilit et al. clearly state that the first category implies not only location but a wider physical context such as lighting, noise level, network connectivity etc. A common criticism of the definition is the lack of time related aspect [CK00].

In [ADB$^+$99] Abowd et al. provide a thorough analysis of context and context-awareness and contribute, perhaps the most popular, definition of context:

> Context is any information that can be used to characterise the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between the user and an application, including the user and the application themselves.

We can see how the definition is user centric, focusing on interaction between user and application. Abowd et al. [ADB$^+$99] also distinguish between *primary* and *secondary* types of context. The former are: *identity*, *location*, *activity* and *time*. All other types of context are declared as *secondary* as they are typically associated with at least one of the primary types. Having mentioned the location, we acknowledge that it is possibly the most extensively practically exploited type of contextual information [HSK04]. It is agreed in the literature that the pioneering work on location-awareness was the ActiveBat ultrasonic location system, presented in [HHS$^+$99].

Chen and Kotz [CK00] distinguish between contextual information that requires a change in behaviour of mobile computing applications as opposed to contextual information users should be only notified of directly. Accordingly, they provide for a suitable definition of context:

> Context is the set of environmental states and settings that either determines an application's behaviour or in which an application event occurs and is interesting to the user.

Depending on the point of view at which context is considered various authors have proposed a number of approaches to categorisation of contextual information. In analogy to the definitions, the categorisations themselves remain diverse, vague and ambiguous — thus, open to interpretation. Different suggestions for context categorisation are nicely summarised in [MPR04].

In this work we embrace the definition of context as offered by Abowd et al. [ADB$^+$99] and stated above. As we create an information centric view of the world, modelling it from the point of view of data objects[1] (Section 3) we interpret the term *entity*, as used in the above definition, to be a piece of information. The *situation* of an entity is then used to denote a set of information exposure threats as experienced by a piece of information in an environment. However, we do not consider the context as being relevant only to the quality of interaction between user and application — we see it as determining wider application behaviour, as in the context definition by Chen and Kotz [CK00].

2.2.2 Context-Aware Computing

In order to be used, contextual information has to be *modelled*. Context models determine internal representation of contextual information as well as its level of abstraction. Different approaches to the former are nicely summarised in [SLP04]. Widely accepted structuring and modelling of contextual information has been proposed by Henricksen et al. in [HIR02] and [HI04].

Context-awareness deals with the problem of ways in which abundant contextual information can be leveraged effectively by applications. The term context awareness has been first introduced by Schilit et al. in [SAW94]. They identify four types of context aware applications, along two orthogonal axes, depending on whether the adaptation task gets *information* or executes a *command* and whether the task is triggered *automatically* or *manually*. CASPEr would thus be classified as obtaining information and providing for automatic adaptation.

Pascoe [Pas98] suggests a taxonomy of context aware features as: contextual sensing, contextual adaptation, contextual resource discovery and contextual augmentation. According to this taxonomy, CASPEr represents a contextual adaptation application.

Abowd et al. [ADB$^+$99] combine the above two classifications and define three categories of context-awareness features applications may support as: presentation of information and

[1] A generic term used to refer to a collection of data.

services to a user, automatic execution of a service and tagging context (*context augmentation*) with information for subsequent retrieval. CASPEr falls into the second category.

Finally, Chen and Kotz [CK00] distinguish between *active* and *passive* context awareness. The former refers to applications that automatically adapt their behaviour to contextual changes while the letter denotes applications that only relay information about context to users. CASPEr qualifies largely as active context awareness application, however, in order to mitigate a threat it is sometimes sufficient just to make user aware of its presence. Thus, we see CASPEr as actually crossing the boundary between active and passive context awareness. In other words, CASPEr lies on the boundary between seamless and seamful approaches to context-aware adaptation design [MG04] [Cha03] [Cha02].

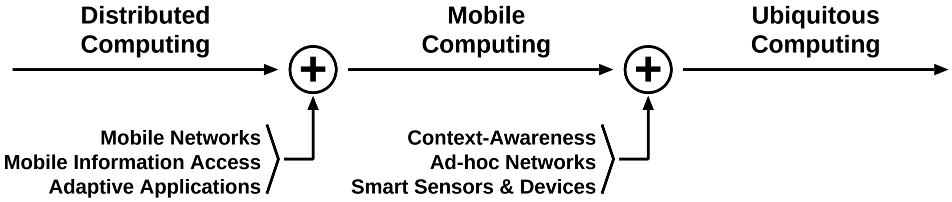


Figure 2.1: Evolution chain.

Figure 2.1 [SLP04] shows the placement of context-awareness, and other relevant factors, in the evolution chain leading from distributed to ubiquitous computing via mobile computing. In general, when we refer to the term context-awareness in this thesis, we assume the features as defined by Pascoe [Pas98]. Context sensing and modelling techniques, mechanisms and tools are outside the scope of this thesis. We assume their existence for the context-adaptation, i.e. information exposure threat mitigation, purposes as required by the material presented in this thesis.

2.3 Autonomic Computing

Prompted by the growing complexity of computer systems and software in general, seen as the main obstacle to further progress of the IT industry, in late 2001 IBM released a manifesto in which they introduce *Autonomic Computing* as a visionary approach to addressing the issue. As pervasive computing inherently extends complexities present in the traditional systems we see the IBM's vision of the solution as highly applicable. In fact, we believe that the notion of pervasive computing itself should encompass autonomic computing to a large extent.

The concept of autonomic computing describes computing systems that can manage themselves independently of infrastructure or service availability given high-level objectives to accomplish. The notion of autonomy in decision making and enforcement is particularly important from the point of view of security in highly dynamic and heterogeneous environments such as ubiquitous computing. From the point of view of CASPEr, to be able to guarantee continuous information exposure protection, its design must facilitate autonomous operation.

IBM specifies the concept of autonomic computing through eight distinct principles. We briefly overview the principles and state how they apply to CASPEr where applicable. As the CASPEr concepts referred to below are not yet meaningful, we also provide pointers to the corresponding explanatory chapters of the thesis. The eight principles of autonomic computing are:

1. *An autonomic system needs to "know itself". An autonomic system will need detailed knowledge of its components, current status, ultimate capacity, and all connections to other systems to govern itself.* Through its *containment*-based model of the world CASPEr facilitates continuous spatial and temporal tracking of sensitive pieces of information within a system (Chapter 3). In other words, it "knows itself".

2. *An autonomic system must configure and reconfigure itself under, often unpredictably, varying conditions.* Through its containment based model of the world, CASPEr dynamically reflects relevant changes in environment and it re-evaluates its decision upon any condition changes. CASPEr is modular and allows for dynamic, run-time, reconfigurations and addition of threat mitigation components and strategies (Chapter 5).

3. *An autonomic system never settles for the status quo — it always looks for ways to optimise itself.* CASPEr decisions are dynamically re-evaluated to best balance the tradeoff between *information utility*, system usability and the level of protection provided (Chapter 4).

4. *An autonomic system must be self-healing — it must be able to recover from events that may cause parts of it to malfunction.* This principle is not directly applicable to CASPEr but is to the platforms on which CASPEr is deployed. Having said this, CASPEr degrades gracefully when the richness or completeness of the relevant containment-based model of the world or availability of *threat mitigation operations* is decreased (Chapter 3).

5. *An autonomic system must be an expert in self protection.* CASPEr contributes to a target platform meeting this principle.

6. *An autonomic system must be context adaptive.* Context is leveraged in CASPEr for information exposure threat modelling which, when required, triggers an appropriate adaptation process.

7. *An autonomic system must implement open standards not to be isolated in a heterogeneous environment.* CASPEr is platform localised model and it does not currently

engage in systematic interaction with services or other entities in the environment. Interaction for the purposes of context sensing is considered out of the scope of CASPEr and is handled by dedicated components of the target platform.

8. *An autonomic system will marshal IT resources to shrink the gap between the business and personal goals of users in an optimal fashion.* One of the fundamental benefits of CASPEr is that it explicitly considers the tradeoff between *information utility* and system usability versus the level of provided protection in its decision making process (Chapter 4).

In other words, not only is CASPEr is designed to be autonomous in itself (principles 1, 2, (3) and 6 above) but it also contributes to the overall autonomy of a target pervasive computing platform (principles 5, 6 and 8 above). Note how certain aspects of CASPEr apply to more than one autonomic computing principle. This is so due to rather blurry boundaries in the statement of the principles of autonomic computing themselves.

2.4 Ubiquitous Computing and Security

The motivation for information security in pervasive computing environments is equivalent to the motivation in all other computing systems. It can be summarised by stating that information must not be obtained, modified or access to it denied in an unauthorised fashion. However, many of the assumptions and usage scenarios underlying classical security concepts simply do not hold for ubiquitous computing [TS04a] [Sta02] [SC03] [NWET04]. At the same time, security is one of the unavoidable hurdles on the path to realisation and wide embracement of the holistic vision of disappearing computer and omnipresent information.

In [Sta02], Stajano defines security as a complex process that encompasses assessing threats (bad things that may happen), vulnerabilities (weaknesses in systems and their defences) and attacks (threat actualisations), estimating likelihoods for attacks given vulnerabilities, estimating attack costs, developing safeguards and countermeasures and applying them in the most optimal manner. Inherent characteristics of pervasive computing redefine the way we think about every single aspect of the above definition.

2.4.1 The Challenges

Realising that pervasive computing systems must provide adequate security as an inherent property, researchers have striven to identify fundamental challenges that should lead the design of security models and solutions for ubiquitous computing, rather than jumping straight to securing particular usage scenarios [TS04a] [SC03]. The more so as it is profoundly hard, perhaps impossible, to predict all emerging shapes of ubiquitous computing, as Mark Weiser himself candidly observed:

> Neither an explication of the principles of ubiquitous computing nor a list of the technologies involved really gives a sense of what it would be like to live in a world full of invisible widgets. To extrapolate from today's rudimentary fragments of embodied virtuality resembles an attempt to predict the publication of Finnegan's Wake after just having invented writing on clay tablets. Nevertheless the effort is probably worthwhile.

Therefore, to sense what there is for security in ubiquitous computing, researchers try to extrapolate from the fundamental characteristics of the vision, mostly through comparison with and from the experience gained from more traditional computing environments, clinging onto limited set of practical socio-technological insights provided, so far, by mobile computing. In [TS04a] Thomas and Sandhu explore the challenges and research directions in building models, protocols and architectures to support security in pervasive computing environments. They identify seven fundamental challenges as:

1. *The need to integrate socio-technical perspective.* If pervasive computing is to be intertwined into our daily activities issues related to usability of and confidence in security, as well as how deployed security models and solutions fit into wider sociological, cognitive, economic and legal perspectives need to be considered. CASPEr facilitates meeting this challenge through accounting for the subjective, *information utility* impact of the information exposure mitigation process (Chapter 4).

2. *Breakdown of classical perimeter security and the need to support dynamic trust relationship.* This challenge, and the role of CASPEr in addressing it, was outlined in the thesis introduction.

3. *Balancing non-intrusiveness and security strength.* Pervasive computing puts into spotlight the tension between usability and security. Security relevant information has to be sensed from context and utilised minimising user's involvement. CASPEr tries to minimise intrusiveness by leveraging standard data management and handling procedures for information exposure mitigation (Chapter 4).

4. *Context awareness.* As stated above, the ability to utilise contextual information for threat model establishment and mitigation is crucial for making security more proactive and less intrusive in pervasive computing environments. CASPEr leverages contextual information for information exposure threat correlation (Chapter 3).

5. *Mobility, dynamism and optimality.* In a pervasive computing system users migrate freely through environments and so does data. No guarantees on connectivity or service availability can be provided in general. Threat models change unpredictably. Security models, protocols and architectures must be highly adaptive. The core aim of CASPEr is to be able to adapt to dynamic changes in information exposure threat models in an autonomic fashion.

6. *Resource constrained operations.* Irrespective of the resource poverty of a target platform can security of a piece of information be compromised. Resource poverty envisaged for some pervasive computing platforms poses serious constraints on security operations, protocols and mechanisms that can be deployed. Although CASPEr is not expected to incur substantial resource overheads (Chapter 6), in Chapter 5 we present μCASPEr, a policy-based instantiation of CASPEr for highly constrained pervasive computing platforms.

7. *Balancing security and other service tradeoffs.* For the vision of pervasive computing to take off it has to offer the right set of tradeoffs between attributes like security, usability, quality-of-service and cost for every application area. Furthermore, as pervasive computing systems tend to have highly personal components, these tradeoffs must be tunable. CASPEr explicitly accounts for a set of criteria, referred to as *action impact*, denoting system wide side effects in the threat mitigation process (Chapter 4).

We do not fully agree with the challenge 3 above as stated by Thomas and Sandhu. Security decisions often make systems behave in a way which, even if fine tuned, is not exactly as desired or expected by users. If the decisions account for the dynamic state of environment they may cause system to behave in a manner which seems inconsistent over time. In some cases, inability of users to explain the inconsistencies in system's decision making may result in a substantial drop in their confidence in the overall system reliability. Thus, we advocate a certain degree of seamful design [MG04] [Cha03], necessary for user awareness of the motivation behind certain decisions made.

In another effort [SC03], Stajano and Crowcroft identify, at a higher level of abstraction and specificness, four challenges for security in pervasive computing environments as: *control* (over the invisible, proactive and autonomic systems), *ownership* (of omnipresent embedded components and data), *privacy* (issues raised by the volume and permanence of private information collected at every step in an unnoticeable manner) and *mistrust* (in the behaviour of the “hidden” computers).

2.4.2 The Security Properties

A universally accepted taxonomy divides computer security threats based on whether they affect confidentiality, integrity or availability. We briefly review the three from the point of view of pervasive computing.

Confidentiality

Confidentiality is a property that is violated if information is revealed to unauthorised principals[2]. Threat to confidentiality is known as *disclosure*.

[2] Principal is used to denote entity in its most abstract sense (people, agents, devices...).

The opinion that prevails in literature, [Sta02], is that information confidentiality in pervasive computing is stressed due to mainly two factors: i) the ease of passive eavesdropping on wireless communications technologies, and ii) the amount of information of sensitive nature contained within pervasive computing devices and infrastructure components. The severe consequences of the latter, confined to mobile computing, have been hinted at in Chapter 1. The prospect of automatic collection, retention and use of sensitive information by "invisible" pervasive computing systems and infrastructure has fuelled research in the areas of trust and privacy preserving technologies. A comprehensive and up to date overview of research in the area of trust in ubiquitous computing environments is provided in [NWET04].

Communications confidentiality is typically ensured through encryption[3]. It is commonly stated in the literature that standard cryptographic primitives are too costly to be performed on mobile computing devices. However, suitable cryptographic primitives have been devised for deployment on highly resource constrained devices — *peanut processors* [Sta02] [And01], such as sensor platforms. Furthermore, the popular and ubiquitous mobile computing devices, like mobile phones and PDAs, have been shown to sustain even the most expensive public key operations [AVTO03] nowadays.

CASPEr focuses almost exclusively on information confidentiality protection as threatened by information exposure. Information exposure threats point at many more, perhaps not directly obvious, instances in which information confidentiality can be compromised in pervasive computing environments beyond eavesdropping on wireless channels or pervasive computing platform compromise. One of the benefits of CASPEr, for its deployment in environments characterised by wide heterogeneity, is that it leverages a plethora of standard information handling and management procedures, subject to availability and depending on a context, to ensure information confidentiality rather than being tied to availability of a single protection mechanism (e.g. encryption).

Integrity

Integrity is a property that is violated if information is modified in an unauthorised way[4]. Threat to integrity can be dubbed *corruption* [Sta02].

As with respect to communications integrity, cryptographic mechanisms such as digital signatures, chains of hashes [ABC$^+$98] or Message Authentication Codes (MACs) are standardly used to protect information integrity [Sta02]. Considering pervasive computing devices and "invisible" infrastructural components containing large volumes of potentially sensitive information, not only is their own integrity critical but, perhaps more importantly, trust [NWET04] in them not to alter the information they legitimately handle.

In some cases, CASPEr provides for limited integrity protection as a side effect of confidentiality protection, as discussed in Chapter 6.

[3] An excellent textbook on cryptography is [Sch95b].

[4] By a principal not authorised to alter information in the particular way.

Availability

Availability is a property that is violated if a system does not perform its advertised service in a timely fashion upon an authorised request. From a point of view of data objects, and CASPEr, availability is violated when information is not promptly accessible to an authorised user as requested. The threat to availability is called *denial of service*. Gligor [Gli83] proposes the concept of *maximum waiting time* as a criterion for assessing service availability.

The vision of ubiquitous computing poses challenges for availability along a number of axes. Roaming mobile code may abuse host platforms, wireless communications channels are prone to jamming, platforms hosting services allowing for public queries may be abused by sleep deprivation [Sta02], etc. If the vision of ubiquitous computing is to become reality, services provided by it must seem uninterruptible to end users.

We do not discuss availability, or mechanisms ensuring it, any further in this thesis as it is considered out of the scope.

2.5 Information Flow Control

Denning and Denning [DD77] define *information flow control* as regulation of information dissemination [5] among objects in a system. Information flow control is all about information confidentiality protection. An information flow policy consists of a set of information *security classes*, a binary *flow relation* defining legal flows among the classes and a method of *binding* the classes to information storage objects[6]. Any operation, or a series of operations, that uses a value (information content) of some data object, A, to derive a value of another object, B, is said to cause a flow from A to B. For the flow to be legal the given information flow policy needs to allow for it.

While Denning and Denning [DD77] confine information flows to objects within a single system, in CASPEr we consider information flows between objects in a system and objects, or principals, outside the system — in its respective environment, or context. In CASPEr, in order to distinguish legality of information flows, we adopt the definition of information flow policy as specified by Denning and Denning.

2.5.1 Classifications, Clearances and Security Labels

The concepts of *classification*, *clearance* and *security label* as well as formal policy models [And01] originate from the military sector.

[5] We use the term dissemination control in this thesis with slightly different semantics, as introduced by Park et al. in [PSS00] and presented in the next section.

[6] Although Denning and Denning are not explicit in the semantics of an object we assume it to be an "undefined" primitive concept, a passive container for information (Chapter 3).

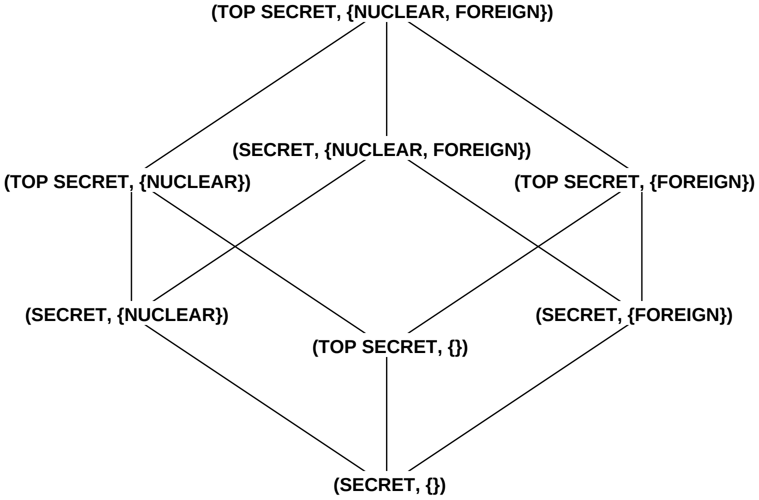


Figure 2.2: Example security label lattice.

In order to quantify sensitivity of documents for purposes of access control as well as handling, in the Second World War and the Cold War era, NATO countries moved to a document marking scheme. The basic scheme consisted of four labels called *classification levels*, namely *Unclassified*, *Confidential*, *Secret* and *Top Secret*, in the order of strictly increasing sensitivity respectively. The basic scheme has since been extended by additional labels both in the US and in the UK — even hindering the data flow between the two jurisdictions [And01].

While classification applies to documents (information containers) the labels may also be associated with principles, originally government employees, to denote the level of trust they have been vetted with. In this case, the labels are called *clearances*. Classifications and clearances play a role in information flow and access control as outlined below and in Section 2.7 respectively.

To be able to implement *need-to-know* principle [And01] a system of codewords was devised to complement security classes. Codewords are used for qualification of information of a particular classification level. A classification level together with a set of codewords forms a *security label*, or a *compartment*. From the point of view of principals, the codewords can be seen as denoting areas of competence — guiding the need-to-know. Figure 2.2 shows an example lattice of security labels — with the codewords enclosed within the curly braces.

2.5.2 The Foundations: Bell-LaPadula

The seminal work on information flow policies, and more widely, security models, was published in 1973 by David Bell and Leonard LaPadula [BL73]. It is widely know as the Bell-LaPadula or *multilevel security* model. The systems that implement it are referred to as *multilevel secure* (MLS) systems. The principle feature of the Bell-LaPadula model is that information is never allowed to flow downward — where the information flow source object is at a higher classification level than the destination object. The Bell-LaPadula model rests on the following two properties:

- The *simple security property*: no process may read data at a higher classification level — also known as *No Read Up* (NRU).
- The $\star -$ *property*: no process may write data at a lower classification level — also known as *No Write Down* (NWD).

Only the active entities in a system may initiate and carry out an information flow. That is why the above properties are stated in terms of processes — programs in execution. A classification level of a process can be considered to be equal to the highest level among the objects it has ever touched. The fundamental innovation of the model is considered to be the $\star -$ property. It was inspired by the prospect of malicious code, such as “trojan horses”, leaking information to objects of lower classifications.

The original Bell-LaPadula suffered ample criticism regarding issues such as covert channels, composability and the cascade problem and polyinstantiation — nicely summarised by Anderson in [And01]. Perhaps most notably, Bell-LaPadula is silent about controlling object creation and destruction in a system. Furthermore, the practise has shown it to be too rigid to model information flows in realistic organisations as it does not account for the naturally occurring information *declassification*. In other words, it forbids the natural feedback between parts of an organisation if it involves information of differing classification levels.

A significant advantage of the model, however, is that it can be expressed in terms of a simple mathematical formalism that facilitates formal proofs of security of a given system. This is often lost in more expressive and complex security models, such as e.g. Harrison-Ruzzo-Ullman (HRU) model (the access matrix) [HRU75] [HRU76]. Despite its many criticisms, the Bell-LaPadula model is considered one of the biggest contributions in the information security field.

2.5.3 MLS System Examples

Multilevel security has been implemented in a number of ways, with or without hardware support, as operating system add-ons or with specialised operating system kernels, as application wrappers or with application awareness. They all, however, rest on the existence of a *reference-monitor* — a mediator component that can be verified to ensure security policy enforcement, akin trusted computing base [Gro] nowadays.

In 1983 Honeywell launched the *Secure Communications Processor* (SCOMP), a derivative of Multics [Fra83]. It had formally verified hardware and software and was extensively used in the US government sector. SCOMP served as a model for the development of the

Trusted Computer Systems Evaluation Criteria [oD85], also known as the *Orange Book*. This was followed by *Blacker* [BB94] — a series of encryption devices forming a MLS system, *NRL-Pump* [KM93] — a one way data transfer device, and a series of related products [And01].

Majority of the available MLS systems are derivations of Unix. Examples are AT&T System V/MLS [Amo94], Compartmented Mode Workstations [Hub94] [CFG$^+$87] [BPWC90], trusted X Window [EMO$^+$94] etc. In a need to allow users to run standardised applications to process sensitive information, and still provide for the MLS properties, Purple Penelope [PW98] was developed by Britain's Defence Evaluation and Research Agency, as a Windows NT MLS wrapper.

2.5.4 The Lattice Model Formalisation and Extensions

The seminal work on systematisation of formalisation of the lattice model of secure information flow was done by Denning in 1976 [Den76]. Since then the lattice model has become highly popular for computer security models. An example lattice has been shown in Figure 2.2.

In [DD77] Denning and Denning utilise the lattice model in their, yet another, seminal work on static validation of information flows in programs. They divide information flow control into run-time, where a dedicated system component (a monitor) enforces the policy through monitoring read and write operations of processes (examples from the above section), and static, more complex, involving analysis of program code and structures to determine flows between input and output objects. The latter rests on extensive programming language analysis and compiler support.

Realising the limitations imposed by absence of declassification methods in previous information flow control models, Myers and Liskov develop a new model targeted at highly decentralised systems [ML97]. The crucial difference to the previous work, e.g. by Denning and Denning [DD77], is that the Myers-Liskov model allows individual users to explicitly declassify (i.e. downgrade) data they own at their own discretion. As such, the model focuses on providing security guarantees to individual users rather than a monolithic organisation. The model provides for both static program analysis and run-time information flow checking for more dynamic entities, such as file systems.

An important contribution to static information flow control was offered by Myers in [Mye99]. Myers develops JFlow, an extension to Java adding statically checked annotations, and supporting features previously never integrated with information flow control, such as objects, sub-classing, dynamic type tests, exceptions etc.

Static program checking is not applicable for analysing information flows implied in information exposure, as defined by CASPEr, due to their dynamic nature — i.e. they do not involve solely program entities identifiable through code analysis of any sort but entities that may, or may not, be present in the program's execution environment (context).

2.6 Information Dissemination Control

Information Dissemination Control (DCON) is targeted at *controlling* and tracking information after it has been delivered to a legitimate recipient in the form of a digital object. The notion of control encompasses the usage of, including the access to, the digital object as well as its further dissemination, through e.g. super-distribution[7]. The trust model assumed in DCON is that the digital object's receiving end can never be trusted. In other words, that it is not possible to separate honest and dishonest users of the digital content. In CASPEr, on the other hand, we assume information custodians can be trusted not to willingly leak sensitive information. For CASPEr, this relaxes the need for formal verification of system components, hardware and software, as well as their tamper-proofness, as required in some DCON usage scenarios (outlined below).

2.6.1 Shapes of DCON

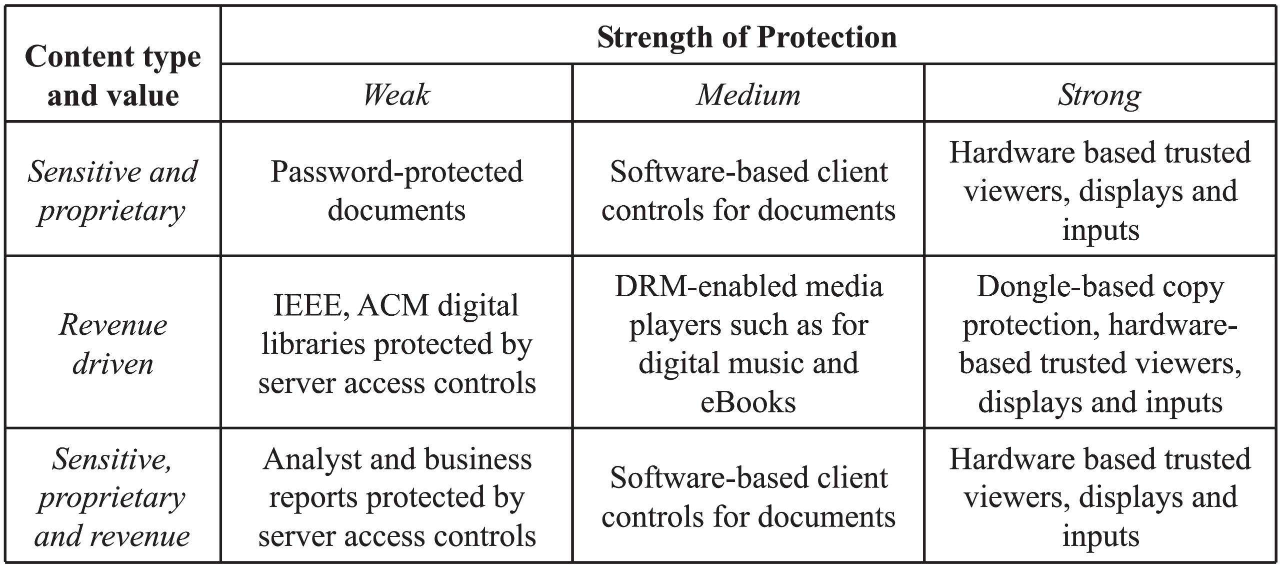
DCON comes in many different shapes. Most notable example in the past several years is Digital Rights Management (DRM) — driven largely by the entertainment and aimed at ensuring revenues from copyrighted digital content distribution. The seminal work on systematisation of DCON comes from Park et al. [PSS00], taken forward by Thomas and Sandhu [TS04b].

In [PSS00] Park et al. identify two main categories of DCON systems as *Payment-Based Type* (PBT) and *Payment-Free Type* (PFT). In PBT, a payment function is required to ensure financial revenue from information dissemination. PBT systems typically characterise the entertainment industry, where they are known as DRM systems. Security breaches in PBT DCON systems result in financial loss to the content provider and copyright holder. In PFT DCON systems, no payment is involved but dissemination must be controlled to meet information security requirements, most notably confidentiality. PFT systems are deployed for handling information of inherently sensitive and/or proprietary nature. Examples are intelligence classified information, business and trade proprietary information as well as private personal information such as medical or bank records. Security breaches in PFT DCON systems may result in much more severe damage to information originator and related parties than solely financial loss[8].

Table in the Figure 2.3 represents a decomposition of the DCON space along two axes, as defined by Thomas and Sandhu in [TS04b]. The vertical axis categorises information by the type of its significance to the originator while the horizontal axis denotes the required strength of protection. The cells specify some example schemes and requirements for the respective DCON systems. The figure shows that a DCON system may require dedicated

[7] An approach to digital content dissemination by encouraging users to pass the content to others.

[8] While super-distribution approach to information dissemination may be tolerable, or even encouraged, in PBT scenarios this is not the case in PFT systems.



Content type and value	Strength of Protection		
	Weak	*Medium*	*Strong*
Sensitive and proprietary	Password-protected documents	Software-based client controls for documents	Hardware based trusted viewers, displays and inputs
Revenue driven	IEEE, ACM digital libraries protected by server access controls	DRM-enabled media players such as for digital music and eBooks	Dongle-based copy protection, hardware-based trusted viewers, displays and inputs
Sensitive, proprietary and revenue	Analyst and business reports protected by server access controls	Software-based client controls for documents	Hardware based trusted viewers, displays and inputs

Figure 2.3: DCON systems categorisation.

software and hardware components, as well as their formal verification, to ensure higher levels of confidence in the protection of more sensitive data.

Information dissemination control had been researched, to an extent, prior to the introduction of the term DCON, [PSS00], under the term *originator control* (ORCON) [AHK^{+}91] [MMN90]. However, no efforts have been made to understand the holistic problem of dissemination control and generalise individual instances as well as supporting system architectures in a systematic fashion.

In pervasive computing research arena, DCON systems have been investigated largely as the enabling technology for information privacy preservation [HZ04] [KK02] [Lan02] [MPB03]. From the industry perspective, Mobile Phone Work Group of the *Trusted Computing Group* (TCG) [Gro] focuses on enhancement of TCG concepts as needed for adoption in the mobile computing arena.

2.6.2 DCON Architectures

In [PSS00], Park et al. generalise and systematise security architectures for controlled information dissemination by means of a taxonomy. The taxonomy is established based on three factors: the presence of a *Virtual Machine* (VM), the deployed *Control Set* (CS) type and the digital content distribution style. Figure 2.4 shows a diagrammatic representation of the taxonomy — identifying the 8 distinct DCON architectures.

Park and Sandhu [PS02a] define VM as a trusted, tamper resistant, recipient-side ap-

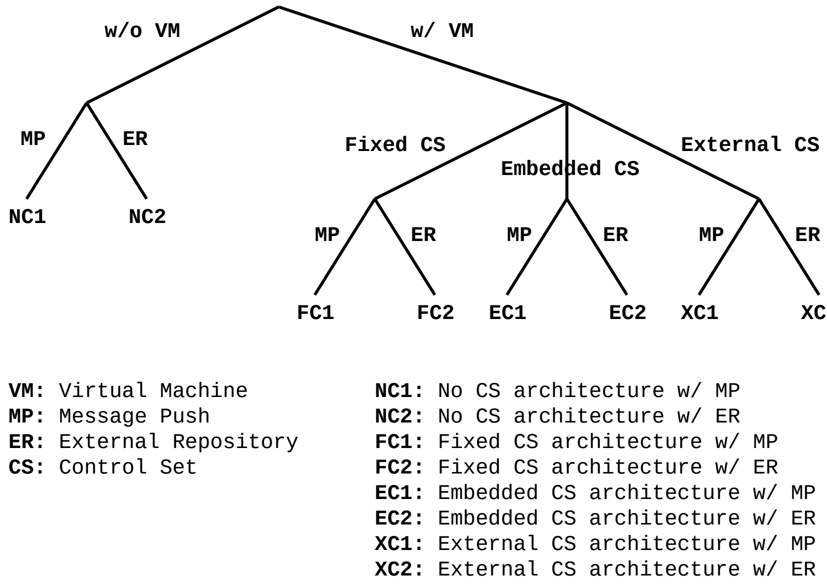

w/o VM
w/ VM
MP
ER
NC1
NC2
Fixed CS
Embedded CS
External CS
MP
ER
MP
ER
MP
ER
FC1
FC2
EC1
EC2
XC1
XC2
VM: Virtual Machine
MP: Message Push
ER: External Repository
CS: Control Set
NC1: No CS architecture w/ MP
NC2: No CS architecture w/ ER
FC1: Fixed CS architecture w/ MP
FC2: Fixed CS architecture w/ ER
EC1: Embedded CS architecture w/ MP
EC2: Embedded CS architecture w/ ER
XC1: External CS architecture w/ MP
XC2: External CS architecture w/ ER

Figure 2.4: DCON Architectures Taxonomy.

plication software and/or hardware[9] component that runs either standalone or on top of a vulnerable computing environment and employs control functions to provide the means to control and manage access and usage of digital information. Trusted hardware in conjunction with attested[10] software layer forms the *Trusted Computing Base* (TCB) [Gro]. Effectively, the VM performs the role of a reference monitor [And72] of a trusted computing base. Systems that do not have a VM cannot perform explicit control of digital content usage (NC1, NC2 in Figure 2.4). They usually rely on legislative deterrents to prevent illegitimate re-dissemination of digital information. The trust model assumed for CASPEr, bearing confidence in custodians not to compromise information confidentiality willingly and/or knowingly, eliminates the need for a full VM-like reference monitor and stringent requirements on the formal verification of correctness of its components.

CSs encapsulate rules governing the legitimate use of disseminated digital objects as enforced by a VM. In the DRM terminology, CS represents the licence. The taxonomy distinguishes three types of control sets: *fixed*, *embedded* and *external*. Fixed control set is hardwired into a VM and applies equally to all digital objects processed by the VM. Most notable example of the fixed CS is the DVD encryption system (DVD-CSS) [Ass]. Embedded control set is tightly, indivisibly, bound to each digital object. This binding is accomplished by cryptographic means. External control set is distributed by a *control centre* as a separate object to the digital content it applies to. Access to the digital content is,

[9] For example, *Trusted Platform Module* (TPM) [Gro05].

[10] The term usually used for chained verification of integrity of software platform running on trusted hardware.

however, impossible without the presence of the CS.

Park et al. [PSS00] identify *message push* (MP) and *external repository* (ER) as two possible dissemination styles. In MP, digital content is delivered to the recipient directly. In ER, the recipient obtains digital content from a dedicated server. ER allows for content distributor to forbid local storage of digital content on the recipient's side. External CS can be used to force the user to connect to control centre on every access to digital content or periodically. In conjunction, the ER and external CS, provide for the maximum level of control of digital content use and flexibility of permission revocation.

DCON Architectures and Pervasive Computing

Most of existing DCON, especially DRM, solutions require extensive support from the infrastructure and target platforms — for which the vision of pervasive computing poses a number of challenges.

In all of the architectures, apart from NC1 and NC2, *digital container* [SBW95] is used to ensure tamper-resistance of distributed digital content and CS. Digital container is effectively a cryptographic wrapper, an electronic envelope, for the distributed contents ensuring control of access to it and its usage. The complexity of the cryptographic mechanisms involved in a digital container may present a problem in their deployment for resource deprived ubiquitous computing devices. Similar considerations apply to the use of VMs. To what extent can we expect the standardisation of their functionality and the prevalence of trusted computing enabled devices at a reasonable cost?

Both embedded and external control sets offer the flexibility of per digital object, per recipient tailoring and can also be combined with a fixed CS. For the access to and the usage of digital contents to be bound to a single recipient the CS usually embeds unique identifier of the target VM. However, the approach binds the digital content to the target platform rather than an actual authorised user. This somewhat contradicts the pervasive computing vision where individual users are not assigned ownership of any particular device and expect their data and choices to move with them right across myriad of, invisible or otherwise, devices.

In pervasive environments, characterised by unpredictable service quality and availability, forcing the client to connect frequently to the ER or the control centre in order to be able to use digital content may have severe consequences on the usability front. Furthermore, repeated downloads of digital content, if its local storage is forbidden, may be infeasible resource-wise. Therefore, one of the primary considerations in designing DCON, and especially DRM, systems for ubiquitous computing environments needs to be the trade-off between digital content availability and its usage control effectiveness and flexibility.

2.7 Access Control

Anderson [And01] describes *Access Control* as a traditional centre of gravity of computer security. Access control is the process of mediating requests to access resources within a system through determining whether to grant or deny them based on a set of criteria. The definition of the rules and criteria according to which access is to be mediated is referred to as *access control policy*. An *access control mechanism* represents the implementation of the access control functions as imposed by the access control policy. To be effective, an access control mechanism must work as a reference monitor [And72].

Access control ensures that only *authorised* access requests made by *subjects* are allowed be carried out on *objects* in a system. The concept of a subject is often confused to that of a *user* [San93]. Every human being known to a system represents a unique user. A subject, on the other hand, is a process in a system running on behalf of a user. While every subject is associated with a single user, each user may have many subjects within a system at any point in time. Each of the subjects may acquire different set of access rights, up to the maximum as assigned to the user by a wider security policy. Objects represent resources and entities in a system. Thus, all subjects are also objects while the reverse does not hold in the general case.

The access control paradigm aims at controlling immediate access to objects without taking into account information flow paths among objects implied by an outstanding collection of authorisations. Thus, it is inadequate for addressing general information flow issues as presented in Section 2.5 above. Often, information flow control mechanisms are designed to complement access control [Den76] [Sto81]. With respect to this, CASPEr is conceptually closer to the information flow control paradigm as it effectively attempts to address flows of information from objects to non-system entities in their environment — with no reference to subjects making explicit requests for access to objects.

A taxonomy that has established itself in the literature groups access control models into three classes: *Discretionary Access Control* (DAC), *Mandatory Access Control* (MAC) and *Role-Based Access Control* (RBAC). In this section we outline the main characteristics of the three classes as well as some of their extensions.

2.7.1 Discretionary Access Control

Traditionally, discretionary access control policies enforce access on the basis of the identity of a user on whose behalf the access request is made (by subjects) and a set of explicit rules that specify the access request types to be granted to known users. The term discretionary is owing to the ability of users to grant access rights they have on objects to other users at their own discretion. Note that in the context of DAC, the semantics of a user and a subject converge from the perspective of authorisations. In other words, DAC models typically do not provide for restriction of authorisations assigned to a subject relative to the authorisations available to the user.

The Access Matrix

The subject-object access rights of a discretionary access control policy are conceptualised in the form of an *access matrix*. The concept of the access matrix was first proposed by Lampson [Lam74] and subsequently formalised by Harrison, Ruzzo, and Ullman[11] in [HRU75] and [HRU76]. The latter work offers an extensive analysis of the complexity of determining an access control policy for Lampson's access control model. Harrison et al. [HRU76] have also shown that it is undecidable whether an access right can leak among processes where access matrix model is leveraged. This is somewhat addressed by Sandhu in [San92], but remains a substantial criticism of the model.

The name of the model proposed by Lampson [Lam74] comes from the data structure — a matrix, used for representation of authorisations that hold in a system at a time. The access matrix model is defined by a triple (S, O, A), where S is the set of subjects, O is the set of objects — on which subjects may exercise privileges, and A is the access matrix. The rows of A correspond to subjects ($s \in S$), the columns correspond to objects ($o \in O$) and each cell $A[s, o]$ contains the authorisations of s on o. For example, $\texttt{write} \in [s, o]$ authorises s to write o. Only the operations authorised by access matrix may be performed. The semantics of the term discretionary imply that if subject s is the owner of object o (i.e. $\texttt{own} \in [s, o]$) then s can modify the permissions for accessing o. In other words s can modify all the cells in the access matrix column corresponding to o. Therefore, the access control matrix is a dynamic entity.

The access matrix model does not say anything about types of objects or operation permissions. Thus, the model can accommodate different access control settings.

Access Matrix Implementation

Although the matrix structure is good for conceptualisation of authorisations in an access control policy it is of little practical value when it comes to implementation. For any system of realistic proportions the access matrix will be of a prohibitive size, and typically sparsely utilised. There are three practically sound approaches to implementing the access matrix proposed in the literature:

- *Authorisation Table.* Authorisation table is a table comprised of non-empty entries of what would be a sparse access matrix. Each tuple in the table corresponds to a single authorisation. This approach is popular in database management systems [CFMS94].

- *Access Control Lists* (ACL). ACLs imply dissecting the access matrix column-wise and associating each of the columns to the respective object. Each object is, thus, associated with a list of all authorisations applicable to it. The typical example of ACL implementation is in the Unix operating system.

[11] Usually referred to as the HRU model.

- *Capabilities.* Capabilities, as opposed to ACLs, imply dissecting the access matrix row-wise and associating each of the subjects known to the system with the corresponding row. Each subject is, thus, associated with a list of all authorisations it can exercise on objects in the system. An example of a capability system is the EROS [SSF99] operating system. Public key certificates also represent a form of a capability.

Capabilities and ACLs imply almost opposing sets of benefits and drawbacks for the authorisation control and management — such as permission delegation and revocation. Furthermore, ACLs are suited for systems with central authorities while capabilities facilitate distribution. Anderson [And01] offers an in-depth treatment of the two approaches.

Extending Basic Authorisations

Although the access matrix model remains the basic concept for expressing a discretionary access control policy, a number of extensions have been developed since its inception. Most notably, the support for *conditions* on authorisations and the concept of *groups* have imposed themselves as standard features.

Conditions are introduced to constrain the validity of authorisations. For illustration, conditions may take the form of system and context state predicates, they may be expressed in terms of the content of an object subjected to access request [Dat03] or they may be dependent on the relevant authorisation and execution history [EAC98] [AF03].

To reduce administrative and management complexities the concept of groups has been introduced. Grouping is applicable to all elements of an access control triple — the user, the object and the action. Grouping reflects natural structure of entities and their authorisation relationships. While the role of the user and object groups are obvious, grouping actions may be used to reflect privileges [SD92] or prioritisations and implications thereof [RBKW91].

Further evolution of the access matrix model has gone in application and data model specific directions. For example, a number of approaches to access control in object oriented systems exploit the concept of encapsulation to provide for partial authorisation on objects, as in [RSC92] and [Mica].

2.7.2 Mandatory Access Control

Mandatory access control policies mediate access requests based on rules and regulations mandated strictly by a central authority. Unlike DAC, MAC typically allows for no permission passing among users solely at their own discretion. The most widely accepted form of mandatory access control policy is the multilevel security policy (MLS), already mentioned in Section 2.5, and based on classification of subjects and objects in a system by assigning them security labels. The exact semantics of the subject and object classification within a system employing a MAC policy depends on whether the policy is aimed at information confidentiality or integrity protection, or both.

Discretionary and mandatory access control policies are by no means mutually exclusive. Rather, they can be applied effectively jointly. In this case, an access request is granted if both of the following hold: i) there exists an (discretionary) authorisation for it, and ii) it satisfies the mandatory policy. Intuitively, the discretionary and the mandatory policies can be seen as representing layered filters, or mediators, for access requests. To be authorised, a request needs to pass through both the filters.

Confidentiality-Based MAC Policies

A confidentiality-based MAC policy aims at controlling direct and indirect flows of information among objects, on one side, and subjects, on the other side, in order to prevent information leakage. The more general concept of information flow control outlined in Section 2.5 focuses on flows of information solely among objects in a system. Subjects may be interpreted as objects themselves or may be seen as the entities initiating and carrying out the information flow.

Confidentiality-based MAC policies rest on the Bell-LaPadula multilevel security model, first introduced in [BL73], and also outlined previously in Section 2.5. The two properties of the Bell-LaPadula model, can be rephrased to reflect the goals of a MAC policy as follows:

- *NRU*: A subject is granted a read access to an object *iff* the subject's clearance dominates the class of the object.

- *NWD*: A subject is granted a write access to an object *iff* the subject's clearance is dominated by the class of the object.

In any real system the operations available are likely to go beyond read and write (e.g. create, delete, append etc.). However, the two suffice to illustrate the point of the model as the other operations can typically be reduced to read and write for the purpose. Interestingly, the above rules allow an *Unclassified* user to write a *Secret* object, possibly damaging its contents. Thus, the NWD property is sometimes modified to require the security labels of a subject and a object to be equal for a write to be authorised.

A common misconception in a wide body of literature is the reference to the original Bell-LaPadula model [BL73] as the *BLP* model. The BLP model is a MAC policy model, a variant on the Bell-LaPadula model, introduced by Sandhu in [San93]. The BLP model is an example of the symbiosis of DAC and MAC policies in which the DAC policy can be seen as bounded by the MAC policy.

Integrity-Based MAC Policies

The MAC policy outlined above is inadequate for protecting integrity. Inspired by the Bell-LaPadula model, Biba [Bib77] proposed a dual model that focuses on information integrity protection. The basic concept in the Biba model is that information should not be allowed to flow to objects of a higher integrity class.

If we represent integrity classes as a lattice, with high integrity placed toward the top of the lattice and low toward the bottom, then the flow of information as authorised by the Biba policy is from the top of the lattice to its bottom. This is directly opposite to the Bell-LaPadula model. Consequently, Biba proposed the following, two property, formulation of the model:

- The *simple integrity property:* a subject is granted a read access to an object *iff* the clearance of the subject is dominated by the class of the object — also known as *No Read Down* (NRD).
- The *integrity $\star -$ property*: a subject is granted a write access to an object *iff* the clearance of the subject dominates the class of the object — also known as *No Write Up* (NWU).

Biba also proposed alternative criteria facilitating greater dynamism of the model. The model is often criticised for its narrow treatment of the integrity problem as a whole.

In [San93], Sandhu shows how the BLP and the Biba model can be combined in systems where both confidentiality and integrity are a concern.

DAC-MAC Hybrid Models

To overcome individual limitations of DAC and MAC models a number of hybrid models have been developed. Perhaps the most notable is the Chinese Wall [BN89] policy model, an attempt to address the issue of commercial discretion with mandatory controls. The goal of the model is to restrict information flow to prevent conflicts of interest. The authorisation decisions are made not on the basis of labels but on what a user has accessed previously.

Focusing on addressing the Trojan Horse issue, [Kar87] and [BF85] propose interposing, between subjects and objects in the system — processes and the file system in particular, a protected layer, a mediator, imposing further constraints on access. Stoughton, in [Sto81], offers a model which marries the information flow model of Denning [Den76] with DAC to derive a hybrid model that he refers to as *access flow* model. In [WOR$^+$74], Walter et al. propose application of mandatory properties in a discretionary context: access control lists are used in place of object classes while subject clearances are cumulative classes of the accessed objects. Set inclusion is used instead of the dominance relationship among security labels. The model has been taken forward by Bertino et al. [BVFS98].

In [San93] Sandhu has shown how a number of mandatory access control policies can be expressed as lattice-based models.

2.7.3 Role-Based Access Control

Role-Based Access Control (RBAC) models [FK92] [SCFY96] represent an alternative to DAC and MAC models which has been gaining increasing attention over the past several years, especially for commercial applications. The principle motivation behind RBAC is the ability to specify, administer and enforce organisation specific security policies in a way that matches the natural structure of the organisation. RBAC models have been widely researched and have gained considerable level of maturity.

In the real world, authorisations individual users assume within an organisation are determined by their *roles* within it. This includes specification of users' organisational duties, their responsibilities as well as qualifications. In general, identity of a user is often of little importance to access control beyond accountability. Based on this observation, RBAC models introduce the concept of *role* as the bearer of authorisations. A role is a semantic construct around which a RBAC access control policy is formulated [FK92].

In the RBAC model permissions are associated with roles, and roles are assigned to users. The logical independence in the specification of user authorisations via roles greatly simplifies the security policy management and administration. Users can easily be assigned to roles and so can roles be granted new permissions. Users may activate and deactivate roles they are assigned to, and in some implementations the individual permissions, at their own discretion, to match the authorisations required to accomplish a task. Each mapping of a single user to a set of active roles is referred to as a *session.* Users may have multiple running sessions in a system at any point in time. While groups in the DAC model define sets of users, the roles define sets of privileges assigned to users.

In [SCFY96], Sandhu et al. recognise the concept of role as only a base for more complex features. They define a family of four conceptual models to serve as a basis of reference:

- $RBAC_0$: the base model with users associated with roles and roles associated with permissions. It is entirely up to the user's discretion as to which roles are activated in a given session.

- $RBAC_1$: RBAC_0 extended by the concept of role hierarchies (RH) as a natural way to reflect an organisation's lines of authority and responsibility.

- $RBAC_2$: RBAC_0 extended by the concept of constraints that control user-role, permission-role and session-role assignments. Activation of roles is not entirely at the user's discretion any more.

- $RBAC_3$: is a model that combines RBAC_1 and RBAC_2.

Figure 2.5 illustrates diagrammatically the features of the four models. US National Institute of Standards and Technology (NIST) has published a standard for RBAC in [FSG$^+$01]. RBAC_2 is also referred to as *parametrised* RBAC. RBAC_2 is of particular interest to pervasive computing applications as it provides a natural way to express authorisation rules that incorporate contextual information.

RBAC model inherently supports three important security principles: *least privilege*, *separation of duties* and *data abstraction* [SCFY96]. Although it offers a number of advantages,

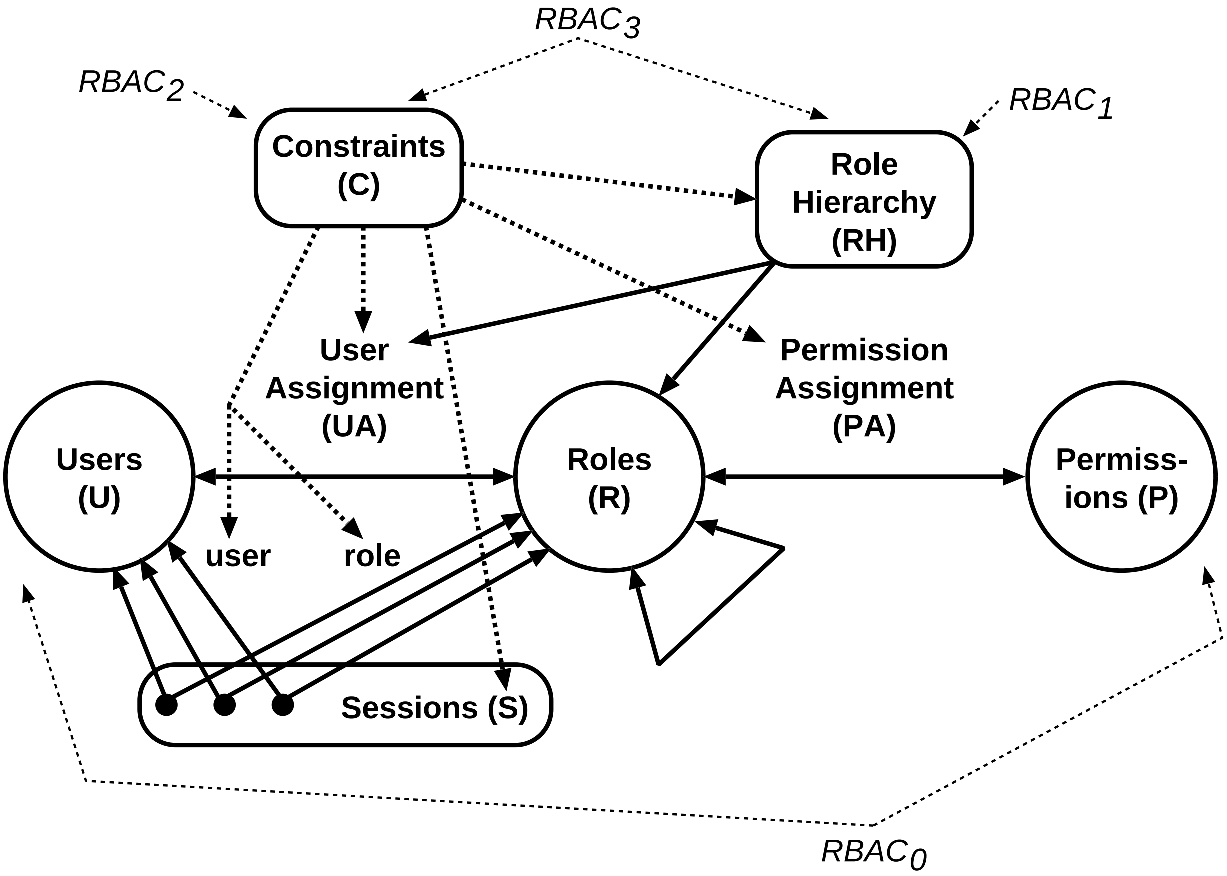


Figure 2.5: RBAC models.

the model cannot be seen as a panacea for all access control issues. One of the main criticisms is the inability to deal with situations in which sequences of operations need to be controlled, as in workflow-based systems. The concept of *active security* is introduced, as outlined below, to address this issue.

In the original RBAC paper, Ferraiolo and Kuhn [FK92] state that RBAC is a form of mandatory access control but not based on multilevel security requirements. However, RBAC models are generally considered policy neutral as they can be configured to enforce both mandatory and discretionary access control policies [OSM00]. RBAC may also be used to complement DAC and/or MAC in a layered access mediation architecture [SCFY96]. In [Bar97], Barkley offers an in-depth comparison between RBAC_0 and ACLs.

2.7.4 Active Access Management

DAC, MAC and RBAC are based on the subject-object view of security. A subject is given access to an object solely based on the rights it possess within a system. In the real world, however, users are often assigned authorisations based on a specific task and activity they are to accomplish. This carries over to distributed computing environments, workflow and transaction management systems with multiple points of access, control and decision making.

Thomas and Sandhu [TS97] introduce the concept of *active security models*, referred to as active access management in the context of access control [BMY02], to denote approaches to security modelling from the perspective of activities and tasks. Preliminary ideas on active security originate from [TS93] and [ST94]. In active security models permissions granting, usage tracking and revocation are coordinated with the progression of tasks automatically. Active security models can, thus, be seen as a move toward the continuity of access mediation, ensuring permissions are assigned to subjects precisely as and when needed for accomplishment of a task.

To accomplish the above, active access control models account for a wider *context*, such as the current state of tasks or workflow, in access mediation. The notion of context as used in this area, [TS97] [BMY02] [BEM03] [Tho97] [GMPT01], denotes the internal state of a system and its components directly relevant for the authorisation process. The term context, as used in pervasive computing and in this thesis (Section 2.2) is a wider concept that encompasses the digital and physical environments of a target entity.

Work on active access management inspired a wider *information usage control* paradigm, outlined in Section 2.8.

2.7.5 Context-Aware Access Control and Architectures

In pervasive computing setting, context in which a user accesses information is highly dynamic and, in general, unpredictable. This is reflected onto the relevant threat models. Consequently, not accounting for contextual information in an access control decision process may put data at risk. Furthermore, the ability to sense and exploit contextual information for the authorisation process may facilitate proactiveness and reduce intrusiveness of a pervasive system as a whole. Thus, a wide body of research concentrates on "enhancing" access control models with context-awareness. Majority of the recent work focuses on extending RBAC models, mainly through parametrisation ($RBAC_2$) or introduction of specialised role types ($RBAC_0$). Note also that authorisation conditions enabled DAC variants may also be utilised for context-aware decision making.

The most accessible, and widely exploited, contextual factors correspond to spatio-temporal information. Bertino et al. [BBF01] and Joshi et al. [JBLG05] propose a RBAC model extension that incorporates temporal, periodic and non-periodic, role enabling and role dependency specification. Extensions of RBAC model by the concept of *spatial role* — a role whose activation is conditioned on spatial position of the requesting user or the target object, seem to be a popular approach to access control spatial context-awareness [HO03a] [HO03b] [BCDP05].

Generalising beyond spatio-temporal information, Covington at el. [CLS$^+$01] [CMA00] propose Generalised-RBAC (GRBAC) model, an extension to RBAC by the concept of *environmental role* — a role that may be used to capture any sort of contextual condition. GRBAC is the most widely cited of the context-aware approaches to access control but is by no means the only one [GMPT01] [SNC02] [CM03] [ZP04]. The former two attempt at addressing some perceived fallacies of GRBAC. Alternatively, OASIS RBAC allows for specifying context constraints in its parametrised model [BMY02].

Context-aware access control models, alike any context-aware application, depend on system architectures which provide for the availability of relevant contextual information. In [CFZA02], Covington et al. propose a context-aware architecture to meet the GRBAC [CMA00] model requirements; [SNC02] is a part of wider GaiaOS [RHR$^+$01] security architecture [VGC01] [AMRCM03]. Myriad other examples exist in the literature. These architectures represent a general model of pervasive systems context-aware security architectures, and CASPEr draws from them (Chapter 5).

2.8 Information Usage Control

The access matrix model of access control has remained fundamentally unchanged since its inception, over three decades ago. The core model has, over time, been elaborated in a number of directions, such as DAC, MAC and RBAC, to meet the need of real-world policies. To address the challenges posed by modern systems, coming from different perspectives, researchers have further offered the concepts of trust management, digital rights management, active access management etc.

Park and Sandhu [PS02b] introduce the concept of *Usage Control* (UCON) as a comprehensive and systematic approach for controls on usage of digital objects in an attempt to unify modern access control, trust management and DRM. The core of UCON is a family of what Sandhu and Park [SP03] call ABC models, built around the concepts of authorisations (A), obligations (B) and conditions (C). UCON is essentially a generalisation of the access control paradigm over the three concepts.

Figure 2.6 depicts ABC model components [SP03]. The rights in UCON are not predetermined and stored in some form of a static structure, like access matrix. Rather, they are determined on access request, with respect to the ABC factors. Such authorisation is the job of the *usage decision functions*. Obligations require an explicit action to be performed by the subject to gain or retain access. An example would be clicking the “ACCEPT” button on a license agreement. Conditions, as in RBAC$_2$, specify contextual factors that predicate access. Authorisations derive semantics from traditional access control.

In addition to the above factors, UCON provides for *continuity* and *mutability* properties. The former ensures ongoing right controls and their immediate revocation for the duration

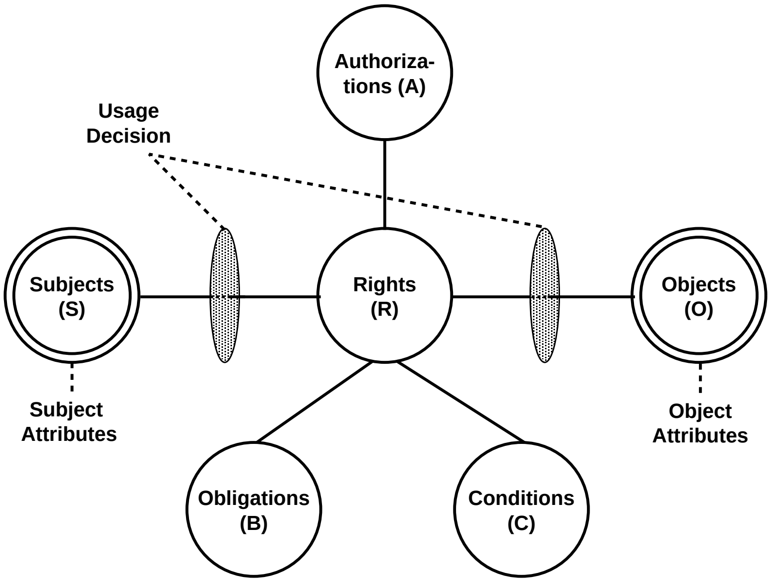


Figure 2.6: ABC model components.

of an access — unlike in more traditional access control where the decisions are tied to the point of access. The latter provides for modification of access decision relevant attributes as a side-effect of the user's actions. This is particularly useful in DRM scenarios, e.g. to limit number of viewing of digital content or to debit the user's account per access.

As specified in [PS02b] [SP03], UCON encompasses access control policies such as DAC, MAC and RBAC as well as trust management, DRM and active access management. [PS02b] also presents a selection of UCON applications. UCON seems to have been developed as an integral part of a wider information dissemination control paradigm, outlined in Section 2.6. In [PS02a] Park and Sandhu show how basic ORCON and UCON can be combined to provide for full information dissemination control. Implementation of UCON involves digital container [SBW95] protected information, control set (e.g. in the form of a license) and client-side reference monitor [And72].

2.9 Mobile Device Data Protection

As hinted at in Chapter 1, proliferation of mobile devices poses a number of challenges for data security. Most notably, mobile computing breaks the secure perimeter model, both in its digital and physical dimensions. This brings about unpredictability in information security threat model estimation, as discussed previously, in Chapter 1. The most widely recognised threat of information confidentiality compromise is due to the increased likelihood of the physical theft of the mobile computing devices themselves. A number of approaches for protecting data contained on mobile devices have been proposed. However, most are highly rigid and indiscriminate thus seriously hampering information availability, mobile device usability and functionality.

The first line of defence, most widely exploited and, unfortunately, often considered a *panacea*, has been the cryptographic protection of data at rest — i.e. contained within persistent storage. Cryptographic filesystems, e.g. [Bla93], represent the standard mechanism for this. However, entrusting the capacity to encrypt/decrypt data to mobile devices themselves fully may serve equally well to potential device thieves. Corner and Noble, in their work on Zero-Interaction Authentication (ZIA) [CN02], suggest use of *authentication token* for wireless, automatic, proximity-based authentication. The ZIA token is a device worn by the user, providing the user's decryption keys to their mobile devices on request. Unless the token is available, the data is (re-)encrypted. Extending encryption of data at rest, in [Pro00], Provos outlines a method for cryptographic protection of data on swap space. On the commercial side, a number of suites have been developed to ensure mandatory, holistic, data encryption on mobile storage devices. Perhaps the most widely recognised, and also certified under the Common Criteria [cc:b], is the Pointsec [Poi] mobile device data protection suite.

Other threats of information compromise in pervasive computing, beyond physical theft, have also been recognised. For example, utilisation of public services or untrusted communication links and the inability to control data flow centrally have been widely discussed. Lacking a flexible, adaptable, fine-grained and reliable enough solution, approaches that fully disable software and hardware features of the target platform that might potentially lead to information compromise have been suggested both in the academic world [KFJ03] [PKKJ04] as well as in the commercial sector. Furthermore, security policies that prohibit use of mobile devices with corporate infrastructure altogether are becoming increasingly popular, especially in the governmental sector. It is clear how, and to what extent, such approaches affect the usability and reduce the benefits brought about by mobile computing. This can be seen as one of the prime motivations behind CASPEr.

The most significant departure from the indiscriminate mobile device data protection was offered by Corner and Noble in [CN03], building on their previous work on ZIA [CN02]. In [CN03], Corner and Noble describe an API and the related functionality which exposes ZIA token's cryptographic functionality to applications running on mobile devices. More interestingly, the ZIA token offers a proximity-based call-back service to the applications — allowing them to implement their own adaptation methods. This is in addition to the event-driven data storage encryption. Although the work represents a significant advance in terms of flexibility and context-awareness, the threat model space remains binary as determined by the state of token presence. The CASPEr model represents a step forward with respect to threat modelling granularity as well as protection continuity. The CASPEr framework effectively encompasses the functionality of [CN03].

2.10 Modeling the World

In Chapter 3 we introduce an approach to modeling the world based on the notion of *container*, a protective enclosure, and its hierarchical structuring into *containment trees*. The inspiration for the model originates from the research conducted in the area of mobility theory as well as from the approaches to modeling space in geo-information systems.

To be able to express, reason about and control the location-dependent behavior and movement of processes in distributed systems, research in the area of mobility theory (mobile computation in particular) has devised a number of ways to model the state of the world. The common set of entities represented by such models are the (mobile) processes, communications channels (between the processes) and locations (in which computation occurs). The processes represent active entities that embed the mobile computation and exhibit dynamic behavior. The π calculus [Mil99] and its derivatives, such as asynchronous, distributed or nomadic π calculus, model distributed communications systems solely on the basis of processes and communications channels among them. MOB_{adtl} [FMSS03] models the world as a flat structure based on the notion of a *neighborhood* as a building block. Join-Calculus [FG96]and Cardelli's work on Ambient Calculus and Mobile Ambients [CG98] [CG00] propose hierarchical models based on the notions of *locations* and *ambients* respectively. A further hierarchical model of the world, based on the concept of a *seal*, is proposed by the Seal Calculus [CVN05] — itself partially inheriting from the π calculus. The similarity between the above notions of *neighbourhoods*, *ambients*, *locations* and *seals* and the concept of a *container* as introduced in this thesis goes only as far as all of them represent a form of an enclosure, physical or virtual, within which further entities may exist and whose migrations assume its contents. The semantics of an enclosure in the mobility theory, however, reflects the fact that its purpose is representing, reasoning about and controlling the behavior (e.g. mobility) of mobile computation entities (the processes), e.g. mobile agents, within a distributed system. Consequently, the related formalisms, e.g. calculi, algebras and policies, are of significant complexity. The notion of a *container*, as we introduce in Chapter 3, and its role within the overall model are of a significantly different nature. Firstly, the introduced model itself is comprised solely of static entities, not capturing mobile computation in any of its forms. Secondly, the role of a container or, more precisely, of its boundary is solely in reasoning about the threat models present in its exterior and interior and their dependence — effectively playing the role of a fully passive filter. Lacking the need of supporting mobile computation or any active behavior, we model the world (Chapter 3) in a manner which resembles the subset of the ambient calculus [CG98] which describes the structure of space with no active processes.

Originally, the inspiration for the concept of a container comes from the research in representation of space for geo-information systems. Research on spatial databases [SCR$^+$99] offers a number of geo-information systems inspired ways of structuring the world in *field-oriented* and *object-oriented* ways. Egenhofer and Rodriguez [ER99] [RE00] offer an ontology of space based on the *container-surface* paradigm and related algebras. The notion of an container, as used in our work, extends the Egenhofer's container into the virtual world in which it resembles a passive ambient from Ambient Calculus work [CG00] with an explicit notion of a container's boundary and its threat filtering characteristics.

2.11 Summary

The material presented in this chapter serves mainly a two-fold purpose. The initial three sections on context aware computing, autonomic computing and the general perspective on security issues in ubiquitous computing described the wider setting of this thesis and the constraints implied by it. The sections on information flow control, information dissemination control, access control and information usage control depicted the information security big picture to which CASPEr contributes. Special emphasis was placed on identifying the individual entities and the roles they play in each of the mechanisms, nature and source of the particular threats and the threat types addressed. These are the characteristics that most clearly distinguish CASPEr from the related work. For completeness purposes briefly outlined the current trends in mobile device information security protection. Rather than directly contrasting CASPEr to each of the mentioned security models and mechanisms in this chapter, we postpone the comparison to Chapter 6, after having introduced all CASPEr concepts in detail. There, we present a qualitative comparison of CASPEr with the major relevant information security paradigms — crystallising the contribution this thesis makes and precisely specifying the role CASPEr plays in the information security big picture. Since the majority of the novel concepts introduced by CASPEr rely heavily on the approach to modeling the world that we present in Chapter 3, in the last section of this chapter we related it to the contributions in the areas of mobility theory and geo-information systems.

Chapter 3

Containers and Containment — Modelling the World

3.1 Chapter Overview

In this chapter we introduce an approach to modelling the world in an information-centric manner, founded on the concept of a *container* as a protective enclosure and its structuring into higher-level *containment* entities — the basis for describing the state of the world. The model itself is aimed at providing for well-founded, structured and data object grained information exposure threat analysis as well as reasoning about localised threat mitigation, continuously throughout information life-time. The concepts and formalisms introduced in the chapter lie at the foundation of CASPEr and form a basis on which the material presented in the following chapters rests.

In the initial section, we provide a formal definition and characterisation of information exposure threats, as used throughout the thesis. We define the threats in a wider context of *contextual effects* — a concept that captures implications a contextual state has on target entities, and argue for the benefits of explicit contextual effect reasoning. We proceed to motivate and introduce, in Section 3.3, the fundamental concept of *container* as a protective enclosure. The protective capabilities are conceptualised as container *transparency* — a quality of the container's boundary to filter information exposure threats originating in its exterior and potentially affecting entities enclosed within its interior. Paving the way to the domain ontology, we present a basic container classification together with a relationship specifying container nesting capabilities. Based on the latter, as presented in Section 3.4, containers can be organised into higher-level entities that we refer to as *containments* which, in turn, are used to describe the state of the world. To facilitate model manipulation and maintenance, we introduce a number of related formalisms, leveraged throughout the thesis. We also show how the model is deployed in units called *realms*, fully honouring the autonomy of individual ubiquitous computing devices, tailored to their resource capabilities. Combining the container transparency and the structure of the containment-based model of the world, in Section 3.5 we present the overall approach to fine-grained threat analysis. Furthermore, we show how the model facilitates reasoning about localised threat mitigation and introduce two categories of threat mitigation operations. Finally, in Section 3.6, we specify a property that guarantees the safety, with respect to information exposure protection, of tailoring the model expressiveness to the capabilities of the target deployment devices.

3.2 Information Exposure: from Context to Contextual Effect Awareness

3.2.1 What is a Contextual Effect?

Research in the area of context aware computing offers a number of approaches to capturing, modelling at various levels of abstraction and dissemination of contextual information. Apart from the cases in which the obtained contextual information is provided for user awareness, the aim of context-aware computing is adaptation. Context adaptation binds a *contextual state* to one or more adaptation actions. We use the term contextual state in this thesis to denote contextual information that describes an environment at any level of abstraction. Contextual state is comprised of contextual fragments. The coupling between contextual states and adaptation actions is usually specified as a set of rules, forming a context-adaptation policy, or is, more often than not, hard-wired into context-adaptive applications themselves. What remains implicit in such approaches is the actual interpretation of a particular set of contextual fragments that leads to the adaptation. In other words, the answer to the question: *What is the cause of the particular adaptation as seen in the context?*.

For example, while availability of multiple communications links means *increased connectivity* from a point of view of context aware networking, it may mean *higher risk* from the security stance. Similarly, whereas switching displays in favour of a bigger one may mean *enhanced information presentation capabilities* from a Human Computer Interaction (HCI) point of view, it may signify a *bigger probability of an unauthorised party overseeing the information* from the information security angle. Each of the interpretations causes a different adaptation procedure.

Definition. *A Contextual Effect is a concept that qualifies and quantifies a single consequence, i.e. a specific impact, that a contextual state as a whole or a set of its fragments has on a target entity or a set of target entities.*

In general, the mapping between contextual states and contextual effects is *many-to-many*, as hinted at in the above example.

3.2.2 Why Explicit Contextual Effects Modelling?

One of the first proposals that we make in this work is explicit modelling of contextual effects. Doing so leads us away from the traditional two step adaptation process:

contextual state → adaptation

To a three step context adaptation process proceeding as:

contextual state → contextual effect reasoning → adaptation

Explicit contextual effect modelling can be seen as abiding by the more general *separation of policy and mechanism* principle first published in [LCC$^+$75] and since then a "modus vivendi" in many aspects of computing. The fundamental advantage of the approach is simplification of changes in policy required to be made to cope with altered operational circumstances. Honouring the principle, explicit contextual effect modelling allows for dynamic changes in available application and platform adaptation mechanisms with no, or very little, need for any policy or component updates. This is of particular importance in the area of ubiquitous computing characterised by weak or non-existing service availability guarantees. Even more so when the governing principles of autonomic computing [IBM01] (Section 2.3) are considered, in particular:

- *"An autonomic computing system must configure and reconfigure itself under varying and unpredictable conditions."*
- *"An autonomic computing system never settles for the status quo — it always looks for ways to optimise its workings."*

We believe that explicit modelling and reasoning about contextual effects is one of the necessary steps toward the joint vision of ubiquitous autonomic computing. Furthermore, explicit contextual effect modelling facilitates context reuse at a different dimension from the pure context abstraction approaches.

Henricksen and Indulska [HI04], and Dey and Abowd [DA00] earlier, recognised a similar need. However, they attempted at addressing it by introducing yet another level of abstraction of contextual information. In [HI04], Henricksen and Indulska, propose the *situation* as a programming concept which describes in a sufficiently abstract way contextual information that determines application behaviour. Situations can be combined promoting reuse and enabling incremental formation. They are expressed using a form of predicate logic in terms of contextual fragments. However, the process of abstracting, rather than correlating, contextual information retains the embedding of the application-specific semantics. Explicit contextual effects modelling, on the other hand, extracts this semantic information — in particular the qualification and quantification of impact a contextual state has on a target entity.

When we refer to the terms *context* or *contextual state* in the rest of the thesis we assume contextual information as a source of contextual effects.

3.2.3 Contextual Effects Characterisation

Contextual effects have two fundamental characteristics: a *type* and a *degree*. Contextual effect type is intended to capture the nature of a contextual effect. While the type can be seen as qualifying a contextual effect, the degree represents its quantification. Semantics of exposure degree are subject to application-specific context interpretation. Contextual effect type ensures identification of an appropriate adaptation process to perform while the degree is intended to facilitate the choice of appropriate level of adaptation. In addition to the type and the degree, contextual effects may also be characterised by a set of application specific *attributes*. The attributes may be used for a variety of purposes, most notably for constraining the set of target entities affected by the contextual effect according to some criteria. We present relevant examples in the next sections.

3.2.4 Information Exposure Threats as Contextual Effects

Whenever we consider an "accident" in the context of computer security we can distinguish between the *mechanism* and the *process*. The mechanism denotes a characteristic of a systems component, a specific "feature" or a set thereof, that enables mounting an attack that leads to an "accident". The *process* is described as the actual sequence of steps, or operations, performed to exploit the mechanism. A single mechanism may be exploited by multiple processes in different ways.

Definition. *Information Exposure Threat characterises information exposure process and captures the probability that it occurs in a specific context.*

The other way around, we say that an information exposure process as it occurs is a *materialisation* of a threat. Referring back to the definition of information exposure, an exposure mechanism assumes a particular set of information management and handling procedures. On the other hand, context determines the feasibility of and sets the constraints for existence of a particular information exposure process. Consequently, information exposure threats represent a class of contextual effects.

For example, in order for a piece of information to be overseen (*the process*) while being displayed on a screen in a particular form (*the mechanism*) an unauthorised party must be able to exist within a certain proximity to the screen (*the context*). Similarly, for a piece of information stored on a storage of a mobile device in plain text (*the mechanism*) to be exposed the device needs to be in a place (*the context*) where it can be abducted (*the process*) by a malicious party. In the information exposure terminology, the notion of a *threat model* represents a set of information exposure threats that exist in a given context for a piece of information.

Explicit modelling of information exposure threats represents a clear instance of the separation of policy and mechanisms principle with all of its benefits, especially emphasised in the target setting of ubiquitous and autonomic computing. In the rest of the thesis we use the terms *threat* and *exposure* to mean *information exposure threat*, unless explicitly stated otherwise.

3.2.5 Information Exposure Threats: The Probabilistic Nature

Presence of an exposure threat and existence of an exposure mechanism are necessary but not sufficient causes for the exposure to actually occur, i.e. for the threat to turn into a process. Even if we could, hypothetically, assume perfectly precise and detailed context models we still would not be able to reason with absolute certainty about factors such as motivation, knowledge and resource capability of an attacker to leverage an exposure mechanism. Thus, we see information exposure as a highly non-deterministic process. Moreover, due to its nature, information exposure is not necessarily evident even as it occurs.

Due to the above reasons we need to resort to statistical tools and probabilistic modelling to establish information exposure threats in an environment. In statistics, obtaining a causality relationship between events rests mainly on empirical methods which are sometimes aided by the deployment of inference processes to complete or set the foundation of the relevant causality structure. Experimentation in the context of information exposure threats is, however, highly infeasible due to their sensitive nature.

Therefore, we assume a probabilistic correlation process between individual contextual fragments and information exposure threats. Frequently heard objections to probabilistic measures of security in general are due to the essentially unrepeatable nature of the key events. This is even more so in the case of information exposure due to frequent difficulty in identification of the key events themselves. Furthermore, for security, the uncertainty often concerns one-off events. Thus, a subjective, Bayesian like, "strength of belief" interpretation of probability is required - given by the probabilistic correlation approach.

We do not devote more space to exploration of the relevant statistics and probability instruments as they are well established and can be found in any textbook on the topic. In practise, we expect the correlation of context and information exposure threats to come from risk modelling approaches involving historical experience and expert opinion.

3.2.6 Information Exposure Threat Characterisation

Being contextual effects, information exposure threats inherit the characterisation by the *type*, *degree* and *attributes*.

Threat Type

We distinguish between *primitive* and *compound* threat types. At this point we define only the former while the latter are defined in Section 3.5 after all the pre-requisite concepts have been introduced.

Definition. *Primitive information exposure threat type denotes the nature of the exposure mechanism — as the enabler of the exposure process.*

In other words, primitive exposure threat types represent the nature of the respective information leakage channels. Unless specified otherwise, we use the term threat *type* to mean primitive information exposure threat type. For the purpose of clarity of the material to be presented we provisionally introduce the following threat types:

- *Optical* information exposure denotes that a piece of information can be "seen".
- *Acoustic* information exposure occurs when information can be "heard".
- *Physical* information exposure threat type denotes that a device containing sensitive information, such as a mobile computing device or a removable storage device, can come under the physical command of a unauthorised party.

Information threat typification shall be systematised along the definition of *compound* threat types in Section 3.5.

Exposure Degree

The standard way to quantify information leakage is in terms of amount of data that leaks in a unit time, e.g. *bits/sec.* Such a metric characterises an ongoing process. Due to its subtle nature, information exposure may not be evident as a process, i.e. as it occurs. Therefore, we characterise information exposure threat *severity* as the absolute proportion of data that leaks from an affected data object (e.g. a file) should the threat materialise. Throughout the thesis we refer to this metric as the *width* of an information exposure channel.

To quantify information exposure threats, unlike information exposure process, along the width of an exposure channel threat *degree* needs to account for two uncertainties: i) context capturing uncertainty; and ii) threat materialisation uncertainty. The latter is context dependent and has already been discussed. The former is an intrinsic property of the context modelling process at all levels of abstraction. It has been a topic of a wide area of research in context aware computing and we consider it out of the scope of our work.

Information exposure threat degree can be expressed as:

$$degree(u_{context}, p_{threat}, w_{channel}) : val$$

where the first two arguments are the two uncertainties and $w_{channel}$ is the exposure channel width. The level of measurement used for expressing the exposure degree is considered application specific. It is dependent on the methods employed for establishing and expressing the uncertainties in the context - threat correlation process.

Threat Attributes

Information exposure threat attributes serve the purpose of constraining threat reasoning to specific target entities — pieces of information, in our case. For example, in an office with a number of screens oriented in different directions and a location infrastructure able to determine orientation of individuals, e.g. ActiveBat [HHS+99], we can constrain reasoning about the threat of, line-of-sight, *Optical* information exposure to screens in the visibility field of a particular person. Furthermore, in multi-level security environments, we can leverage the attributes to match threats to affected data items based on information classification and principal clearances (Section 2.5).

3.3 Container - the Basic Building Block

3.3.1 Motivating Example

To create an intuitive feel for the notion of a container and the manner in which we build upon it in the following section to develop an approach to modeling the world, we commence this section with a simple example drawing from the motivating scenarios presented in Chapter 1. Consider Alice being in a busy airport lounge, waiting for the departure of her flight, and in a possession of a number of corporate sensitive documents contained on her mobile computing devices as follows:

1. Displayed full-size on:
    (a) 14” laptop display (full-size),
    (b) 4” PDA display (full-size).
2. Stored on:
    (a) Laptop’s fixed storage device:
        - Encrypted, and
        - Unencrypted.
    (b) PDA’s fixed storage device:
        - Encrypted, and
        - Unencrypted.

Alice’s environment represents a source of *Optical* and *Physical* information exposure threats. What can we then say about the threats as experienced by the individual documents? Intuitive thinking suggests that they experience varied exposure degrees depending on their actual “whereabouts” — i.e. the data handling and management procedures leveraged for each.The physical size of Alice’s laptop screen permits its contents to be observed, or captured, from a distance that makes the process unobtrusive to her. In other words, we say that the displayed document is, in this case, under the *credible* threat of Optical exposure. The same, however, does not hold for the document shown on the PDA's screen. We consider a threat as credible if it necessitates active mitigation. On the other hand, the documents contained on the storage devices are immune to the threat of Optical exposure. However, owing to their physical size and mobility characteristics, the two devices are prone to being abducted. Consequently, the data contained within them is under the threat of Physical exposure — but not uniformly so. Regarding the information present on the laptop's storage, in spite of the inherent level of security of the environment, the threat for the unencrypted data can still be considered above the credibility threshold. Encrypting the data may be considered enough to decrease the experienced threat degree sufficiently to be considered as an adequate threat mitigation action in this case. On the other hand, as the PDA is significantly easier to steal, conceal and smuggle, both encrypted and "plain-text" sensitive documents on its storage device may be considered under the credible degree of Physical exposure.

The example clearly shows the delineation between the context, seen as exposure threat source, and data management and handling procedures, providing for threat mitigation. We refer to the example throughout the section as an illustration of the motivation for modelling the world based around the concept of *container*, which we introduce next.

3.3.2 Container: The Definition

Container, alike contextual effect, represents an abstract concept. The fashion in which the definition of a *container* is phrased tries to retain the level of abstraction for the purpose of the concept generalisation beyond the presented application. An exception to this tendency is making the definition information centric, with no loss of generality, as required in the context of information exposure threats.

Definition. *A container is a semantic construct that represents a physical or a virtual enclosure in which another container or, ultimately, a piece of information may exist. An enclosure is a bounded region, physical or virtual, with clearly distinct interior and exterior delimited by its boundary.*

In other words, a container is effectively defined by its *boundary.* The semantics of an enclosure imply that migrations as well as destruction of a container assume all of its contents. Other than that, the manner in which a container boundary delimits container's interior and exterior is specific to the nature of contextual effects being modelled — information exposure threats in our case. Semantically, enclosure is a more abstract concept than a container. The latter encompasses various application-specific properties of the boundary — as clarified shortly. In the context of information exposure threats, container may be seen as a conceptualisation of threat affecting characteristics of information handling and management procedures. For instance, the motivating example has implied that storage devices block the threats of Optical exposure and that encryption lessens the degree of Physical exposure.

Container Transparency

The fundamental concept that makes container an enclosure is the *boundary*. Contextual effects may arise from the state of context inside or outside a container, relative to the boundary. The primary role of the container boundary is to confine its contents with respect to information exposure threats occurring in the container exterior. For a contextual effect originating outside a container to affect entities contained within the container it has to "cross" its boundary. The crucial point is that in the process of crossing the container boundary the threat can be quantitatively affected. This is why we refer to container as a protective enclosure.

For example, for a sound made inside a room to be audible outside it, the room's boundary (consisting of walls, doors, windows etc.) has to be permeable for a set of corresponding frequencies at the particular volume. In the context of information exposure, we reverse the point of view and say that a threat of *Acoustic* information exposure, in this case, penetrates into the container that represents the room. Similarly, if information displayed on a screen can be observed by a third party we say that a threat of Optical information exposure penetrates into the container representing the display. Whenever we refer to information exposure we assume such, reversed, point of view.

Definition. *Container transparency is a quality of the container boundary to reduce degree of contextual effects as they cross it.*

Let C be a set of transparency relevant container characteristics and T be a set of contextual effect types. Assuming that the degree of contextual effect takes values from a domain D (the examples in the rest of the thesis use the set of real numbers, $\mathbb{R}$, as the domain without loss of generality), we can specify the form of the transparency function as:

$$transparency : \mathrm{P}(C) \times T \times D \longrightarrow D$$

where $\mathrm{P}(C)$ is the power set of C. The degree of an exposure prior to crossing a container boundary is always greater or equal to its degree afterwards. Container transparency affects the degree of information exposure threat through lessening the threat materialisation likelihood in the context and/or the width of the exposure channel. An example of the former is displaying sensitive information on a smaller screen while a reduction in data transmission rate over a wireless channel illustrates the latter.

The concept of the enclosure is not unknown in the context of modeling the world in the mobility theory, as referred to in Section 2.10. However, the semantics of the enclosure boundary in the mobility theory capture the role of enclosure in confining active entities — mobile computation, their behavior and its effects. On the other hand, the semantics of a container boundary as introduced in this thesis are of a different nature. They capture the role of a container as a protective enclosure, a passive filter for information exposure threats, with respect to the static entities within its interior. Furthermore, the concept of a container is devised solely to reflect, in a passive fashion, the respective aspects of the state of the overall world — the modifications of which are external to the model.

Going back to the motivating example of Section 3.3.1, the notion of transparency allows us to conceptualise the intuitive difference among the information exposure threat degrees as experienced by sensitive data contained on the respective devices — modelled as containers. This is a first step toward the formalisation and generalisation of reasoning about information exposure threats as experienced by individual pieces of information in a particular setting. For example, while the PDA screen is less transparent for the threats of Optical exposure, the storage devices are fully *opaque*.

Multiplicative vs. Additive Transparency

We distinguish between two main classes of transparency functions as: *multiplicative* and *additive*. Let k represent the *transparency coefficient* which is dependent on the contextual effect type and the transparency relevant container characteristics. Multiplicative transparency functions take the general form of:

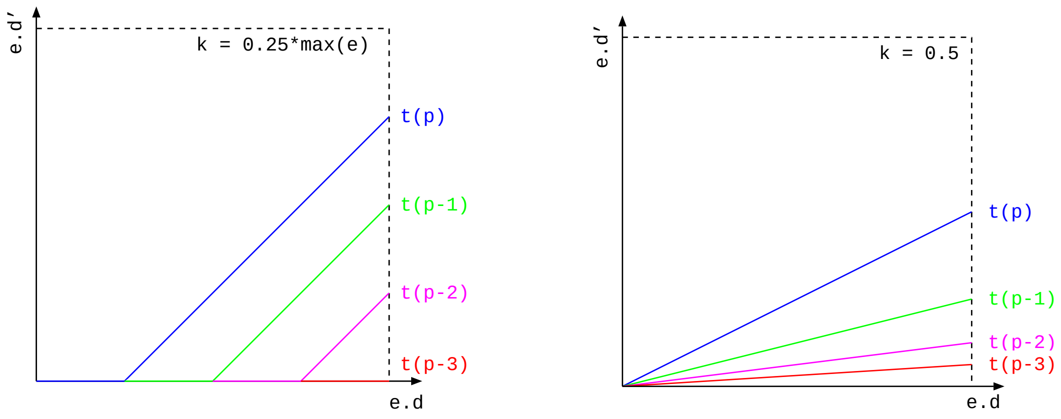
$$transparency(X \subset \mathrm{P}(C),\ e.type \in T,\ e.degree \in \mathbb{R}) = k\ \times\ e.degree,\ 0 \leq k \leq 1$$

Where e denotes information exposure threat of type $e.type$ and degree $e.degree$. The class of additive transparency functions takes the form of:

$$transparency(X \subset \mathrm{P}(C),\ e.type \in T,\ e.degree \in \mathbb{R}) = \begin{cases} e.degree - k & \text{if } e.degree \geq k\ ; \\ 0 & \text{if } k > e.degree \end{cases}$$

Figure 3.1 is an illustration of the behaviour of each of the two classes of the transparency functions. Each of the individual transparency functions in both of the graphs corresponds to an incremental change in a single threat type relevant container characteristic. For example, the size of a GUI window in which information is displayed. In Figure 3.1(a) each of the unit reductions in GUI window size (starting with size p) accounts for a decrease in the exposure degree by a quarter of the maximum exposure degree. In Figure 3.1(b) each of the unit size reduction accounts for 50% of the *experienced* exposure degree. Note that the former is an absolute change while the latter is a relative change. The difference in the effect of unit impact in the two example plots is solely for the purpose of clarity of the graphical presentation.

Figure 3.2 depicts percentage impact of a single unit decrement in the GUI window size across the exposure degree range. The chosen exposure degree range, $[0, 10]$, serves solely the purpose of illustration. The transparency coefficients remain the same as in Figure 3.1 respectively. To analyse the graphs, we consider again the case of the two screens used in

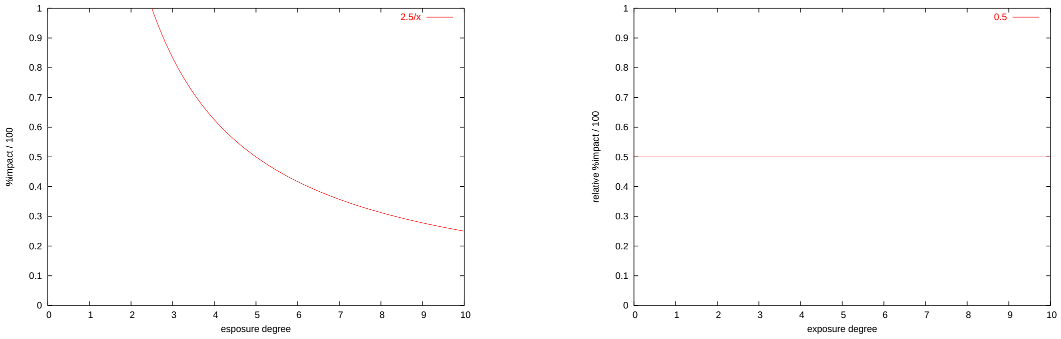


(a) Additive transparency.

(b) Multiplicative transparency.

Figure 3.1: Example transparency functions.

the motivating example. What the graph in Figure 3.2(b) effectively tells is that no matter how close in front of the screens a third-party is they are always less likely to observe the contents of the smaller (PDA) display. However, we believe that above a certain proximity the difference in the screen size would not, in practise, impact the likelihood of information exposure. With respect to this, the additive transparency function in Figure 3.2(a), in our opinion, describes the container transparency effect more realistically. This applies to the class of additive transparency functions in general. The same effect can be obtained if multiplicative transparency functions are defined non-linearly — with the transparency coefficient varying across the exposure degree range. We, however, do not consider this option as it entails substantial modelling and operational complexity.

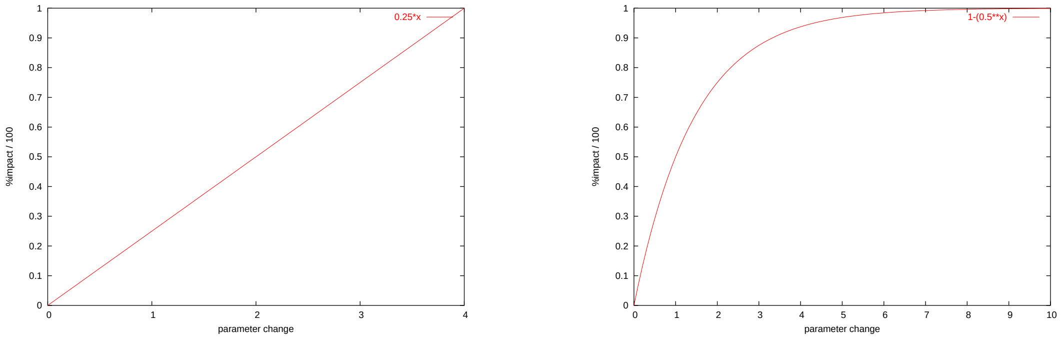


(a) Additive transparency.

(b) Multiplicative transparency.

Figure 3.2: Percentage impact across exposure degree range.

Finally, Figures 3.3(a) and 3.2(b) plot the cumulative percentage impact over the unit decrements in the GUI window size for the additive and the manipulative transparency function classes respectively. As implied by the plots, the additive transparency coefficient scales linearly with the number of unit changes in the respective transparency characteristic. This is not the case for the multiplicative transparency functions — thus the shape of the plot in Figure 3.3(b). This further reinforces our stance that the additive transparency better reflects reality.



(a) Additive transparency.

(b) Multiplicative transparency.

Figure 3.3: Cumulative transparency impact - single parameter change.

3.3.3 Toward a Container Ontology

The term ontology originates from philosophy where it denotes the study of being and existence — aiming to discover what entities and what types of entities exist and what are their relationships. The term ontology, as used in computer science, is derived from the usage of the term in philosophy. In computer science and information theory, ontology is an exhaustive and rigorous conceptual schema about a domain — also referred to as a *domain ontology*. Ontologies have so far been most commonly used in artificial intelligence and knowledge representation.

The anatomy of an ontology consists of *concepts*, their *attributes* and their *relationships*. A concept is any entity about which something can be said, it can be concrete or abstract, it can be a process or an object. Concepts are also usually referred to as *classes*. Attributes are used for describing concepts and they may be used for establishing more complex relationships between concepts than would otherwise be possible. An ontology lacking attributes is known as a *taxonomy*. Finally, relationships allow us to fully conceptualise the domain of interest.

To allow for systematising the knowledge about containers and facilitate concept reuse we lay foundations for a container ontology which is envisaged to be extended in application specific manners. The ontology is directed toward the domain of information exposure threats. However, we emphasise that the concept of a container and the relationships to be introduced are generalisable to the wider concept of contextual effects.

Container classification

The first step in developing the container ontology is the definition of *container class* and the corresponding classification.

Definition. *We say that two containers are of the same class iff:*

1. *The containers are characterised by the same set of attributes, and*
2. *They, under the same valuation of their attributes:*
    - *exhibit the same level of transparency for all known information exposure threats, and*
    - *form the same set of relationships with other containers.*

A particular valuation of container class attributes is a property of a container class instance and we call it container *state*. The possible relationships among containers shall be introduced shortly. We use the term container *de-facto* transparency to denote transparency inherent to a container class. This means that the transparency of an instance of a container class may never be greater than its de-facto transparency, irrespective of the values of the particular container attributes. The de-facto transparency is, thus, a property of container class rather than of its instance.

Figure 3.4 shows a simple container classification hierarchy. It is by no means complete but is sufficient to illustrate the concepts presented in this thesis. Having said this, we see the top two levels of the classification, together with the *Data Item* container class, as universal (depicted in red). The most abstract container class is the *Container* class. It represents the concept of a container in the most general sense. The next level in the classification essentially serves the purpose of bridging the physical and virtual worlds:

- ***Physical*** containers denote entities that exist solely in the physical world, i.e. can be characterised by three dimensions in Euclidean space and by their physical volume. Examples of Physical containers are: a room, a physically secure perimeter, inside of a car etc.

- ***Virtual*** containers, as opposed to Physical containers, denote entities existing solely in the virtual realm. Virtual containers cannot be characterised by Euclidean dimensions or physical volume. However, they do have explicit virtual boundaries within which other virtual containers may be confined. The semantics of virtual and physical boundaries are equivalent. Examples are: a GUI window, an encryption envelope, etc.

- ***Intermediate*** containers have a role of bridging the gap between physical and virtual worlds. They represent physical enclosures within which only Virtual containers may be confined. In other words, Intermediate containers have virtual volume — alike

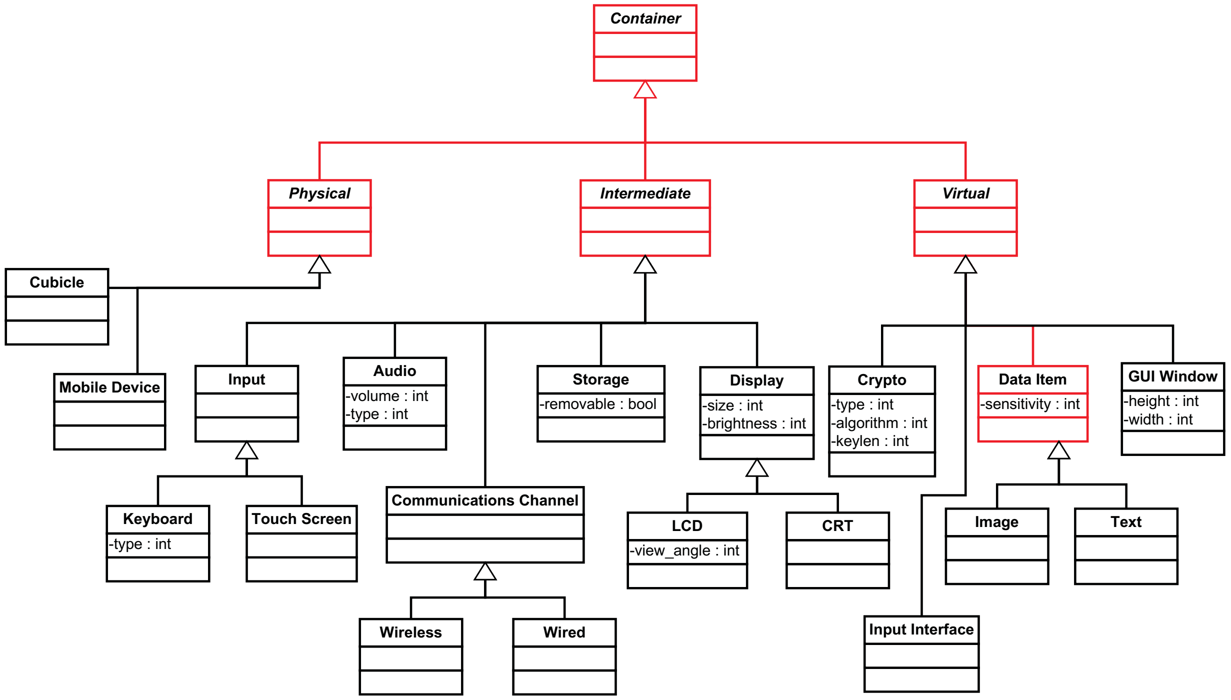


Figure 3.4: Example container classification.

Virtual containers, but their size can be described with Euclidean dimensions. Thus, they draw from both the worlds. Examples are: a screen, a storage device, a network cable etc.

Depending on the nature of a container, its boundary is itself physical or virtual, and so are its interior and exterior. However, note that the interior of an Intermediate container is virtual while its exterior is physical.

Data Item. In order to be able to reason on how information exposure threats affect information we define the *Data Item* container class. A *Data Item* serves as a data representation of information. The notion of a data item does not, in general, correspond to the traditional concept of a file. It represents a finer-grained entity grouping information that is affected in the same way by all information exposure threats. In our work, a data item groups information of the same sensitivity level. The sub-file granularity of a data item facilitates more flexible, focused and finer-grained information protection.

Figure 3.4 also shows sub-classes of the *Data Item* class. They are introduced to represent individual data types in cases where the container classification granularity is at such a level that the threat transparency relevant attributes can be defined on a per data type basis. Examples of data type specific attributes would be the image quality and information obfuscation level in a text document.

Relationships

As we have previously mentioned, relationships play an important role in the conceptualisation of an ontology domain. The following two are fundamental for establishing the container ontology: the "is a" and the "may contain" relationships.

The "is a" relationship. The "is a", or sometimes referred to also as "kind of", relationship is reflected in the structure of the container classification hierarchy as shown in the Figure 3.4. With respect to the relation, every entity is said to be a *child*, a *sub-class*, of at most one *parent* class, or *superclass*. Semantics of the relationship imply that all the attributes of the parent class are inherited by its children. For example, a *Data Item* "is a" *Virtual Container* and *Keyboard* "is a" *Input Device*.

The "may contain" relationship. The "may contain" relationship specifies the nesting compatibility between instances of container classes. Let C be the set of container classes and A be the set of all container class attributes. For each $a \in A$ there is a set D_a of the values the attribute can acquire — its domain. The "may contain" is a binary relationship defined on tuples of form $\langle c, \{(a_1, V_1), \ldots, (a_n, V_n)\}\rangle$ where $c \in C$, $a_1, ..., a_n \in A$ are attributes of the particular container class and $V_i \subseteq D_{a_i}, i = 1, \ldots, n$. In other words, the "may contain" relationship is defined for pairs of containers based on the values of their attributes. The relationship is irreflexive, asymmetric and intransitive. It is also not total. In graph theoretic notation, the "may contain" relationship imposes a directed acyclic graph structure on instances of container classes.

We model the relationship using the *containable* predicate. If container `a` "may contain" container `b` then `containable(a,b)` evaluates to true. At the level of abstraction of Physical, Intermediate and Virtual containers, the "may contain" relationship is defined as follows:

- *Physical* containers "may contain":
    1. *Physical* containers, dimensions and volume permitting.
    2. *Intermediate* containers, dimensions and volume permitting.
- *Intermediate* containers "may contain" *Virtual* containers only.
- *Virtual* containers "may contain" *Virtual* containers only.

At the lower levels of the hierarchy, the "may contain" relationship is specified on a per container class basis — which constrains, but does not override, the above rules. In the general case, the "may contain" relationship is inherited down the container classification hierarchy and can be further specialised by each sub-class.

Applied to containers of class *Data Item*, the *containable* predicate always evaluates to *false*. In other words, data items may not contain any other containers irrespective of their class or state. This establishes the information centric approach to modelling the world that we take. We say that containers of class *Data Item* are *atomic* containers.

3.4 Containment - the Model of the World

Building on the concept of a container and leveraging the container classification and the "may contain" relationship allows us to represent the world in a structured and well-founded way. In this section we introduce the notions of *containment* and *containment based model of the world*. The model is built around the concept of a *Containment Tree*.

3.4.1 The Model of the World: Containment Trees

We model the world by nesting containers to represent higher level entities in a structured manner. Observe that this is analogous to the way in which *entities* are nested in Ambient Calculus [CG98] [CG00]. For example, a laptop is itself comprised of a number of containers — a screen, a storage device, a communications interface, a input device etc. These, in turn, may contain further containers — all the way down to data items.

Definition. *A Containment Tree is a rooted, acyclic, connected and directed graph in which nodes represent containers and edges denote the contains relationship. All the paths from the root to any of the outer, i.e. leaf, nodes of the graph are of finite length. Each node is of a finite degree.*

We say that container `a` *contains* container `b`, and write `a` → `b`, if container `b` exists within the boundary of container `a`. The *contains* relationship reflects the state of the real world rather than being a static property of containers or container classes.

We say that a containment tree is *well structured* if the "may contain" relationship is never violated during the tree formation. In a well structured containment tree, data items are always leaf nodes.

Finiteness of the path lengths is guaranteed by the existence of the atomic container in conjunction with the fact that the "may contain" relationship induces a direct acyclic graph structure on containers. Degree of a node, i.e. its branching factor, is equivalent to the number of edges leading away from the node. Finiteness of the container branching factor for Physical and Intermediate containers is guaranteed by the physical size constraints of the real-world entities they model. With respect to Virtual containers, the container degree finiteness is ensured by software and hardware platform resource overheads involved in management of the real-world entities they represent.

When we refer to the term *containment* with respect to a particular container it denotes the sequence of nested containers from the containment tree root to the container. Otherwise, in the general sense, the term denotes the overall state of being contained. Containment of a data item can be interpreted as a particular data management and handling procedure the data item is subjected to. For example, being displayed on a large, instead of on a small, screen or stored in an encrypted, rather than in the plain-text, format as in the motivating example (Section 3.3.1).

Definition. *A containment-based model of the world is, in graph theoretic terminology, a finite forest of containment trees.*

A forest is a directed, acyclic graph in which each of the disconnected components has a root. When we say that a containment based model of the world, or simply model of the world, is *well-structured* it means that all of the containment trees comprising the model are well-structured as defined above. The finiteness of the forest is ensured by the finite number of higher level entities, such as individual ubiquitous computing platforms, that exist in the world.

3.4.2 Containment Expressions

Containment Expressions represent the syntax of Containment Trees. We make extensive use of containment expressions for identifying containers in a realm, as specified in Section 3.4.4 and leveraged throughout the following two chapters. Furthermore, formally defined syntax provides for well-founded and sound description and reasoning about the structure of the world — assumed throughout the thesis. To present the syntax of containment expressions we break them down into atomic expressions and provide a graphical representation of the matching containment tree fragment. The notation that we use is: i) lower-case letters of Latin alphabet are used to refer to containers, assuming their state but not their contents; and ii) capital letters of Latin alphabet denote containment trees. In both cases we use the `typewriter` font style.

Containment Expressions and their Representation

To start with, absence of contents at any level is represented simply by "0". Thus, a model comprised of only "0" at the top level represents an empty world.

A tree with only a root node `a`, whose state is defined by the valuation of its attributes $(a_1 = v_1, \ldots, a_n = v_n)$, is written as the expression `a` : $a_1 = v_1, \ldots, a_n = v_n$; :

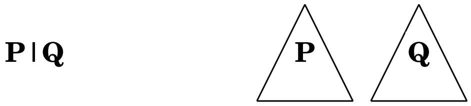
a : attributes_list ;
a

Note that in the containment tree representation we omit the attributes valuation for clarity purposes. Furthermore, from now onward we omit the specification of container attributes valuation in containment expressions unless required for the purposes of argumentation.

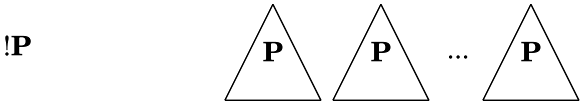
A forest, consisting of two trees with only root nodes a, of the same container class but different attribute valuations (not explicitly shown), is written as the expression a|a:

A tree, with a root node labelled a, leading to a subtree P is written as the expression a[P]:

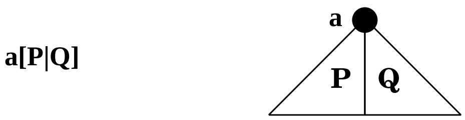
A forest, consisting of two trees P and Q, is written as the expression P|Q:



Multiple instances of the same tree P is written as the expression !P:



A tree obtained by joining two trees P and Q at the root a is written by the expression a[P|Q]:



3.4.3 State of the World

Using the containment expressions, any well-structured state of the world at the level of abstraction of Physical, Virtual and Intermediate container classes, is given by:

$$
\begin{aligned}
world &\leftarrow world|world\\
world &\leftarrow physical\\
world &\leftarrow intermediate\\
\\
physical &\leftarrow physical|physical\\
physical &\leftarrow physical[physical]\\
physical &\leftarrow physical[intermediate]\\
physical &\leftarrow 0\\
\\
intermediate &\leftarrow intermediate|intermediate\\
intermediate &\leftarrow intermediate[intermediate]\\
intermediate &\leftarrow intermediate[virtual]\\
intermediate &\leftarrow 0\\
\\
virtual &\leftarrow virtual|virtual\\
virtual &\leftarrow virtual[virtual]\\
virtual &\leftarrow 0
\end{aligned}
$$

where *physical*, *intermediate* and *virtual* represent instances of container classes Physical, Intermediate and Virtual or any inheriting classes respectively. Note that the state of the world definition above obeys the “may contain” relationship for the respective containers. Thus, any state described following the above rules is well-structured. Observe that this syntax is similar to the subset of the ambient calculus [CG98] which describes solely the structure of space with no active processes, like in semistructured data format described in [Car99].

The above definition has little practical value in cases where “may contain” relationship is defined for more specialised containers. Rules for well-structured, syntactical, representation of the state of the world would then have to be specified in terms of container classes and their respective states for which the “may contain” relationship is explicitly specified in the container ontology. The above set of rules illustrates the general approach.

Figure 3.5 represents a snapshot of a partial state of the world representing two containment trees. The container tree on the left shows a couple of mobile devices, a PDA and a laptop, physically contained within a office. The other containment tree represents a mobile phone not present in the office or not identifiable as such. The containment trees also represent the relevant internal “configuration” of the mobile devices at the point in time.

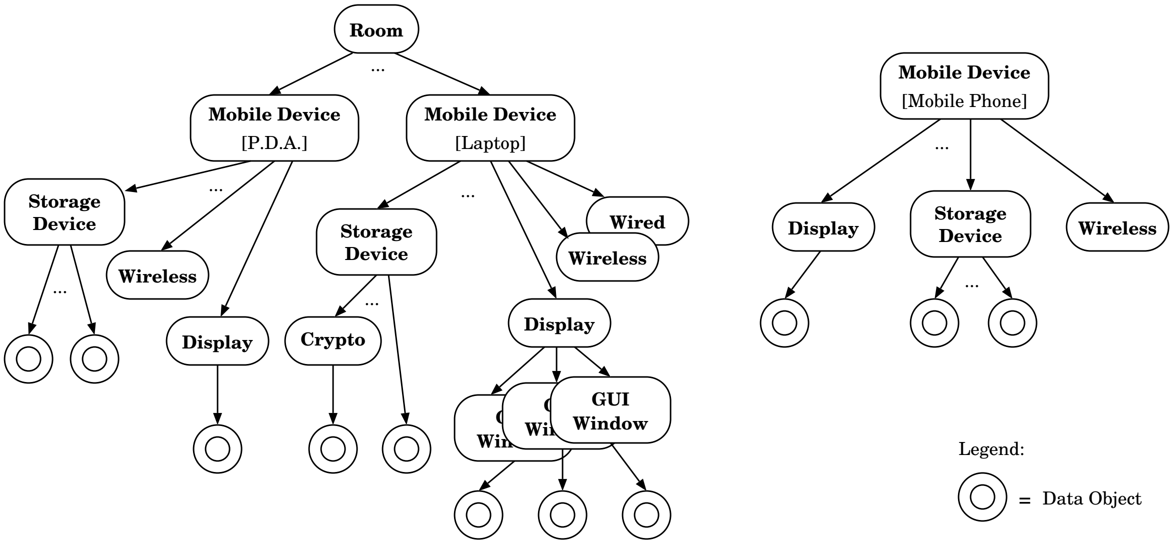


Figure 3.5: Snapshot of a partial state of the world.

Note that a single container in the model may represent multiple corresponding real world entities that are of the same class and exhibit the same state.

The Congruence Relation

Abiding by the conventional mobility theory, we also define a congruence relation, written as $\equiv$. The congruence relation is important for policy matching in the policy model introduced in Chapter 5. Under the $\equiv$ relation, containment trees are equivalent up to a simple rearrangement of parts and up to a value of a unique container identifier, if specified. Not only is the congruence relation reflexive, symmetric and transitive both horizontally ($X \equiv Y \wedge Y \equiv Z \Rightarrow X \equiv Z$) and vertically ($X \equiv Y \wedge \alpha \equiv \beta \Rightarrow \alpha[X] \equiv \beta[Y]$) but it is also commutative ($X|Y \equiv Y|X$), associative ($X|(Y|Z) \equiv (X|Y)|Z$), and “ignores zeroes” ($X|0 \equiv X$).

3.4.4 Containment Path Expressions

To be able to reference a container within a containment model we introduce *containment path expressions*, or simply path expressions. The practical significance of the path expressions for this thesis is within the policy model introduced in Chapter 5. A containment path can be defined as a sequence of containers linked by the *contains* relationship. Path expressions are specified using the following syntax:

$$
\begin{array}{rcl}
\mathit{element} & \leftarrow & \mathtt{a}[:\ \mathit{attr\_list}] \\
 & | & \star \\
\mathit{attr\_list} & \leftarrow & \mathit{attribute} \\
 & | & \mathit{attr\_list}, \mathit{attribute} \\
\mathit{attribute} & \leftarrow & \mathrm{ATTR\_NAME} = \mathit{value\_range} \\
\mathit{value\_range} & \leftarrow & \mathit{value} \\
 & | & \mathit{value\_range},\ \mathit{value} \\
\mathit{value} & \leftarrow & \mathrm{VAL} \\
 & | & \mathrm{VAL} - \mathrm{VAL} \\
\mathit{path} & \leftarrow & \mathit{element} \\
 & | & \mathit{path}\ /\ \mathit{element} \\
 & | & \mathit{path}\ /\ ...\ /\ \mathit{element}
\end{array}
$$

Terminal symbols in the above syntax are written in capital. The term ATTR_NAME is a container class attribute identifier while the term VAL is an attribute specific domain value identifier.

A matching set of a path expression is determined as follows:

1. A trivial expression element `a` matches a container of class `a` or of a more specialised class.

2. Expression element $\mathtt{a} : \mathit{attr\_list}$ matches a container of class `a` or of a more specialised class *iff* for all attributes in the *attr_list* the values of corresponding attributes for the container `a` match those specified in the *attr_list*. To match, an attribute value needs to be equal to or within a range of the values for the respective attribute in the *attr_list*. The comparison relations are value type specific.

3. Expression element $\star$ matches a container of any class and any attribute valuation.

4. Expression e_1/e_2 matches $\mathtt{a}_1 \rightarrow \mathtt{a}_2$ if e_1 matches $\mathtt{a}_1$ and e_2 matches $\mathtt{a}_2$. If the optional *attr_list* is specified for any of e_1 and e_2 rule 2 above applies individually.

5. Expression $e_1/.../e_n$ matches $\mathtt{a}_1 \rightarrow ... \rightarrow \mathtt{a}_n$ if $\forall i,\ 1 < i \leq n$ every e_{i-1}/e_i matches $\mathtt{a}_{i-1} \rightarrow \mathtt{a}_i$ according to the previous rule.

Unique container instance identifiers (or the lack thereof). In general, the containment matching capability of path expressions is not geared toward referencing unique containers in terms of individual real world entities that they represent. Thus the absence of an explicit referral to a unique container identifier of any form. Rather, path expressions are designed to refer to all containers and containments affected in a similar manner by an information exposure threat — a property of container class and the relevant state of its instances. The design rationale, made heavy use of throughout Chapter 5, is in the simplification of the threat mitigation process. It also has substantial consequences on the model representation size overheads as otherwise each individual instance of a relevant real world entity would require a separate container representation due to the unique valuation of its attributes.

3.4.5 Model Maintenance - Realms and Authorities

Although we talk about modelling "the state of the world" we do not envisage a holistic containment-based picture of a ubiquitous system to exist in a centralised fashion at any point in the system. This would be in a direct collision with a number of ubiquitous and autonomic computing characteristics and requirements. The model is devised to be established, maintained and referred to for information exposure threat, or more widely contextual effect, reasoning in a highly distributed, independent and autonomic fashion, across individual ubiquitous computing platforms, in units representing only small portions of what would be a true holistic "state of the world". Each of the "portions" represents the containment "configuration" of, and is only relevant to, the individual ubiquitous computing platforms themselves.

Definition. *A model authority, or simply authority, is a ubiquitous computing platform resource-capable to establish, maintain and do the information exposure threat reasoning in a portion of the containment based model of the world. A model realm, or simply realm, is a portion of the containment based model of the world maintained by a single model authority.*

Individual realms are not envisaged to overlap. However, they are not constrained to representing individual ubiquitous computing devices. In a temporally and spatially tightly bound group of devices, such as in a *Personal Area Network* (PAN) [Zim96], realms of resource-potent platforms may encompass the devices incapable for fulfilling the model authority role themselves. We refer to a group of devices bounded in such manner as a *collaboration group.*

Figure 3.6 depicts model realms and authorities for the partial model of the world snapshot from Figure 3.5. Individual realms in the figure are enclosed in the dash-lined squares and labelled to denote their respective authorities: the *PDA*, the *Laptop*, the *Mobile Phone* and the *Location Service* respectively.

The granularity at which a model authority models its realm depends on its model establishment and maintenance as well as its computational capabilities. To model and manipulate

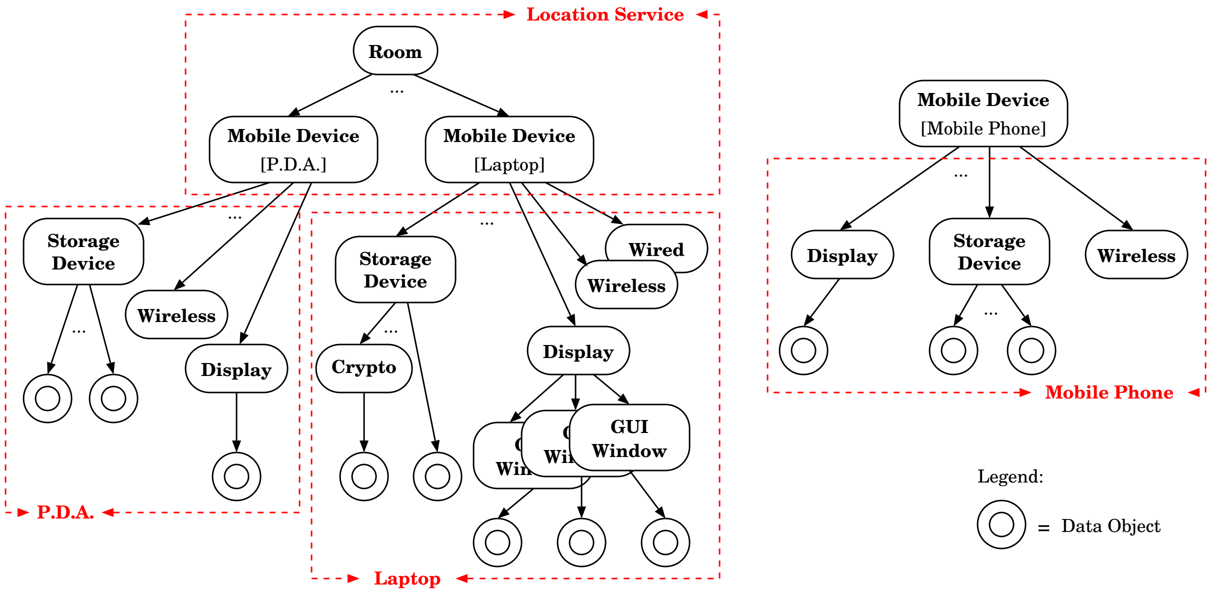


Figure 3.6: Partial snapshot of the Model of the World state.

the full range of Physical, Intermediate and Virtual containers, an authority needs support and awareness at both the system and application layers of its respective software platform.

We also differentiate between *component realms* and *component authorities*. Component authorities represent processes — active entities within a single software platform, that are "responsible" for one or more containers. Apart from providing information necessary for container modelling, component authorities may also provide for *container manipulation operations*, as will be introduced in Section 3.5. Component realms, in analogy to model realms, are non-overlapping portions of a realm that are the responsibility of a single component authority.

3.4.6 Model Update Operations

To reflect dynamic changes in the configuration of the world we provide for updating the model through four operations: `enter`, `leave`, `migrate` and `state_update`. All of the operations are assumed to be relative to a single realm.

The *enter* operation reflects a container or a containment tree entering a realm. It is defined as:

$$\text{syntax: enter(ctree } ct_{\mathtt{a}}, \text{ cpath } cp_{\mathtt{b}}) \qquad \text{rule: } \mathtt{a}[X] \mid \mathtt{b}[Y] \xrightarrow{t} \mathtt{b}[\mathtt{a}[X] \mid Y]$$

The operation accepts two arguments, full specification of the containment tree entering a realm ($ct_{\mathtt{a}}$) and the path expression identifying the target container (of the realm) under which it should be "attached" ($ct_{\mathtt{b}}$). The rule specifies, in the containment expression notation, how the enter operation behaves. In the left side of the rule, $\mathtt{a}[X]$ corresponds to the $ct_{\mathtt{a}}$ argument of the operation's syntactic representation. Node $\mathtt{b}$ represents the container that the $cp_{\mathtt{b}}$ path expression points to.

Updates to the state of the world are represented by the labelled transition relation, written $\xrightarrow{t}$. The label t on the transition symbol ($\rightarrow$) denotes a side-effect of the operation that we call a *trigger*. A trigger propagates up the affected containment from the point of update to the tree root — direction opposite to the tree edge orientation. At each container encountered on the path, the trigger can be *matched* by none or more operations — "triggering" their execution. We leverage triggers in Section 4.5.3.

The *leave* operation reflects a container or a containment tree leaving a realm. The operation is defined as:

$$\text{syntax: leave(cpath } cp_{\mathtt{a}}) \qquad \text{rule: } \mathtt{b}[\mathtt{a}[X] \mid Y] \xrightarrow{t} \mathtt{b}[Y]$$

The *migrate* operation reflects a change of containment for a container or a containment tree within a single realm. It is defined as:

$$\text{syntax: migrate(cpath } cp_{\mathtt{a}}, \text{ cpath } cp_{\mathtt{b}}) \qquad \text{rule: } \mathtt{c}[\mathtt{a}[X] \mid Z] \mid \mathtt{b}[Y] \xrightarrow{t} \mathtt{c}[Z] \mid \mathtt{b}[\mathtt{a}[X] \mid Y]$$

The *state_update* reflects the changes in attribute values of individual containers:

$$\text{syntax: update(cpath } cp_{\mathtt{a}}, \text{ attr\_list list)} \qquad \text{rule: } \begin{cases} \mathtt{a}[X] \xrightarrow{t} \mathtt{a}[X] & \text{single } \mathtt{a}; \\ \mathtt{a}[X] \xrightarrow{t} \mathtt{a}[X] \mid \mathtt{a}[X] & \text{multiple } \mathtt{a}; \end{cases}$$

The *list* argument represents a set of $(attributename, value)$ pairs to be updated for a container pointed to by cp_{α}. The latter case occurs when the `state_update` operation affects only one of multiple real world entities represented by a single container $\mathtt{a}$. It effectively leads to the duplication of the containment tree rooted by the affected container — the tree copies differ only by the attributes of the respective container.

We say that the model update operations "reflect" rather than "cause" containment model state change. As such, they do not play an active role in authorising an actual update. One of the consequences is that they do not guarantee the well-structuring of the model of the world. They may, however, be used for indicating any inconsistencies for audit purposes. Having said this, the model update operations can be used in a proactive fashion, prior to the actual real world state change, to allow or deny the real world actions by analysing the resulting model of the world.

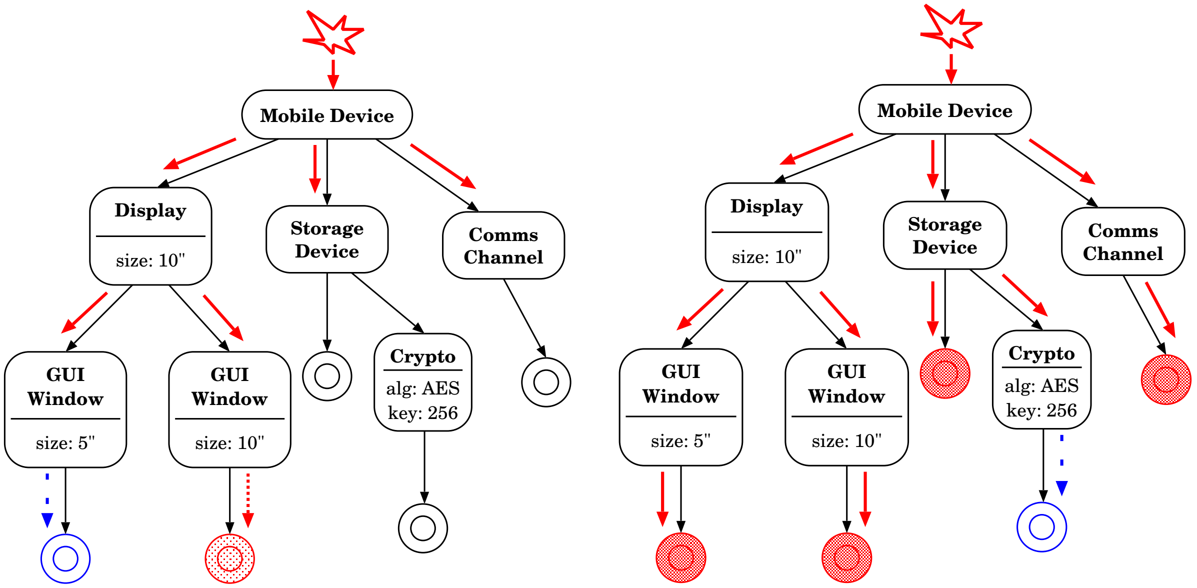
3.5 Contextual Effect Propagation and its Consequences

3.5.1 Contextual Effect Propagation

Given an information exposure threat and its source container in a containment model, the concept of container transparency in conjunction with the structure of the containment model allow us to reason about severity of the threat as experienced by the leaf nodes — the data items. We refer to this reasoning process as *threat propagation*. The process consists of traversing the containment tree, starting with the threat source node, by following the directed edges and, for each encountered container, applying the transparency function to the threat as experienced by the container. The process stops either at the leaf nodes — indicating data item exposure, or when the degree of the threat is reduced to an insignificant value.

Let e be a contextual effect and $\{t_1, \ldots, t_n\}$ a set of corresponding transparency functions associated with containers (enumerated as $1, \ldots, n$) on a single threat propagation path down a containment tree. Overall transparency of the data item containment described by the propagation path is simply the composition of the individual transparency functions. For information exposure threat e the containment transparency is given by:

$$t(e) = t_n \circ t_{n-1} \circ \ldots \circ t_1(e)$$



(a) *Optical* threat type.

(b) *Physical* threat type.

Figure 3.7: Example threat propagation in a realm.

Figures 3.7(a) and 3.7(b) illustrate an example propagation of threats of types Optical and Physical, respectively, in a realm representing a mobile computing device. In both figures the threats are assumed to occur outside the container denoting the mobile device itself — in its environment. For example, the threats inherent in a public place setting of the motivating example from Section 3.3.1 and motivating scenarios of Section 1.1. The changing solidity of the arrows denoting threat propagation reflect the impact of the transparency of the encountered containers. Arrows coloured blue represent insignificant exposure degrees which can safely be ignored.

In Figure 3.7(a) we can see that the threat of type Optical is fully blocked by the de-facto transparency of Storage and Communications Channel containers. The figure also depicts the difference in transparencies of the GUI window containers based on their size. Figure 3.7(b) shows that the experienced Physical exposure threat affects all data items except for the one that is stored encrypted. The figure also illustrates how the concepts introduced in the chapter so far contribute to the formalisation of the intuitive information exposure threat reasoning process from the motivating example in Section 3.3.1.

Upward Threat Propagation

Contextual effect propagation, as presented, does not account for contextual effects occurring within a container's interior propagating to its exterior. This would be characterised as "upward" threat propagation in a containment tree. In the real world, a contextual state within a container may well give rise to information exposure threats affecting data items not necessarily contained within the container — the neighbouring containers. Simple examples would be Acoustic information exposure among adjacent offices. A slightly subtler example are threats originating from a network interface. However, allowing for explicit upward threat propagation would have considerable consequences on the complexity of the threat mitigation approach presented in Section 4.5 and related mechanisms.

Therefore, we resort to indirect upward threat propagation through container contextual state modelling. We reflect the relevant contextual states within a container's interior in the overall state of the container itself. This, in turn, influences contextual state within the parent container. In this way, the actual context — threat correlation process is performed at a level in the containment tree from which the standard threat propagation process can cover all portions of the realm realistically affected by a threat. As the information on the actual threat origin is thus lost, so is the ability to mitigate it at, or close, to the origin. However, this is in-line with our approach of pushing protection toward threatened entities (data items).

3.5.2 Controlling Threat Propagation - Threat Mitigation Operations

Not only does the containment based model of the world allow for continuous spatial and temporal reasoning about information exposure threats data items are affected by, but it also provides for reasoning about localised threat mitigation.

Definition. *A threat mitigation operation is any operation whose execution causes, directly or indirectly, reduction in information exposure degree as experienced by a piece of information or otherwise decreases quality and/or quantity of exposed information experiencing a threat.*

We sometimes refer to threat mitigation operations as *protective actions*. Based on their targets of execution and level of indirection in the threat mitigation process, we distinguish between two broad categories of protective actions as *container manipulations* and *information manipulations*:

- *Container manipulation.* The natural way to mitigate threats in the proposed model is to exploit the relationship between container state and container transparency. In order to reduce exposure degree as experienced at the data item level in a containment tree to what can be safely ignored, the container manipulation operations affect state of containers and/or containment configuration of a realm. Thus, they indirectly affect one or both of the likelihood of threat materialisation and information exposure channel width aspects of the exposure degree. We further distinguish between three classes of container manipulation operations:
  - *Container modification* actions modify state of a container that exists in the threat propagation path to decrease its transparency for the threat type. The alteration of a container state assumes modification to the values of its transparency relevant attributes. An example of a container modification is GUI window resizing — impacting on the likelihood of materialisation of a threat of type Optical through decreasing visibility radius of the exposed information.
  - *Container insertion* operations involve instantiation of a new container and its placement somewhere on a threat propagation path toward one or more of the exposed data items. The effect is akin adding a level of filtering for the exposure threat. Example of a container insertion is data encryption — enclosing a data item in a container of class Crypto.
  - *Containment migration* operations are aimed at relocating threat exposed data items, possibly with a portion of their containment, to a containment not affected by the threat. Examples of containment migrations are: GUI window migration among displays of different sizes, hand-offs among communications links of different inherent relevant information security properties and use of alternative data input devices.

- *Information manipulation.* Information manipulation actions, unlike the container manipulation actions, do not come as a property of the presented model. They are aimed at increasing information exposure "tolerance" of exposed pieces of information through reductions in their representational quality or information content quantity.
  - *Information reduction* actions are aimed at reducing information content of data items. They encompass various types of content reduction, such as in [HL98], or information obfuscation, e.g. [BPB+04], usually dependent on data type and format. A simple example of information reduction operation is image quality degradation.
  - *Information subsetting* operations involve complete omission of threat affected pieces of information. Example is map feature selection as in [CSD01] [Cha02], albeit based on a set of criteria different to information sensitivity.

We will show, in Chapter 4, how the concept of information manipulation operations can be subsumed in that of container manipulation actions.

Threat mitigation operations, in general, can be thought of as changes in data handling and management procedures. As a lead into the next chapter, we wish to emphasise the fact, by now intuitive, that by carefully choosing data management and handling procedures for threatened pieces of data we can successfully mitigate information exposure threats. This is implicit in the above examples and in the motivating scenarios of Section 1.1.

3.5.3 Information Exposure Threat Characterisation Revisited

In Section 3.2 we have defined primitive threat types and we have assumed them throughout the chapter so far. Having presented the relevant material, we now finalise the definition of information exposure threats by introducing *compound* threat types. The definition of compound threat types rests on what we call *common* threat mitigation operations. For a mitigation operation to be *common* for a number of threat types, its execution has to always have an impact on all of them simultaneously. The impact itself can differ, as shall be accounted for in Section 4.5. The definition of the compound threat type is recursive, as follows:

Definition. *Compound information exposure threats represent a level of abstraction over the primitive information exposure threats. They are defined by existence of threat mitigation operations which affect two or more threat types, themselves primitive or compound, simultaneously. Thus derived threat types form a hierarchical structure.*

The requirement that the threat types form a hierarchy underpins the threat mitigation approach, and the supported concepts, presented in Chapter 4. Figure 3.8 shows a provisional threat "typification" as used for examples throughout the rest of the thesis. The typification is considered provisional as in practise we expect it to be defined in an application specific

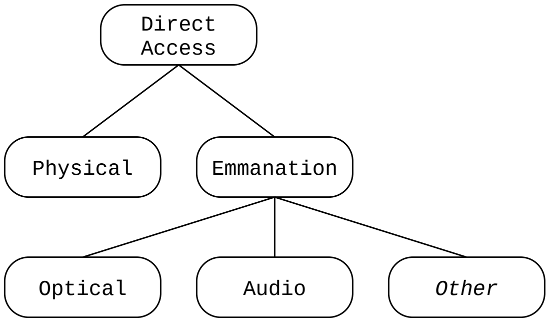


Figure 3.8: Example threat "typification".

manner. Note also that a hierarchy does not necessarily need to take a form of a tree as in the figure. Below we provide explanation of the individual types from Figure 3.8.

- *Direct Access* represents the most general threat type. Mitigating operations associated with the *Direct Access* threat type affect one or more of the information content quality, quantity or representational form of a threat exposed data item. As such, they do not in any way impact on relevant properties of a specific information exposure channel. Rather, they affect availability of information through information exposure channels in general. Consequently, we can say that Direct Access threat type mitigating operations affect all other, information exposure channel bound, threat types. Most notable examples of mitigating operations associated with the Direct Access compound threat type, as used in this thesis, are information manipulation actions, data item container manipulation actions and the insertion of a Crypto container (i.e. data encryption).

- Information exposure threats of type *Physical* arise solely from the state of context in the physical world. As stated previously, they denote likelihood of a physical or intermediate container coming under physical command of an adversary. Rather than being associated with a particular exposure channel, threats of this type are meant to signify their irrelevance once a piece of information is physically in hands of an adversary. Corresponding mitigation operations are targeted at entities in the physical world. As an example, apart from the obvious threats of mobile and storage device abductions, wire-tapping can also be considered a Physical information exposure threat. This is as it requires direct physical manipulation of a container representing a wired communications link.

- *Emanation* threat types are associated with information exposure channels involving free-space information propagation. Emanations may represent a primary information flow channel or they may arise as its side-effect. An example of the latter are electro magnetic fields created around wired communications links, displays or other electronic devices — giving rise to, so called, *Tempest* attacks [Age82]. In our model, they are enabled by specific properties of entities that are represented as Physical and Intermediate containers (e.g. a display) and are expected to be mitigated at the corresponding levels.

- *Optical* emanations occupy human visible portion of spectrum. They may involve direct [TC03], line-of-sight, as well as indirect, *Soft Tempest* [KA98] [Kuh05], information exposure.
- *Audio* emanations, in analogy, occupy the audible portion of the frequency spectrum.
- *Other* is a generic emanations threat type which subsumes various free space communications technologies in use, e.g. IEEE 802.11x, Bluetooth, GPRS etc. We expect this threat type to be replaced with individual primitive emanation threat types grouping together channels of similar propagation characteristics, thus addressable by the same set of mitigation operations.

Threat establishment and propagation processes are always expressed strictly in terms of primitive threat types. The compound threat types play a role solely in the threat mitigation reasoning process, as presented in Chapter 4.5.

3.5.4 The Join Algebra

In the general case, threat occurrences of different types are not mutually exclusive. Having said this, they may be correlated. In other words, context-threat mapping is many-to-many. To formally support reasoning about co-occurring threats we specify a simple threat composition algebra. The algebra is specified around the *join* operator, written as $\oplus$, and is thus named the *join algebra*. We make extensive use of the algebra for the threat model analysis and mitigation processes detailed in Chapter 4. The following tables summarise the algebra.

$\oplus$	$e_1^{t_1}$	$e_1^{t_2}$
$e_2^{t_1}$	$max(e_1^{t_1}, e_2^{t_1})$	$[e_2^{t_1}, e_1^{t_2}]$
$e_2^{t_2}$	$[e_2^{t_2}, e_1^{t_1}]$	$max(e_2^{t_2}, e_1^{t_2})$

Table 3.1: Join operator & primitive threat types

Table 3.1 defines behaviour of the join operator ($\oplus$) for threat types that do not exhibit super- sub- threat type relationships as specified by threat typification. Each table element of the form $e_n^{t_x}$ represents exposure degree of an individual instance n of a threat type t_x.

The square brackets, [], denote a vector. The table shows that exposures of different, non-overlapping, threat types are not "composable" — the result of the join operation is a vector consisting of the values of the individual exposure degrees. This preserves information on types of occurring threats which is required in the threat mitigation and related processes presented in Chapter 4. As for co-occurring threats of the same type, the algebra definition reflects the fact that joint exposure is no more significant than the highest degree individual exposure of the type.

$\oplus$	e^{t_1}	e^{t_2}	$[e^{t_1}, e^{t_2}]$
$e^{c(e^{t_1}, e^{t_2})}$	$max(e^{c(\ldots)}, e^{t_1})$	$max(e^{c(\ldots)}, e^{t_2})$	$max(e^{c(\ldots)}, max(e^{t_1}, e^{t_2}))$

Table 3.2: Join operator & compound threat types

Table 3.2 defines behaviour of the join operator for threat types that exhibit sub- super-type relationship. Table elements e^{t_1}, e^{t_2} and $[e^{t_1}, e^{t_2}]$ preserve their semantics from Table 3.1. $e^{c(e^{t_1}, e^{t_2})}$ represents exposure degree of a compound threat type that is a super-type for primitive threat types t_1 and t_2. What this table demonstrates is that compound threat types are "forgetful" about the sub-types. Such behaviour stems from the fact that by mitigating exposure of a compound threat type we simultaneously address exposures of all respective sub-types.

3.6 Intensity Reduction Property

We use the term *intensity* to mean exposure degree in the context of information exposure threats. Let c be an instance of any container class and e be an information exposure threat. Then the *Intensity Reduction Property* (IRP) may be defined as:

$$degree(t(c, e)) \leq degree(e)$$

where t denotes container transparency function. In other words, after crossing a container boundary, information exposure threat can never gain on its intensity. This is implicit in the way we introduced container transparency in Section 3.3.2. IRP is relied upon by the approach that we propose for threat mitigation in the next chapter.

A corollary of the above is:

$$degree(t_{c_n} \circ \ldots \circ t_{c_1}(e)) \leq degree(t_{c_i} \circ \ldots \circ t_{c_1}(e))$$

where $1 \leq i \leq n$ and $t_c(e) \equiv t(c, e)$. In other words, the highest intensity an information exposure threat can ever have is the intensity as determined at its source. Note that the IRP does not judge the correctness of the container transparency definition, with respect to the real world entities they model, or its precision. IRP solely formalises the property of the proposed approach to modelling the world which states that degree of a threat cannot increase passively as the threat propagates through a realm.

The following two sections outline a number of ways in which a resource deprived ubiquitous computing platform can keep realm modelling complexity at bay. We also show that the IRP holds universally, irrespective of the method used for, or the extent to which, a model is simplified to match a platform's capabilities.

3.6.1 IRP & the Modelling Granularity

As stated previously, Section 3.4.5, the containment model is intended to be established and maintained in an independent fashion across ubiquitous computing platforms at a granularity suiting individual platform's resources and capabilities. The question that naturally arises under such circumstances are the consequences of variable model granularity on information exposure threat reasoning. We see the containment modelling granularity as having two facets, *vertical* and *horizontal*, denoting respectively its:

- Level of abstraction, and
- Level of detail.

Level of Abstraction. The level of abstraction is determined with respect to container classification, relative to the "is-a" relationship. The higher a container class is in a classification the more abstract we say it is. For example, if container of class Communications Channel is used instead of that of class Wired, for a corresponding real-world entity, we say that the realm is modelled at a higher level of abstraction.

We see the level of abstraction as a vertical component of containment modelling granularity. The consequence of increasing the level of abstraction is a potential decrease in the specificness of relevant container transparency attributes and, consequently, coarser grained transparency modelling. Therefore, we may say that increasing the level of abstraction of a containment model is, in general, safe as it can result in exposure degree over-estimation but not under-estimation.

The Level of Detail. Reducing the level of detail of a containment model implies omitting containers, rather than substituting them for containers of more abstract classes. Thus we see it as *horizontal* component of the containment modelling granularity. An example would be modelling Data Item containers as contained directly within Storage Device and Display containers without accounting for the existence of Crypto and GUI window container classes respectively.

Let `a` and `b` represent any two containers such that `containable(a,b)` holds and let e be a contextual effect. Then, from the IRP definition and the corollary, we have:

$$degree(t(a,\ t(b,e))) \leq degree(t(a,e)) \leq degree(e)$$

In other words, should we omit any of the containers a or b from the containment model, the subsequent threat analysis is guaranteed not to ever under-estimate the degree of threat experienced. The potential threat overestimation, however, is considered safe.

In a nutshell, decreases in modelling granularity, both vertical and horizontal, can never result in information exposure threat degree under-estimation. This, in turn, means that no matter how resource (in)capable a ubiquitous computing platform is never shall it cause information security under-provisioning with respect to the proposed approach.

3.6.2 Container Fusion

Lowering the level of container granularity, vertical as well as horizontal, at which a realm is modelled results in lowering the respective model complexity in terms of the quantity of containers to be represented and, consequently, in terms of computational complexity involved in threat propagation as well as threat mitigation processes, presented in Section 4.5. The third method for addressing modelling complexity of a realm is container *fusion*.

Definition. *Container fusion is approximation of a number of containers using a single container with similar transparency characteristics.*

While a single container represents one or more corresponding real world entities that are in the same state, the result of a container fusion is a container that represents multiple real world entities of different states and of, possibly, different container classes. An example of container fusion would be using a single *Display* container to represent two *LCD* containers of different sizes and viewing angles or a *LCD* and a *CRT* container. Container fusion may involve model vertical granularity decrease. Container fusion is appropriate for longer-lived, tightly coupled containers as otherwise it may incur significant realm maintenance costs for the model authority.

For a container fusion to be “safe” the transparency for any of the threat types of a container representing the fusion must at no point in time be less restrictive than the most restrictive corresponding transparency of any of the fused containers:

$$degree(t(c_{fusion}, e)) \geq \max_{c \in c_{fusion}}[degree(t(c, e))]$$

This defines the Information Reduction Property with respect to the container fusion process.

In summary, in this section we have shown the proposed model may be leveraged on ubiquitous computing devices exhibiting highly variable degrees of resource poverty without ever compromising security of information placed under its jurisdiction.

3.7 Summary

The contribution of the material presented in this chapter is three-fold. Firstly, we have finalised the definition of information exposure threats, initiated in Chapter 1, through their characterisation and, thus, systematisation. Secondly, we have introduced, in a semi-formal manner, an original approach to modelling the world based on the notions of *container*, as a protective enclosure, and *containment* as its higher level structuring. The concept of a container, in a manner in which it is defined, allows for the containment based model of the world to represent relevant entities in the physical as well as in the virtual worlds in a uniform way. We have also set foundations for the development of a container ontology through defining container classification and a couple of relationships. The model has been conceived to form the basis for a third contribution of the chapter — a novel approach to reasoning about information exposure threats. We have introduced *transparency* characteristic of a container as a main instrument for fine-grained threat analysis. The estimation of the transparency for various classes of containers as well as for different threat types has been not been considered as it is out of the scope at this level of abstraction. Further, we have shown how the model enables reasoning about localised threat mitigation and have introduced two classes of threat mitigation operations. The manner in which the model is defined caters fully for the autonomy and resource diversity of target ubiquitous computing devices through its structuring into fully independent management units referred to as *realms*. A number of related formalisms that are made use of throughout next chapters have also been introduced.

Chapter 4

Information Security vs. Utility: Balancing the SeeSaw

4.1 Chapter Overview

In this chapter, we focus on the tools and techniques that enable dynamic balancing between the level of information exposure protection provided and its adverse side-effects, with respect to the information utility and wider system usability impact thus incurred — the process that we refer to as balancing the information security vs. utility "seesaw". The importance of balancing the seesaw lies within user acceptance of the ubiquitous computing vision as a technology that "waves itself into the fabric of everyday life" [Wei91]. We also present a dynamic programming approach to balancing the seesaw and the supporting reasoning structures, a major contribution of the chapter.

In Section 4.2, we introduce the concept of *information utility*, with reference to the wider notion of utility as used in economics and philosophy, and present its facets and the contributing factors. In Section 4.3, we contrast the notion of information exposure with the familiar notion of risk. We show how reasoning about information exposure from the risk perspective is utilised to categorise information exposure degree values into *significance levels* — the basis for fine-grained reasoning about the threat severity. In Section 4.4 we introduce the concept of a *Level of Exposure* (LoE) as a basic building block for the *LoE Model* — the main instrument for matching the severity of experienced threats to the level of protection provided by available threat mitigation operations. The Optimal Cover Determination (OCD) algorithm, Section 4.5, leverages the LoE model to arrive at the optimal, information utility and system usability wise, threat mitigation strategy for a holistic realm in the face of a set of information exposure threats. The computational characteristics of the algorithm make it attractive for deployment on potentially resource challenged ubiquitous computing platforms.

4.2 Information Utility

Although it is not clear which field of study the concept of *utility* originates from it seems that it has lately been most widely studied and, most certainly, exploited in the field of economics and related sub-fields like micro- and macro-economics, behavioural economics, market theory etc; and also, with shared semantics, in the field of game-theory [VMK04]. The concept of *utility* also has deep roots in philosophy owing to the doctrine of *Utilitarianism* brought about by Bentham and Mill [Mil63] — where utility is seen as a moral criterion for the organisation of society.

The meaning of the term *utility*, common to all of its mentioned uses, can be phrased as: *utility is a measure of happiness and/or satisfaction gained from a good or service.* In this form, reflecting its use in economy, the definition does not specify the subject whose happiness and/or satisfaction is measured — the service/good user(s) or its provider(s). Utilitarians argue for the overall happiness of the community rather than any individual subject; the economists take the individualist view. In economy, the concept is most often used to characterise a set of commodities required for a certain level of happiness or satisfaction — standardly modelled by utility functions and expected utility [MCWG04].

Cardinal vs. Ordinal Utility

The economic literature distinguishes two major kinds of utility measurements as *cardinal* and *ordinal* utility.

The concept of *cardinal utility* is based on the assumption that utility can be seen as a measurable quantity with globally uniform semantics. In other words, that it is possible to measure the utility of each individual in the society toward each commodity, or a set of commodities, and then sum these to obtain the total utility the commodities bring to the society as a whole. The concept, consequently, facilitates decision making in a manner that maximises the total utility across a society. We refer to the concept of cardinal utility as *objective utility* for its impersonal nature which allows for the generalisation. Economists usually measure cardinal utility in terms of monetary units.

A general critique to the concept of cardinal utility is in its inherent inability to reason about subjective aspects of the "happiness" a good or service brings to an individual. For this reason, certain economists abandoned the concept of cardinal utility for the theory of *preferences*. In the preferences theory, individual is seen as simply preferring one commodity, or choice, to another. There is no attempt to quantify individual choices beyond what is sufficient for their ordering. No attempt is made to explain the choices either. Economists usually describe the preferences using *utility functions*, an instrument that takes higher values for choices an individual prefers. This approach to utility measurement is also known as *ordinal utility*. The preference based view of utility, however, lacks the ability to compare utilities between individuals or express the total utility a commodity brings to a society as whole. We refer to it as the *subjective utility*.

4.2.1 What is Information Utility?

In analogy to the economics view on utility we express *information utility* as *a measure of happiness or satisfaction gained from a piece of information* — substituting the term *good* for the term *piece of information*.

We say that a gain from a piece of information is maximised when it is available *where*, *when*, in the *form* and by the *means* required by a user for completion of a task. As such, information utility can be seen as a measure of information omnipresence as referred to in Chapter 1. The former two criteria refer to spatio-temporal aspects of information availability respectively. The *form* in which information is available refers to its data representation. The notion of the *means* through which information is available can be linked tightly to the concept of *usability* as used by human-computer interaction researchers. For example, the *means* may refer to the user interface or the physical display.

The latter two criteria cross the boundary between the objective and subjective utility measure and make the concept of information utility inherently subjective. To illustrate this, consider a piece of information as a prerequisite for accomplishing a task. Also assume that user's level of satisfaction is correlated with their productiveness and efficiency. Then, even if the required piece of information is available through the *means* and in the *form* objectively suitable for accomplishing the task a possible mismatch with user's individual preferences on the criteria may have an impact on their individual productiveness and efficiency — thus lowering the utility. The preferences depend on both the physiological and psychological factors characterising an individual user.

The System and User Behaviour Rationality Assumption

The initial economic utility theory relied on the assumption that the humankind was *rational* in its behaviour. In other words, that humans always act to maximise their utility whenever given an option. This has since been debated by behavioural economists and psychologists on the grounds such as individual perception of personal loss, risk aversion, etc.

To frame a number of aspects of our work we make an analogous assumption specific to the concept of information utility which we name the *system and user behaviour rationality assumption*:

Assumption. *Otherwise than with respect to information security decisions, users and systems behave in a manner which maximises the information utility whenever possible.*

In other words, whenever a user or a software component encounters a set of alternatives it always chooses the one that maximises information utility. For example, in case where constraints on a mobile device's display do not permit a full level of detail required for a document to be presented, and where an adequate external display is present, we assume a user chooses to display the information externally. Similarly, in cases where multiple communications links are available we expect the system to use the one with maximum bandwidth and minimum cost given the traffic priority. We also refer to the assumption simply as *rationality* assumption.

We do, however, recognise that the assumption is highly unrealistic considering the current state of the art in the systems and application design — as testified to by a myriad of issues raised and pursued by computer usability and human-computer interaction researchers. On the other hand, neither is CASPEr aimed at what represents the current state of the art nor does it fit the conventional system architectures (Chapter 5). We do see the assumption as an inherent part of the ubiquitous computing vision [Wei91], especially when married with the concept of autonomic computing [IBM01], — the target setting for CASPEr.

4.2.2 The Aim - Balancing The SeeSaw

Information exposure threat mitigating operations, as presented in Section 3.5, affect all aspects of information utility: the *when*, the *where*, the *means* and the *form*.

Container modification and container insertion actions affect one or more of: the *when*, the *means* and the *form*. Containment migration affects one or more of: the *when*, the *where*, the *form* and the *means*. Information manipulation operations impact either on the *form* or on the overall information existence. How exactly is clarified in the next section. As a consequence, information exposure threats can be seen as direct constraints for the overall information utility maximisation as they necessitate application of information utility affecting threat mitigation operations.

To mitigate a threat, container manipulation operations impact one or more of the threat exposure degree components, as stated in Section 3.5. Container manipulation operations accomplish this indirectly, through affecting the containment transparency. As information exposure channels are, by definition, shared with or indivisible from legitimate information flow and storage channels (Section 1.2), constraining them through reductions in containment transparency has a direct impact on information utility.

To preserve the validity of the *system and user behaviour rationality assumption* in the face of information exposure threats the application of mitigating operations has to be guided by the maximisation of the information utility while providing adequate information exposure protection. We refer to the process as balancing the *information security vs. information availability "seesaw"*.

4.2.3 Information Utility Factors

We see the following four factors as fundamental, but not exhaustive and open to extension, for characterising the information utility as affected by the information exposure threat mitigating operations:

- Information content.

- Locality of information.
- Information accessibility.
- User Perceived Quality of Service.

Information Content. Information content represents the quantity of information contained in a data item as available to a user. Information exposure mitigation operations may affect information content either directly, through information manipulation, or indirectly, through container manipulation. An example of the former is information sensitivity reduction through data obfuscation [BPB$^+$04]. As for the latter, a simple example is reduction in size of a *GUI Window* causing only a subset of previously shown information to be displayed. Every threat mitigation operation can be associated with an attribute denoting the resulting information content reduction.

Information content manipulation affects the *where* and the *when* information utility facets — as a piece of information is either there where the user needs it, at a point in time when they need it, or not.

Locality of Information. Locality of information is relevant when we consider container migration operations. Following the rationality assumption, we expect containments in which data items exist at a point in time to represent the data management and handling procedures which maximise the respective information utility. Thus, any containment migration operation may potentially incur information utility penalties.

We distinguish between six types of containment migrations, along two axes. With respect to migration *locality*, containment migrations may be *local*, *semi-local* or *remote*. Local migrations are within a single realm while remote imply crossing the realm boundary. Semi-local migrations are remote migrations among realms associated with tightly bound pervasive computing devices, such as in a *Personal Area Network* (PAN) [Zim96], that exhibit longer-term temporal and spatial associations. An example of a semi-local migration would be a GUI window migration from a laptop to a PDA within a PAN. Orthogonally, containment migrations may be among containers of a same or different classes. As an example of a semi-local migration among containers of a different class consider swapping out contents of a GUI window to the permanent storage for later, post-threat, retrieval and (re-)displaying.

Container migration operations may affect any of the information utility facets. The above example hints at how the transcoding required for a containment migration impacts on the *form* in which information exists in a system.

Information Accessibility. We define information accessibility to denote time delay and cost, resource as well as monetary, of accessing a piece of information in a particular containment. All container manipulation actions may have significant consequences for information accessibility. For example, data encryption, i.e. insertion of a *Crypto* container, incurs both an additional time delay and resource costs for subsequent information access. Furthermore, changes in information locality usually have an associated information accessibility impact.

Shifts in information accessibility caused by threat mitigation operations have an effect on the *where* and *when* aspect of information utility.

User Perceived Quality of Service. Information content, locality and accessibility constitute *objective* aspects of information utility. However, they suffer from the similar lack of *subjective* user happiness or satisfaction indication ability as the cardinal utility does. For example, while one user would happily trade off a level of information obfuscation for retaining the information on a large screen, another may favour the opposite; while one would opt for screen blurring the other may opt for GUI window shrinking, or its migration to a more restrictive display within a PAN; a user may prefer a mobile phone's keypad to PDA's touch screen interface.

In general, users may express *preferences* toward the *form* and the *means* in which information is managed and handled, especially when being presented to them for direct access. *User Perceived Quality of Service* (UPQoS) is introduced to reflect these preferences. Introduction of UPQoS causes a transition from objectivity to subjectivity in the perception of information utility.

	where	*when*	*means*	*form*
Information Content	●	●	○	○
Information Locality	●	●	●	●
Information Accessibility	●	●	○	○
UPQoS	○	○	●	●

Table 4.1: Information utility factor-aspect associations

The Table 4.1 summarises the scope of the impact each of the above information utility factors in terms of the *where*, *when*, *means* and *form* information utility facets. The "●" symbol indicates association while the "○" indicates the absence of such. Otherwise than implied by the four factors, there is no inherent interdependence between the four information utility facets themselves. For example, information may be present *when* or in the *form* needed but not *where* required should a remote, available, display be determined as the only one not exposed; similarly, switching displays implies a shift in the *means* through which information is available *where*, *when* and in the *form* required.

4.2.4 Information Utility Measure

To support the process of balancing the "seesaw" we introduce the notion of the *information utility measure*. Drawing an analogy between the concept of preference in microeconomic terms and preference in the context of the above information utility facets allows us to leverage the notion of the utility function as used in microeconomics [MCWG04] to define *information utility function*. The purpose of information utility function is to expresses the *information utility measure*. The existence of the subjective side to the information utility, as defined above, causes the obtained information utility measure to denote preferences among the alternative threat mitigation operations. The information utility function can thus be defined as associating the preference values to sequences of non-mutually exclusive threat mitigation operations. The sequences consist of one or more operations and denote their sequential execution — making the ordering of the elements of a sequence potentially significant to the preference value assignment in the general case. The information utility measure itself may be defined in terms of integers, labels or as taking values from any other domain for which the suitable ordering relations are defined.

For illustration, consider a very simple set of threat mitigation operations given as $\{blur, shrink\}$ — the former denotes blurring of a whole display while the latter represents decreasing the size of an individual GUI window. Also, each execution of the threat mitigation operations affects the respective container characteristics in a step-wise manner, where the size of each step is pre-defined and constant. With respect only to user preferences, the information utility function for the two threat mitigation operations could be defined as follows:

$$\begin{array}{rcl} blur & \leftarrow & 1 \\ shrink & \leftarrow & 0 \end{array}$$

The above signifies that the user *prefers* display blurring to GUI window shrinking in situations where the two operations represent threat mitigation alternatives. For situations where more radical threat mitigation is required, due to a higher exposure degree experienced, we could also have:

$$\begin{array}{rcl} blur \circ shrink & \leftarrow & 2 \\ blur \circ blur & \leftarrow & 1 \\ shrink \circ shrink & \leftarrow & 0 \end{array}$$

The $\circ$ symbol denotes operation composition. Note that in the above example the ordering of the operations in the sequences is insignificant. While the above focused on user preferences, the following table gives an example of the information utility measures for different containment migration operation alternatives with respect to the information locality criterion (as specified in the previous section):

	Local	Collaborative	Remote
== cclass	6	5	4
!= cclass	3	2	1

The table specifies that migrations within the same realm (*Local*) and to a destination container which is of the same class ($== cclass$) as the source container are more preferred to, for example, migrations to the destination containers of different classes ($! = cclass$).

The suggested approach to defining the information utility function is by no means the sole possible alternative. In Section 5.5, for example, we show how a probabilistic prioritising approach may work as the part of the suggested policy model. However, the analogy with the utility functions as used in economics is interesting as the concept of information utility is, thus, well-founded.

4.3 Information Exposure: The Risk Perspective

The insurance industry [AB00] and economists [Kni04] see risk as a form of uncertainty where the odds are known but one is not certain of the outcome. In the field of computer security and privacy [CSG$^+$03] [Dim03] [Lev95], *risk* denotes a potential harm that may arise from a present process or a future event and is defined in terms of the *likelihood* and *severity* of an accident.

From the economic stance, the severity of information leakage, and consequently information exposure in our terminology, is represented by the value the information under risk has to an organisation or an individual. Information value, in the economic sense, is measured as a direct or indirect monetary loss that the information compromise would inflict on the organisation or the individual. From a more general information security perspective, information compromise may have wider repercussions than solely monetary — possibly as extreme as compromise of national security or even endangering human lives.

Our view of risk is best expressed as: *the combined likelihood and severity of an accident in which data management and handling procedures are leveraged to leak sensitive information otherwise in a legitimate custody.* The term *sensitive* is used to refer to any information that can not be freely distributed, i.e. non-*public* information.

4.3.1 Risk vs. Information Exposure Degree

In relation to the above outlined semantics of risk, information exposure threat degree (Section 3.2) embeds the notion of an accident occurrence in the form of the threat materialisation likelihood. With respect to the accident severity, information exposure threat degree incorporates what can be seen as only one of its components — the exposure channel width, i.e. the relative proportion of information that is to leak should threat materialise. It does not, however, capture the other component that characterises the severity of an accident — the value of a piece of information under the threat. This represents a fundamental difference between the concepts of risk and information exposure threat degree.

The rationale behind the separation of the two is the ability to reason about the threats and the threat propagation in a manner independent of impact they may have on individual pieces of information. Risk can, thus, be seen as the consequence of an information exposure threat affecting a particular data item. Furthermore, owing to the manner in which threat propagation is defined (Section 3.5), the information about any characteristics of potentially threatened data items is not available to the process prior to encountering the leaf nodes of a containment tree. Referring back to Section 3.2, explicit information exposure threat modelling plays a crucial role in the separation of the two concepts.

We illustrate the argument by means of a simple example presented in Table 4.2, Table 4.3 and Figure 4.1.

p w	Low	Medium	High
Narrow	Low	Low	Medium
Medium	Medium	Medium	High
Wide	High	High	High

Table 4.2: Exposure degree mapping.

Table 4.2 shows how exposure degree can be derived from threat materialisation likelihood, denoted by p (column labels) in the table, and exposure channel width, denoted by w (row labels). For clarity purposes we use a three level categorisation for both the dimensions and for the exposure degree, as represented by the values in the table.

e v	Low	Medium	High
Confidential	None	Low	High
Secret	Low	Medium	High
Top Secret	Medium	High	High

Table 4.3: Risk mapping.

Table 4.3 illustrates how exposure degree (e), as experienced by threatened assets of different values (v), leads to different levels of risk. Levels of risk are categorised as *None*, *Low*, *Medium* and *High* starting by the lowest respectively. We tie the notion of information value to information sensitivity classes as determined by a wider information security classification policy. Throughout the thesis we use a simple four level, totally ordered, information security classification policy for the purpose. The classes, given in the order of increasing sensitivity, are: *Public* (non-sensitive), *Confidential*, *Secret* and *Top Secret*.

Finally, the graph in Figure 4.1 plots the relationship between information exposure degree and risk for various values of information based on Table 4.3.

The levels of measurement and the quantification steps used in the example are solely for purpose of the illustration. In practise, they would be determined by external mechanisms,

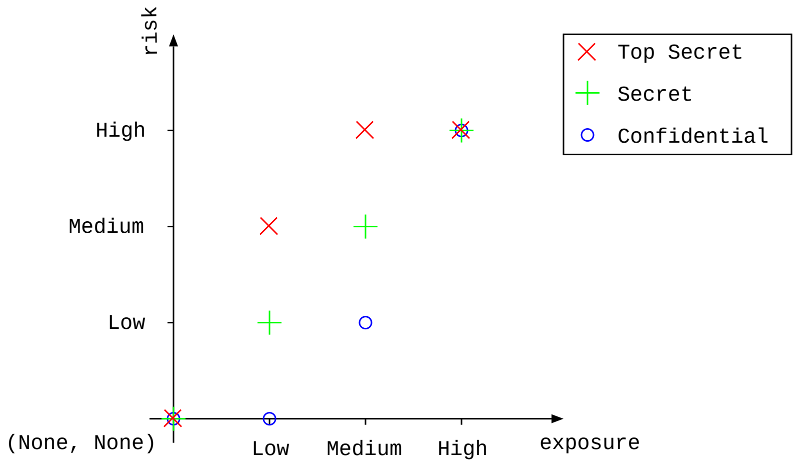
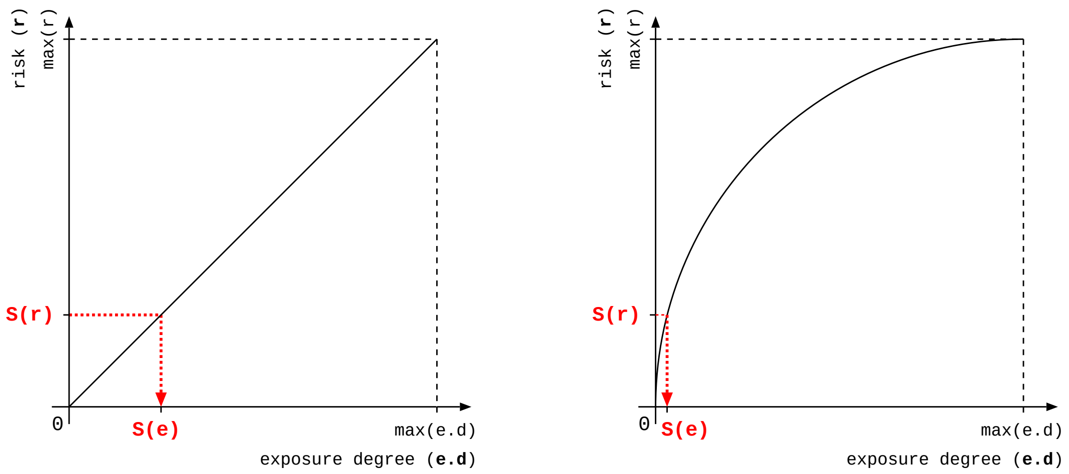


Figure 4.1: Example discrete risk-exposure plot.

such as risk modelling tools (e.g. CORAS [FKG$^+$02] or OCTAVE [oct]) and specific methods employed for context-threat correlation.

4.3.2 Information Exposure Degree Significance Level



(a) Secret sensitivity class.

(b) Top Secret sensitivity class.

Figure 4.2: Continuous risk-exposure mapping - significance levels.

Graphs in Figures 4.2(a) and 4.2(b) show two examples of continuous exposure degree - risk mapping functions. They can be seen as rough approximations of the discrete mappings for the *Secret* and *Top Secret* information sensitivity classes from the Table 4.3. Both the risk and the exposure degree scales are normalised to intervals $[0, max(e.d)]$ and $[0, max(r)]$ respectively. In the rest of the chapter we assume the continuous exposure degree and risk quantisation steps for the purpose of the generality of presented arguments. However, in practise we expect coarse grained valuation of exposure degree as discussed in Section 6.2.

In general, we distinguish between two main levels of risk as *insignificant* and *significant*. The former represents risk that may be safely ignored and for which no explicit risk mitigation operations need to be performed. The latter denotes the opposite and can be further subdivided into levels according to risk mitigation strategies in place. We also refer to the two levels as to *acceptable* and *unacceptable* risk respectively. Both of the graphs in the Figure 4.2 plot the delimiting point between the acceptable and unacceptable risk ranges as $S(r)$. Note that they have the same value in both graphs. The exact sizes and the values of the delimiting points of the two ranges are risk model specific. The dotted arrows in the graphs represent projections of $S(r)$ onto the exposure degree scale according to the respective mapping functions. Thus obtained exposure degree ranges, $[0, S(e)]$ and $(S(e), max(e.d)]$ represent *acceptable*, i.e. *insignificant*, and *unacceptable*, i.e. *significant*, exposure degrees respectively. The semantics of the two ranges are analogous to their risk counterparts.

The important point to note is that, unlike for risk, the sizes of the exposure the significant and insignificant exposure degree ranges vary based on the perceived value of threatened pieces of information. In other words, on their respective sensitivity class. This can be seen in the two plots above. The consequences shall be presented and addressed in Section 4.4.5. From now onward, our interest in the concept of risk goes only as far as the values of the acceptable and unacceptable risk levels.

4.4 Levels of Exposure

In this section, we introduce the concept of a *Level of Exposure* (LoE) — the main instrument for matching the level of protection to provide with the severity of experienced threats. Consequently, the concept plays an important role in minimising the the overall information utility, and wider system usability, impact incurred by the threat mitigation process, as shall be clarified shortly. We organise individual LoEs into the *Levels of Exposure Model* — a structure utilised in reasoning about alternative threat mitigation strategies, presented later in this chapter.

Prior to defining the LoE, we introduce a helper concept of *threat mitigation offset range* in the next section.

4.4.1 Threat Mitigation Offset Range

We say that a threat mitigation operation *offsets* an exposure degree if upon its execution the exposure degree as experienced at a container reduces to within the respective insignificant range. The *absolute offset range* of a threat mitigation operation is the exposure degree range

$(p, q]$ where p is the highest acceptable exposure degree value and q is the highest exposure degree value that the operation offsets. *The relative offset range* of a threat mitigation operation, with respect to the exposure degree q, is the exposure degree range $(p, q]$ such that p is the exposure degree obtained after the operation is applied for the exposure degree q.

The *joint absolute offset range* of a set of threat mitigation operations, $\{op_1, \ldots, op_n\}$, is the concatenation of the respective relative offset ranges of the individual mitigation operations when applied in sequence:

$$(p, i_1] + (i_1, i_2] + (i_2, i_3] + \ldots + (i_{n-1}, q(= i_n)]$$

Where each of the intervals $(i_{k-1}, i_k]$, $1 \leq k \leq n$, is the relative offset range of operation op_k with respect to the exposure degree i_k. Effectively, it is the offset range of a single mitigation operation representing the composition of the individual mitigation operations: $op_1 \circ \ldots \circ op_n$. The order of the composition is unimportant. The *joint relative offset range* for a set of threat mitigation operations is defined analogously.

In the case of multiplicative (Section 3.3.2) transparency functions, the relative offset range of a single threat mitigation operation varies in size depending on exposure degree it is relative to. For example, a 10% exposure degree impact implies the relative offset range of length 0.1 with respect to exposure degree 1 but also, the relative offset range of length 0.05 with respect to exposure degree 0.5. Additive transparency functions, on the other hand, exhibit the relative offset ranges of uniform length across the exposure degree scale. This has important consequences on the threat mitigation process, as presented later in the chapter.

4.4.2 Level of Exposure - The Definition

Our interest in the concept of the exposure threat degree lies not in itself as a detached phenomenon but in its relation to the threat mitigation operations. The relationship is established through the concepts of the absolute and the relative offset ranges characterising threat mitigation operations. Exposure degree represents a unified measure by which the severity of a threat can be matched with the level of protection as provided by one, or more, threat mitigation operations.

Informally, a *Level of Exposure* (LoE) is a semantic construct that binds a set of threat mitigation operations to a range of exposure degrees that they offset, on their own or composed. Conceptually, a LoE is fully independent from the state of a containment model of the world, in general, and the states of individual containers, including data items, in particular. We name this characteristic the *universal validity* property and we say that LoEs are *universally valid*.

Slightly more formally:

Definition. *Let O be a set of all mitigating operations available for a single threat type, primitive or compound, and let $\mathrm{P}(O)$ be the power set of O. A Level of Exposure (LoE)* *represents a (threat type specific) range in information exposure degree* $[p, q]$ *together with a set* $M \subseteq \mathrm{P}(O)$ *and a set* A *of auxiliary operations such that:*

1. $M \neq \{\emptyset\}$.

2. $\forall m \in M$, $\forall e \; p \leq e \leq q$, m *offsets* e.

3. $[\ \neg\exists m \in M, m \textit{ offsets } e \textit{ when } q < e\ ]$.

4. *For any two distinct LoEs* A *and* B, $[p_A, q_A] \cap [p_B, q_B] = \{\emptyset\}$.

We say that a LoE is defined *by* a set of mitigating operations (M) *over* an exposure degree ($[p, q]$). A LoE is *activated* upon experiencing an exposure degree that falls within the exposure degree range the LoE is defined over. No more than a single LoE corresponding to a threat of any one type may be active at point in time.

The first point of the above definition establishes the binding between mitigation operations and the exposure degree range. If the binding did not hold, it would be possible to define LoEs over arbitrary exposure degree ranges independently of the threat mitigation operations available. As such, the LoEs would present little practical value for the threat mitigation reasoning process, presented in Section 4.5. Depending on the exposure degree quantisation granularity and the spectrum of available threat mitigation operations, scenarios in which LoEs are defined on per single exposure degree value are feasible.

The second point states that any threat mitigation operation associated with a LoE has to be able to offset all exposure degrees in the range the LoE is defined over and strictly no less. Otherwise, situations could arise in which a piece of information remains inadvertently at a significant exposure degree after the completion of the threat mitigation process (Section 4.5).

The third point specifies that any threat mitigation operation associated with a LoE should not be able to offset an exposure degree greater than the upper limit of the exposure degree range over which the LoE is defined. However, as no situations may arise in which information security is compromised if this does not hold we regard it as an optional requirement (enclosed within brackets). In cases of extensive resource poverty of ubiquitous computing platforms we may benefit from increasing LoE granularity by relaxing this requirement. On the other hand, the exact matching of the experienced threat degree and the level of protection is desirable as it are likely to minimise the unnecessary information utility impact incurred by the threat mitigation process.

The fourth point above states that LoEs do not overlap. It eliminates ambiguities that would otherwise arise in the LoE activation process.

A LoE may also be associated with a set of *auxiliary operations* that are to be executed upon LoE activation. Auxiliary operations do not, directly or indirectly, influence the experienced exposure degree. Examples of auxiliary operations are local and remote logging, external event triggering or scheduling and other administrative tasks. The set of mitigating operations over which the LoE is defined, together with any auxiliary operations, form the *handling procedure* for the LoE.

4.4.3 LoEs: Threat Specificness

According to the definition, LoEs are threat type specific. This is implicit in the requirement that all threat mitigation operations as included in the set O, and consequently the derived sets $\mathrm{P}(O)$ and M, apply to a threat of the same type. From this point of view, LoEs serve the purpose of conveying information about the offending information exposure channel from the threat propagation to the threat mitigation stage.

Unawareness of threat type behind LoE activation at the threat mitigation stage would require threat mitigation operation selection process to be based solely on the criterion of matching experienced exposure degree. Without the ability to identify the specific information exposure channel to constrain the only operations to consider would be the ones that affect them all. Otherwise, we would not be able to ensure, with sufficient certainty, the threat mitigation. This would result in the ability to balance the level of information exposure protection against the information utility impact thus incurred on the fine grained basis being seriously diminished. Consequently, the threat typification itself would be rendered useless.

Common LoEs

Introducing the compound threat types (Section 3.5) we stated that their sole role is in the threat mitigation process while the LoE establishment is accomplished in terms of the primitive threat types. We identify two cases in which there is no need for LoEs being compound threat type specific. In other words, where no awareness of compound threat types is beneficial.

The first case arises where, owing to a particular container classification: i) all containers are de-facto opaque to all but a single primitive threat type; or ii) for any of the containers, there exist no threat mitigation operations such that they impact on more than a single primitive threat type the container is transparent for. The second case occurs if in every well-structured containment tree all possible threat propagation paths cross containers which exhibit one of the above properties prior to encountering any containers which do not. This depends not only on the container classification but also on the specification of the containable relationship.

In any of the above two cases, performing a mitigation operation associated with a compound threat type cannot have an information utility impact greater than already inflicted by the de-facto container transparency — in terms of the information flow channels affected along the exposure channel itself. As a consequence, we can reason about what would otherwise be compound threat type mitigation operations as being specific to the respective individual primitive threat types. LoEs associated with such mitigation operations are called *common* LoEs. The term *common* comes from the fact that the offset ranges of such mitigation operations are equal for all respective primitive threat types they are applicable to.

The NULL LoE

The LoE definition as given does not allow for a LoE to be defined over the insignificant exposure degree range. This is inherent to the definition of the offset range, due to the lack of necessity for the mitigation of the exposure degrees in the range.

However, for consistency reasons, we would like to be able to refer to the exposure degree range as covered by a LoE. Thus, we introduce a *dummy* mitigation operation that we name the *null* operation. With respect to exposure degree it behaves as an identity function. The LoE induced by having $M = \{null\}$ is called the *NULL* LoE. Defined in this manner, the NULL LoE covers the insignificant exposure degree range.

4.4.4 An Example and a Note on Transparency Functions

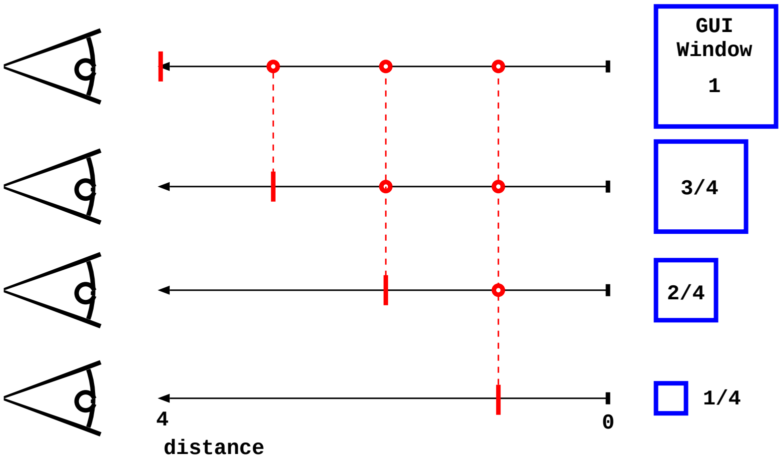
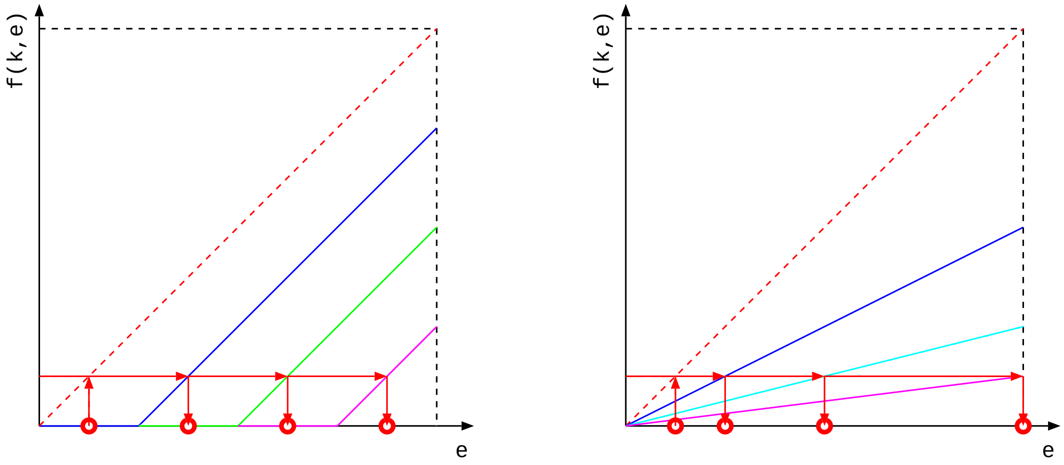


Figure 4.3: Example intuitive representation of LoEs.

Figure 4.3 represents an intuitive illustration of the notion of a Level of Exposure. The example implies a threat of type Optical. The thick red vertical bars denote viewing distances giving rise to significant exposure degree as experienced by data items contained within the respective GUI windows. The smaller the window the bigger the proximity at which the information is considered exposed. The other way around, the figure hints at how the stepwise GUI window shrinking can be leveraged to mitigate the threat. Additive transparency function is assumed. The red circles delimit LoEs defined by the respective GUI window size reduction operation.



(a) Additive transparency function.

(b) Multiplicative transparency function.

Figure 4.4: Example LoEs.

Figures 4.4(a) and 4.4(b) illustrate LoE definition from the point of view of the two types of transparency functions. Red circles on the x axes of the plots represent LoEs. The dashed diagonal represents full transparency, i.e. the identity function. Each of the transparency functions plotted in the graphs corresponds to a single successive unit decrement in GUI window size. In case of the additive transparency functions, Figure 4.4(a), each decrement in size corresponds to a reduction in transparency by a quarter in the absolute sense. For multiplicative transparency functions, Figure 4.4(b), every unit decrement in the window size results in a relative transparency reduction by a half.

LoEs are determined by projecting onto the x axis, with respect to each of the transparency functions, exposure degree value delimiting the significant and insignificant exposure degree regions from the y axis ($S(e)$). Note how, in the Figure 4.4(b), the relative offset ranges corresponding to the single increase in the transparency affecting parameter vary relative to the exposure degree value the operation is applied to.

4.4.5 LoEs: Information Sensitivity Classification Consequences

In the introduction to the LoE definition we have stated that LoEs are to be universally valid. This implies that LoEs should be independent, among other things, of the state of the data items exposed to a threat. However, as shown in the Figure 4.5, and previously in the Figure 4.2, the exposure degree range over which the NULL LoE is defined does vary with the information sensitivity class of the data items. And, consequently, so do the offset ranges of the threat mitigation operations. This essentially makes LoEs information sensitivity class dependant, violating the LoEs *universal validity* property.

For the representational and management complexity reasons we aim to preserve the

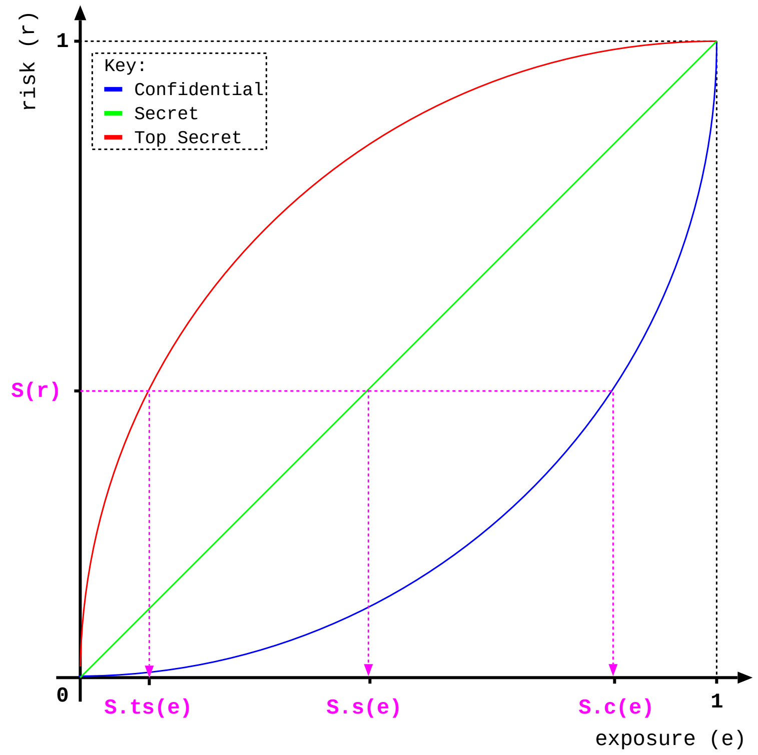


Figure 4.5: The NULL LoE threat type dependence.

universal validity property. We accomplish that through a transformation that we call LoE *semantic mapping*. The term *semantic* is used as LoEs are interpreted in terms of the associated threat mitigation operations in the context of the threat mitigation process. The semantic mapping process consists of re-mapping exposure degrees, associated with different information sensitivity classes, to a common base relative to which LoEs can be defined in a fashion which preserves their universal validity. In other words, so that the offset range of each mitigation operation is the same irrespective of the sensitivity of the threatened data item. The common base assumes the same insignificant exposure degree range. For the common base we choose the exposure degree range corresponding to the highest information sensitivity class. The rationale behind the choice shall be clarified shortly.

In the general case, the semantic mapping has to be done on per individual exposure degree value basis and cannot be generalised across the whole exposure degree range. It consists of solving the equations of the form:

$$f(v_1, e_1) = f(v_i, e_i)[= r]$$

for the value of e_1, where e_1 and e_i are exposure degrees corresponding to risk r relative to information values v_1 and v_i respectively. The mapping can then be given explicitly as a set of coefficients or implicitly in a form of a mapping function. The ideal scenario, on the other hand, is where the semantic mapping coefficient, of the form $k(v_1, v_i)$, is constant across the whole exposure degree range, for every information sensitivity class. This, for example, is not the case for the transparency functions in the Figure 4.5 but is the case for the transparency functions in the Figure 4.4(a).

The mapping coefficients itself can be seen, in both the above cases, as additive or multiplicative, i.e. $e_1 = e_i - k(v_i, v_1)$ or $e_1 = e_i \times k(v_i, v_1)$ respectively. Unsurprisingly, this corresponds to the form of additive and multiplicative transparency functions. Consequently, we can effectively represent the LoE semantic mapping as the transparency of the Data Item containers as determined by the sensitivity of the contained information. For this purpose, the Data Item container class encapsulates the *sensitivity* attribute as presented in the Figure 3.4.

To validate the approach we need to check whether the information reduction property (IRP) (Section 3.6) still holds. This is where the particular choice of the common base for semantic mapping comes into play. For the IRP to be violated a case needs to exist where exposure degree is increased in a threat propagation process due solely to container transparency. However, given the particular semantic mapping common base, this can never arise as for any value of risk it holds that:

$$g(v_1, r) \leq g(v_i, r)$$

where g is essentially f^{-1} and maps risk r to the corresponding exposure degree relative to the asset value v. The validity of the argument is also implicit in the Figure 4.5. Unless the common base corresponded to the highest information sensitivity class, the above statement, and thus the IRP, would not hold in the general case. In other words, exposure degree would always increase when crossing the boundary of a data item containing information of sensitivity higher than the that of the common LoE semantic mapping base.

Consequences of the exposure degree semantic mapping as presented are two-fold. Firstly, it provides for the preservation of the LoEs *universal validity* property — impacting on the complexity of LoE space representation and management. Secondly, it effectively collapses the information manipulation category of threat mitigation operations (Section 3.5) into that of container manipulation operations. In other words, operations that affect information sensitivity through the reduction of the quality and/or the quantity of the exposed pieces of information can, from now on, be seen as manipulating the information sensitivity attribute of the Data Item containers. Thus, the information manipulation operations can be reasoned about as container manipulation operations.

4.4.6 LoEs Model

Definition. *A LoEs Model is a triple* $< L, \gg, \oplus >$ *where* L *is a set of Levels of Exposure,* $\gg \subseteq L \times L$ *is a dominance relation and* $\oplus$ *is the join operator.*

The *dominance* relation $\gg$ states that for any two LoEs A and B, LoE A dominates LoE B (written $A \gg B$) if and only if:

1. A and B correspond to the same threat type or B corresponds to a threat sub-type of the threat type A corresponds to.
2. $q_A \geq q_B$ where $[p_A, q_A]$ and $[p_B, q_B]$ are LoE defining exposure degree ranges for LoEs A and B respectively.

As stated, LoEs corresponding to the threats of different types are comparable only if they are of the same type or exhibit the super-, sub-type relationship in that particular order and not the other way around. The join operator is as was introduced in Section 3.5.4.

LoE vector representation

To represent the state of the world we use LoE vectors of the form:

$$[l_1, \ldots, l_n]$$

where n enumerates all primitive threat types. The dominance relationship is not defined on LoE vectors, only on individual LoEs. A corollary is that never do we reason about any aspect of threats using compound threat types. An exception to this, and the reason the compound threat types were introduced, is during the mitigating operation discrimination process, presented in the next section.

The join operator

The *join* ($\oplus$) operator applied to LoEs of the same primitive threat type has the following semantics:

$$A \oplus B = LoE(q_A \oplus q_B)$$

where A and B are LoEs defined over information exposure degree ranges $[p_A, q_A]$ and $[p_B, q_B]$ respectively. The function $LoE(e)$ maps exposure degree e to its corresponding LoE.

If A and B are not of the same threat type then the $\oplus$ operator is seen as applied to LoE vectors. This is defined by:

$$[l_1, \ldots, l_n] \oplus [m_1, \ldots, m_n] = [l_1 \oplus m_1, \ldots, l_n \oplus m_n]$$

i.e. the operator it "joins" the vectors threat type-wise.

LoEs Model Structure and its Properties

Statement. *The LoEs Model $< L, \gg, \oplus >$ forms a finite lattice.*

A lattice is a structure characterised by the existence of the *least upper bound* and the *greatest lower bound* for any sub-set of its elements. In mathematical terminology a lattice is usually described as a partially ordered set in which any two elements have a *supremum* and *infimum*. In our case, the partial ordering is induced by the dominance relation.

By placing one rational assumption we can intuitively prove that any LoEs model forms a lattice. The assumption is the existence of the information destruction operation. An information destruction operation mitigates threats of all types and its offset range is equivalent to the full exposure degree range. As such it represents the lattice *maximum* — the supremum for any two LoEs in the lattice. Similarly, the *NULL* LoE represents the lattice *minimum.* As a consequence, not only does the LoEs model represent a lattice but, even stronger, it represents a *bounded* lattice.

In [Den76], Denning stated four axioms that need to be satisfied for an information flow policy to form a lattice. Applied to the LoE model $< L, \gg, \oplus >$ they could be formulated as:

1. The set of LoEs L is finite.
2. The dominance relation $\gg$ is a partial order on L.
3. L has a lower bound with respect to $\gg$.
4. The join $\oplus$ operator is totally defined, least upper bound operator.

Although LoEs are by definition associated with a set of threat mitigation operations, their availability may vary based on ubiquitous computing platform CASPEr is hosted on. Given the LoEs model lattice structure, we can represent mitigating operations as directed edges overlaid on the lattice. Each of the edges starts at the vertex representing the LoE the mitigation operation is associated with and ends at the vertex representing the LoE that subsumes the lower bound of the operation's relative offset range. The edges effectively point from dominating LoEs to dominated LoEs denoting the threat mitigation effect of the corresponding operations. We discuss the axioms as they highlight additional properties of the LoEs modelling approach.

The first axiom. The finiteness of the LoE space is the consequence of the finiteness of the number of threat types and the number of LoEs per threat type. Note that the LoE space finiteness does not depend on the finiteness of the number of contextual states causing a LoE activation. An important implication is the ability to change the contextual state - LoE mappings in a dynamic, run-time, fashion without necessitating alterations to the LoE space specification. This further emphasises the importance of explicit threat reasoning for the device autonomy.

The second axiom. Partial order is a relation which is reflexive, transitive and anti-symmetric. The dominance relation is reflexive owing to the use of $\geq$ in the second point of its definition above. Transitivity says that if $A \gg B$ and $B \gg C$ then $A \gg C$. In other words, if an exposure degree value in the range of LoE B can be mitigated by an operation associated with A, and an exposure degree in the range of C can be mitigated by an operation associated with B, then the exposure degree in the range of C may also be mitigated by the operation associated with A. The anti-symmetry says that if $A \gg B$ and $B \gg A$ then $A = B$. The anti-symmetry property of the dominance relation eliminates redundant LoEs by implying that all mitigating operations of the same offset range define a single LoE.

The third axiom. Existence of the lattice minimum, the NULL LoE, in L means that $\forall LoE \in L, LoE \gg$ NULL LoE. The axiom acknowledges the existence of a LoE defined over the insignificant exposure degree range which is "reachable" from all other LoEs. In other words, the axiom states that all LoEs are mitigatable.

The fourth axiom. This axiom has two parts. Firstly, the join operator is required to be totally defined. Formally, for any two LoEs A and B, $A \oplus B$ is defined within L. This holds for two separate reasons: i) composition of the operations associated with the LoEs A and B shall define the new LoE; and, ultimately, ii) the existence of the LoEs model lattice maximum. Secondly, the join operator is required to be the least upper bound. This means that for $A, B, C \in L$ the following properties hold:

1. $A \oplus B \gg A$ and $A \oplus B \gg B$.

2. $C \gg A$ and $C \gg B$ then $C \gg A \oplus B$.

This holds by definition. Both the properties are somewhat analogous to the transitivity requirement of the second axiom above — but in relation to mitigating operations affecting compound threat types. The first property states that mitigating a LoE of a least common compound threat type affects the LoEs of the corresponding primitive threat sub-types simultaneously. The second property extends this to LoEs of any compound threat type common to the respective primitive threat types.

4.5 Threat Mitigation: The LoE Way

Given a set of threats and a containment configuration there are likely to be multiple alternative threat mitigation strategies. For example, in case of a threat of *Optical* information exposure, the actions of resizing the GUI window, hiding it, blurring or blanking the display or migrating the GUI window to a more restricted display are viable. In this section, we explore the issue of choosing the optimal set of operations, i.e. threat mitigation strategy, to mitigate a set of threats as experienced in a realm, according to a set of criteria — most notably, information utility. The major contribution of this section, and this chapter as a whole, is the dynamic programming algorithm that we develop to accomplish this.

4.5.1 Action Impact as a Discrimination Criteria

In Section 4.2 we have introduced the notion of information utility impact as a side-effect of threat mitigation process and sitting at one end of the seesaw that we aim to balance. The other side being occupied by information exposure protection. Although the seesaw is completely defined by identifying the two ends we point-out several other factors that may contribute to its finer balance — and unify them in the notion of *action impact*.

Relevant Factors

The following are the further three factors that we see as relevant for balancing the seesaw:

- Operation reversibility.
- Operation cost.
- System usability impact.

As a consequence of the system and user behaviour rationality assumption (Section 4.2) and the nature of information exposure channels, information utility of a piece of data prior to the execution of a set of threat mitigation operation is always greater than its information utility upon the execution. Thus, once a threat that triggered the adaptation process disappears it is likely that, in order to maximise information utility (i.e. restore it), the threat mitigation actions have to be undone. *Reversibility* of a threat mitigation operation captures the degree to which the action is reversible and any cost, monetary and/or resource, of the reversion process.

Threat mitigation operation's *cost* captures general resource and/or monetary requirements for its execution. It is expected to be of particular relevance on highly constrained ubiquitous computing devices.

Finally, threat mitigation operations may have an impact on the overall system usability. System usability can be defined as the intrinsic ability of a system to meet the functionality and performance expectations derived from its hardware and software platform specification. While the information utility impact is focused solely on effects mitigation operations have on individual pieces of information, the *system usability impact* takes a wider angle on the overall system functionality. High system usability impact is tightly connected with the notion of indiscriminateness of security mechanisms. Examples are: disabling communications interfaces, mandatory and holistic file system encryption and bans on use of removable storage or mobile computing devices, external displays, "unknown" printers etc.

Action Impact

We introduce the notion of *action impact* as a function which combines information utility, together with operation reversibility, its cost and overall system usability impact into a single metric for discriminating between sets of alternative mitigation operations.

The single metric expression is only nominal, introduced for the clarity of the presentation. It is not intended to enforce any particular way in which the contributing factors should be reasoned about in practise. For example, in Section 5.5, we show how probabilistic prioritising can be used, in conjunction with the proposed policy model, to implement action impact based threat mitigation operation discrimination.

Let ι be the action impact function and let A be the set of all protective actions available on a computing platform. Further, let $A^\circ \subset A$ be a set of alternative threat mitigation operations available for a container with respect to the experienced exposure. We can now define:

$$A^{\circ\circ} = \{a \mid a \in A^\circ \wedge \iota(a, c) = min_{x \in A^\circ}[\iota(x, c)]\}$$

where c is containment path expression matching a container to which the threat mitigation operations apply. We assume, without any loss of generality, that the lower the action impact (ι) the more favourable an action is. Thus the minimisation ($min_{x \in A^\circ}$). Then, the set $A^{\circ\circ}$ represents the set of optimal alternative mitigation operations. Consequently, if $\mid A^{\circ\circ} \mid > 1$ then $\forall a, a \in A^{\circ\circ}$ is an optimal choice. The definition is straightforward to generalise to the case where a represents composition of threat mitigation operations.

In case where A° is an empty set, i.e. there are no available mitigation operations matching experienced exposure, $A^{\circ\circ}$ is assigned a single *dummy* element with associated action impact of ∞. The particular value is chosen relative to the desired action impact minimisation and plays a role in the realm-wide optimal threat mitigation strategy discovery algorithm presented in the following sections. The essence to note is that by having an action impact of ∞, the dummy operation is always going to be more expensive than any alternative — putting information security protection in place of the absolute priority over any other threat mitigation operation discrimination criteria.

In the rest of the thesis, whenever we refer to the term *optimal* or any of its derivations we assume optimality with respect to the action impact metric.

4.5.2 The Protective Cover

The concept of the *protective cover* of a containment tree is fundamental for reasoning about optimal threat mitigation strategy for a threat-exposed realm. To define the *protective cover* we firstly introduce a more general concept of the containment tree *cover*. We differentiate between the *simple* and *compound* containment tree covers.

Definition. *The simple cover of a containment tree is defined with respect to a node n of the tree. A simple cover of the tree is a subset, C_s, of the tree nodes such that any path starting at the node n and leading to all the leaf nodes in the subtree rooted by n has to cross at least one node from the cover C_s.*

The compound cover, C_c, of a containment tree is defined with respect to a set of nodes N of the tree. A compound cover of the tree is the union of a single simple cover for every *node $m \in M$, where $M \subset N$, such that $\forall n \in N \backslash M, \forall m \in M$ the node n is not in the subtree rooted by the node m.*

The choice of the nodes in M ensures that there are no nodes in N at which a path may start and reach the leaf nodes without crossing any of the nodes in the cover C_c.

We define *protective cover* of a containment tree by placing the concept of the containment tree cover into the context of information exposure threats. We similarly distinguish between the *simple* and the *compound* containment tree protective covers.

Definition. *The simple protective cover of a containment tree is defined with respect to a threat t originating at the node n of the tree. A simple protective cover then is a simple cover, C_s, of the tree, together with a set of threat mitigation operations M, such that after executing all operations in M on the respective nodes in C_s the data items below the node n experience no more than NULL LoE due to the threat t.*

The compound protective cover is defined for a set of threats T, of one or more primitive types, and a set of the respective threat source nodes N. Compound threat cover extends the concept of a simple protective cover in a manner analogous to that by which the compound cover extends the concept of a simple cover of a containment tree.

In the rest of the thesis, when we refer to the cover of a containment tree we really mean the protective cover of the tree unless we explicitly specify differently. Note the use of the term *at least* in the definition of a simple cover of a containment tree. In the context of a simple protective cover, the term effectively implies that a single threat may be mitigated jointly at several levels in a containment tree. This is exploited in the design of the algorithms for discovering the optimal cover of a containment tree as presented from Section 4.5.5 onward.

Which LoEs to Mitigate?

Given a set of threats and a containment tree, the threat propagation process gives us a LoE vector describing LoEs active at each of the containers in the tree — data items and otherwise. From the intensity reduction property it follows that LoEs as experienced at data item level in the tree are always dominated by the corresponding LoEs active at the inner nodes of the containment tree. Therefore, by mitigating a threat through offsetting a LoE as experienced at an inner node of a tree an unnecessary information utility impact may be incurred. Consequently, irrelevant of the positioning in a containment tree of a node at which threat mitigation operations are applied they need not offset more than the LoEs as experienced by data items in the subtree below the node. Slightly more formally, for a container `v`, the vector of LoEs to mitigate is derived in a bottom-up fashion as:

$$[l_1, \ldots, l_n]_{\mathrm{v}} = \begin{cases} \bigoplus_{i \in children(\mathrm{v})} [l_1, \ldots, l_n]_i & \text{if } !is\_leaf(\mathrm{v}) \ ; \\ [l_1, \ldots, l_n]_v & \text{otherwise } ; \end{cases}$$

The second case represents the base case, where v is a data item, at which point the set of LoEs to mitigate is exactly the set of LoEs experienced.

Cover Complexity

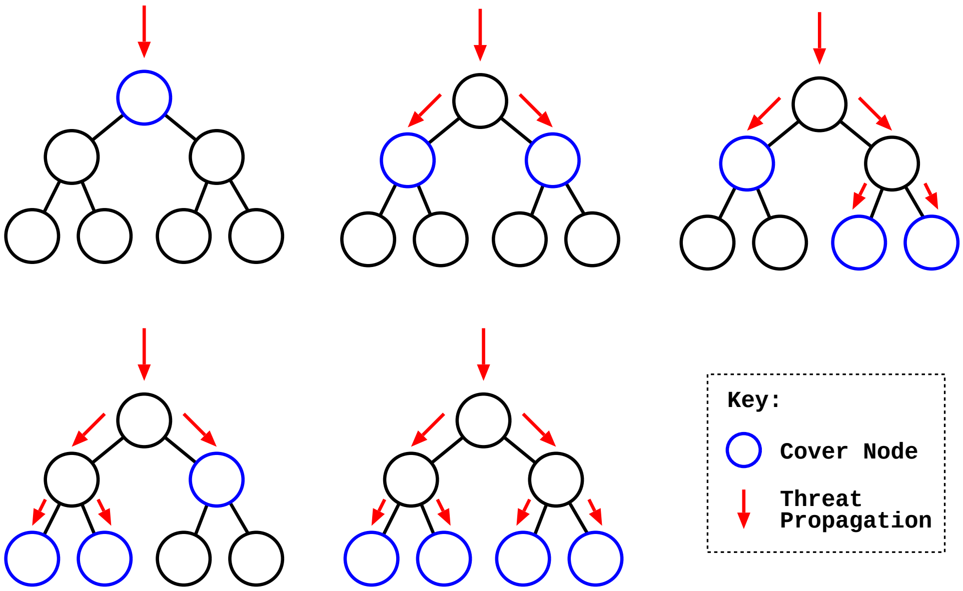


Figure 4.6: Realm covers example.

Figure 4.6 shows all possible covers for a rather simple containment tree such that every threat propagation path crosses exactly a single node in a cover. We assume that the threat originates outside the root node of the tree and that mitigation operations are available for each of the containers. The purpose of the illustration is to hint at at the variety of possibilities for forming a protective cover, even in a very small realm, subject to availability of the adequate mitigation operations.

To gain a slightly more formal feel on the size of the protective cover space, in general, we make two assumptions: i) containment tree structure regularity; and ii) availability of mitigation operations at every node in a tree. The former is related to node branching factors and the tree balance. The latter implies that every node in the tree can be a part in the cover. We, however, do not envisage the assumptions to frequently hold in practise. In other words, the following simple analysis expresses the worst case scenario complexity.

Let N be a node branching factor, uniform across a containment tree, and k be the tree depth, where $k = 0$ denotes the root level. We can then express the number of possible covers for a containment tree as:

$$\left(\left((1^N + 1)^N + 1\right)^N \ldots + 1\right)$$

where the factor 1^N represents the base case, i.e. the number of possible cover combinations in a tree with only a single node. The exponent N, over each of the bracketed terms, accounts for the number of possible covers for N disjoint containment trees, i.e. with no common root. Addition of the common root node for N trees contributes by exactly one cover — the one formed solely by the root node itself. This is accounted for by the term $+1$ in each of the brackets above. The number of such terms is equal to k.

Using the big O notation, the size of such a cover space can be expressed in terms of N and k as:

$$O\left(2^{N\times k}\right)$$

The exponential complexity renders the brute force approaches to finding the optimal cover of a containment tree unfeasible but for small realms.

However, we can observe that the optimal protective cover for a sub-tree of a containment tree is independent of the optimal protective covers for any other, non-intersecting, sub-tree. In other words, the problem of finding the optimal cover exhibits a clear optimal sub-structure. Furthermore, the individual sub-problems recur in what would be a bottoms-up, divide and conquer, approach to finding the optimal cover. This allows us to develop a dynamic programming solution to the problem as presented next.

4.5.3 Information Utility and Containment Configuration

The definition of information utility impact, as used in the action impact metric above, presented in Section 4.2, is implicitly relative to a single data item. In the general case, as we have already seen through examples, mitigation operations may be available at any of the nodes of a containment tree. By the definition of a tree, each non-leaf node has one or more leaf nodes below itself. Consequently, invocation of a mitigation operation at a non data item container may affect information utility of more than a single data item.

The corollary of the above argument is that information utility impact of a mitigation operation is, in the general case, dependent on the particular containment tree configuration at the time of the operation execution. To account for this, information utility impact of a threat mitigation operation is defined as:

$$\upsilon(a,c) = \sum_{x \in data\_items(c)} \omega_x \upsilon(a,x)$$

where the arguments to the information utility impact function (υ) are the mitigation operation id (a) and the path to the container at which the action is to be performed (c). The sum goes through all data items contained in the containment sub-tree rooted by the container referred to by c. The ω_x factor leaves space for weighting information utility impact based on properties of individual data items, e.g. contained information sensitivity — prioritising information utility of the more sensitive pieces of information.

To synchronise the information utility impact values with the containment model updates we use the triggers associated with the model update operations as presented previously in Section 3.4.6. In the following sections, we assume that the information utility impacts associated with the available mitigation operations reflect the corresponding realm configuration.

4.5.4 The 0-1 Knapsack Problem

The algorithm we propose for finding the optimal cover for a containment tree in face of a set of threats builds on the solution to the *0-1 Knapsack* problem which is well known in the literature[1]. For clarity purposes, we firstly briefly introduce the 0-1 Knapsack problem and its solution, which we subsequently build upon.

The Problem

The 0-1 Knapsack problem can be stated as follows. A thief robbing a store has a knapsack that can carry at most W kilogrammes. In the store, he finds n items, each with weight w_i and of value v_i. The thief would, unsurprisingly, like to walk away with the most valuable load. However, the weight he can take away is strictly limited by the capacity W of the knapsack. Also, the items cannot be broken into smaller units — each has to be either included in the load or left aside. Thus the name *0-1 Knapsack* problem. The problem then is: which items should the thief take?

The general *Knapsack* problem is the classical example of a NP hard problem. However, when confined to "small" integers, in the number of items as well as the weights, and when fractioning of items is disallowed it can be solved using a dynamic programming approach [CSRL01], incurring pseudo-polynomial complexity. Dynamic programming, in general, exploits the optimal sub-structure of a problem together with the existence of overlapping sub-problems to provide for tractable solutions to what would otherwise represent intractable problems. The optimal substructure of the 0-1 Knapsack shows itself if we observe that by excluding an item i, of weight w_i, from a knapsack containing the optimal selection of items for its weight w, what we obtain is the optimal load for the weight $w - w_i$.

Sub-problem overlapping implies that a standard recursive solution to the problem would repeatedly encounter the same set of sub-problems rather than solving new sub-problems in every iteration. Dynamic programming exploits this by memorising solutions to sub-problems for reuse. Furthermore, no backtracking is allowed. The memorisation step is where the requirement for the problem confinement to small integers essentially comes into play. Otherwise, the incurred overheads become prohibitive. The existence of the overlapping sub-problems for the problem we attempt to solve is shown in the algorithm presentation, below.

[1] The 0-1 Knapsack problem is a classical example of dynamic programming as found in textbooks on algorithms and data structures, e.g. in [CSRL01].

The Algorithm

Dynamic programming solution to the integer constrained 0-1 Knapsack problem is as follows. We enumerate all the items available as $1 \dots n$, where n is the total number of items. No particular ordering of items is required. Then, we let C be a matrix of size $n \times W$ where $C[i, w]$ represents the optimal solution for knapsack capacity w, $0 \leq w \leq W$, considering only the items $1 \dots i$, $1 \leq i \leq n$. The algorithm can then be given as:

$$C[i,w] = \begin{cases} 0 & \text{if } i = 0 \text{ or } w = 0 \text{ ;} \\ C[i-1,w] & \text{if } w_i > w \text{ ;} \\ \max(v_i + C[i-1, w-w_i], C[i-1,w]) & \text{if } i > 0 \text{ and } w \geq w_i \text{ ;} \end{cases}$$

The first case simply says that if there are no items available for inclusion in the knapsack, or the capacity of the knapsack is 0, then its contents can never have a value of more than 0. The second case states that if the weight of an item being considered for inclusion into the knapsack is bigger than the capacity of the knapsack itself we should drop it and look for the solution considering the items $1 \dots i-1$ only. Finally, in the third case, the value of the knapsack load is maximised either by including the item i and filling the remaining load $(w - w_i)$ by choosing from not previously considered items $(1 \dots i-1)$, or by dropping the item i and filling the weight w with the unexpended items only. We can now see, from the third case, the role played by the memorisation of solutions to the intermediate problems.

The Complexity

The run-time complexity of the above algorithm is pseudo-polynomial in the order of the number of items n and the knapsack weight W, i.e. $\mathrm{O}(nW)$. Thus the “small” integer constraint on the algorithm. The term pseudo-polynomial denotes that the complexity is not expressed in terms of the input length in bits — as usual in complexity theory, but in terms of the size of the input values. The given complexity reflects that the algorithm consists of performing only simple arithmetic operations for each of the $n \times W$ cells of the matrix.

4.5.5 A Single Container Optimal Cover Determination

For clarity purposes, we present the algorithm for finding the optimal cover for a containment tree in two stages. The first stage, presented in this section, consists of finding the optimal set of mitigating operations for a containment tree consisting of a single container. In the second stage, presented in the next section, we generalise the algorithm to provide for the optimal cover for multi-container containment trees.

Setting the Scene

To aid the simplicity of the algorithm presentation we, initially, make one assumption and impose one more general constraint. The assumption and the constraint hold equally in this as well as in the next section. They are subsequently relaxed, in Section 4.5.7.

Assumption. *Owing to a particular container classification and containable relationship definition, every data item can only ever be exposed to a single (primitive) threat type at any one time.*

The assumption allows us to leverage the notion of common LoEs. In other words, it allows us to consider mitigation operations affecting a spectrum of primitive threat types as specific to any of the types. This allows us to ignore the complexities involved in reasoning about the information utility impact of mitigation operations associated with compound threat types, in the initial version of the algorithm — aiding the clear presentation of its core points.

Constraint. *Transparency functions used in the system are of the additive type.*

This constraint is fundamental for the validity of the form in which we state the optimal cover discovery problem in the next sections. The particular form is used as it aids the clarity of the presented solution through its resemblance to the 0-1 Knapsack problem. The more general benefits of using additive transparency functions have been discussed to previously.

The Problem Statement

In this section we address the problem of finding a set of threat mitigation operations available at a single container such that, when executed, they offset at least the LoE active for the container and, at the same time, incur the minimum action impact. In the rest of the chapter we use the term *cost* to mean *action impact* — for its resemblance to the 0-1 Knapsack problem statement.

The problem we are solving can be seen as the “backward” 0-1 Knapsack problem. In the standard 0-1 Knapsack, a thief has broken into a shop and wants to maximise the value of its load, possibly leaving some space in the knapsack empty. In our case, a person has walked into a shop and they want to spend as little money as possible to fill their shopping bag, and even over-fill it slightly, having to carry away goods in their pockets, if that is the cheapest option.

The assumption and the constraint stated above shape the problem in the following manner. As no container may experience exposure degree of more than a single type, the dominance relationship imposes a total ordering on the LoEs comprising the LoEs model as seen from the perspective of every individual container. The total LoE ordering, on per threat type basis, can be seen as a path with the starting point represented by the NULL LoE and its end being the LoE associated with the information destruction operation. We then define the unit of distance along the path to be the exposure degree quantisation step. Thus, every LoE to mitigate can be seen as at a specific, integer, distance on the path. Further, as a consequence of the above constraint we can reason about individual threat mitigation operations as of sticks of fixed lengths. The length of a stick represents reduction in exposure degree caused by the corresponding threat mitigation operation. The cost of a stick corresponds to the mitigation operation's action impact.

The problem of finding the set of threat mitigation operations that offset a given LoE with minimum action impact can, now, be stated as:*given a set of sticks, choose a subset such that when all its sticks are laid end-to-end over the path, starting at its beginning, they reach at least the point on the path representing the experienced LoE, at the minimum cost.*

The Algorithm

In analogy to the 0-1 Knapsack problem, the general case of the above problem is also NP complex. However, constrained to "small" integers, in terms of the number of sticks, i.e. mitigating operations, and the path length, it can be solved using a dynamic programming approach in pseudo-polynomial time. The optimal sub-structure of the problem is given by the following. Let S be a set of sticks that covers a path distance l at the minimum cost and let i, of length l_i, be in S. Then, by excluding stick i from S, we obtain the minimum cost solution for the path distance $l - l_i$.

The algorithm is formally specified as:

$$f(i, l_c, l_{tot}) = \begin{cases} 0 & \text{if } l_{tot} \leq l_c \ ; \\ \infty & \text{if } i = 0 \ ; \\ \min[f(i-1, l_c, l_{tot}), c_i + f(i-1, l_c + l_i, l_{tot})] & \text{otherwise} \ ; \end{cases}$$

The use of function notation is analogous to the use of a simpler table index notation in the previously presented solution to the 0-1 Knapsack problem. The argument i denotes that sticks $1 \dots i$ are to be considered for building the solution. Parameter l_c represents the length of the path *already covered*, while the l_{tot} specifies the total target path distance that needs to be covered. Therefore, the difference $l_{tot} - l_c$ denotes the distance that remains to be covered at each stage, i.e. iteration, of the algorithm execution.

The first case says that if we have already covered the target length, or even gone past it, then there is nothing else to do. The cost associated with doing nothing is 0. The second case states that if we have no sticks to consider, then it is impossible to cover any path distance. This is equivalent to saying that the cost of covering the distance is infinite (∞). In the third case, we aim to choose the the cheapest (min[...]) alternative among: discarding the stick i, currently under consideration, and covering the remaining distance, $(l_{tot} - l_c)$, using the sticks $1 \dots i-1$ only; or, including the stick i in the solution, at the cost of c_i, and covering thus remaining path distance, $l_{tot} - (l_c + l_i)$, by a subset of sticks $1 \dots i-1$. As the algorithm progresses, never is a decision to include a stick in the solution re-evaluated.

i (l_c, l_{tot})

	0	1	2	3
(0,0)	0	0	0	0
(0,1)	∞	2_1	$\min[2_1, 2_2 + 0] = 2_{1/2}$	$\min[2_{1/2}, 3_3 + 0] = 2_{1/2}$
(0,2)	∞	2_1	$\min[2_1, 2_2 + 0] = 2_{1/2}$	$\min[2_{1/2}, 3_3 + 0] = 2_{1/2}$
(0,3)	∞	∞	$2_2 + 2_1 = 4_{1,2}$	$\min[4_{1,2}, 3_3 + 2_2] = 4_{1,2}$
(1,0)	0	0	0	0
(1,1)	0	0	0	0
(1,2)	∞	2_1	$\min[2_1, 2_2 + 0] = 2_{1/2}$	$\min[2_{1/2}, 3_3 + 0] = 2_{1/2}$
(1,3)	∞	2_1	$\min[2_1, 2_2 + 0] = 2_{1/2}$	$\min[2_{1/2}, 3_3 + 0] = 2_{1/2}$
(2,0)	0	0	0	0
(2,1)	0	0	0	0
(2,2)	0	0	0	0
(2,3)	∞	2_1	$\min[2_1, 2_2 + 0] = 2_{1/2}$	$\min[2_{1/2}, 3_3 + 0] = 2_{1/2}$
(3,0)	0	0	0	0
(3,1)	0	0	0	0
(3,2)	0	0	0	0
(3,3)	0	0	0	0

Sticks	1	2	3
Length	2	2	1
Cost	2	2	3

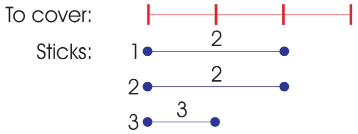


Figure 4.7: Container-local algorithm example.

Figure 4.7 gives an example of the algorithm execution. In the top right corner the figure is the summary of the sticks available, presented both graphically and in a tabular format. The available sticks are enumerated as 1, 2 and 3. The rest of the figure is occupied by a table containing all the values of the algorithm's lookup table upon its execution. The table encodes two dimensions given by l_c and l_{tot} as a single dimension using pairs (l_c, l_{tot}). Each of the cells in the table contains the minimum cost for a corresponding sub-problem. The optimal solution is encircled in the red ellipse. The subscripts of the cell values identify threat mitigation operations comprising the optimal solution. They demonstrate how the solution can be recovered upon algorithm termination. Note that the optimal solution "overshoots" the target path distance although an exact match is available — however, with a lower cost.

4.5.6 The Realm-Wide Optimal Cover Determination

In the previous section, we have presented a dynamic programming algorithm to find the optimal set of mitigation operations to offset a LoE for a single container. In this section, we extend the algorithm to find the optimal cover for a containment tree in the general case. Note that the previously stated assumption and constraint still hold.

The Problem Statement

In the stick-path terminology, we can state the problem of finding the optimal protective cover for a containment tree as follows.

With each leaf node of a containment tree, there is a certain path distance to cover associated. The distances correspond to the LoEs active for the respective data items. The path distance to cover for each non-leaf node is equivalent to the maximum path to cover at each of its children nodes. This was defined, in terms of LoEs, in Section 4.5.2. With each of the nodes there may be none or more sticks associated — denoting the available threat mitigation operations.

When a stick associated with a non-leaf node is used to contribute to the cover sought for the node, it applies simultaneously to the path distance covers for all the containers below the node. This reflects the fact that the application of a threat mitigation operation on a container reduces the corresponding exposure degree experienced at all of its children containers. This is owing to the manner in which the threat propagation is defined. The cost of applying a stick associated with a node is constant whether the node is a tree leaf or not. This would not hold unless information utility impact associated with a threat mitigation operation reflected the containment tree configuration relative to the container the operation is associated with — as specified in Section 4.5.3.

Let S be a set of sticks available for a containment tree — a union of the sticks available for each of the containers in the tree. The problem, then, is to find the minimum cost subset $S_c \subseteq S$ such that, by using the all of the sticks in S_c, we cover the path distances as associated with all of the leaf nodes of the tree. The cost of a set of sticks is simply the sum of costs of individual sticks in the set. In other words, the goal is to offset all LoEs as experienced at the data item containers of a containment tree, at the minimum cost, by leveraging the threat mitigation operations available at all of the containers in the tree. Note that the problem statement implies that a threat may be mitigated gradually as it propagates down a containment tree — thus the subtlety in the protective cover definition of Section 4.5.2.

The Algorithm

The algorithm to find the optimal cover builds on the dynamic programming approach to solving the single container problem, presented in the previous section. It is itself a dynamic programming algorithm. The optimal sub-structure argument for the problem is analogous to the argument presented in the previous section.

The formal specification of the algorithm is given by:

$$f(v,i,l_c,l_{tot(v)}) = \begin{cases} 0 & \text{if } l_{tot(v)} \leq l_c \ ; \\ \sum_{k \in chld(v)} f(k, n_k, l_c, l_{tot(k)}) & \text{if } i = 0 \ \& \ !is\_leaf(v) \ ; \\ \infty & \text{if } i = 0 \ \& \ is\_leaf(v) \ ; \\ \min[f(v, i-1, l_c, l_{tot(v)}), & \\ \qquad c_i + f(v, i-1, l_c + l_i, l_{tot(v)})] & \text{otherwise} \ ; \end{cases}$$

Apart from the parameters described previously, the function takes an additional argument, v, that identifies the container under the current consideration by the algorithm. To account for the slightly extended notation, n_k denotes the number of sticks available at a node k while $l_{tot(k)}$ represents the total target path distance to cover at the node k. The parameter denoting the remaining sticks to consider, i, is relative to the node v.

The algorithm starts its execution either at the root of the containment tree, if the threat is external to the tree, or at all first-hop children of the threat source container simultaneously. If multiple threats of the same type originate at different containers, the algorithm starts in parallel at all the first-hop children of the lowest level threat source container for each of the containment subtrees of the tree. The initial call to the algorithm is of the form:

$$f(v, n_v, 0, l_{tot(v)})$$

The first case of the above algorithm is identical to the respective case in the single-container algorithm. The second case above applies if we have run out of sticks at the current container ($i = 0$), without covering the target path distance ($l_{tot(v)}$), and the container (v) represents a non-leaf node in the containment tree. The rule states that we should then try to cover the remainder of the path distance separately for each of the containment sub-trees below the current node, starting by visiting each of its first-hop children. The overall cost of the next step, thus, is given by the sum of costs of individual covers for each of the sub-trees ($\sum_{k \in chld(v)} \ldots$). As $l_{tot(v)} = \max_{k \in chld(v)} l_{tot(k)}$, at each of the first hop children we only need to search for the cover for the path distance given by $l_{tot(k)} - l_c$ rather than $l_{tot(v)} - l_c$. Note that l_c corresponds to a global variable, while $l_{tot()}$ is container local. Thus, it may happen that as the algorithm progresses down the containment tree, the l_c is already over the local $l_{tot()}$. The first case, then, ensures that the unnecessary path distance “overshooting” does not occur. The third case states that if have run out of sticks ($i = 0$) at the bottom of a containment tree and we still have not covered a target path distance than it must be that we cannot cover it at all — yielding the cost of ∞. The semantics of the last case of the above algorithm is fully analogous to the last case of the single container basic version of the algorithm.

We baptise the above algorithm the constrained *Optimal Cover Determination* (OCD) algorithm. The general, unconstrained, variant of the OCD algorithm is obtained when the assumption and the constraint from the Section 4.5.5 are relaxed, as discussed in the next sections.

An Example

The algorithm is illustrated by the example in Figure 4.8. In the centre of the figure is the containment tree on which the algorithm is executed. The realm consists of tree containers: u, v and w; where v and w are contained within u. Threat mitigation operations are represented as blue-coloured line segments placed next to the nodes they are available at. Each stick has an id, denoted by a number to their left, and a cost, shown on top of them. We can think of containers v and w as representing GUI windows and the container u corresponding to a display. For the display, the example accounts for a two level blurring, available through a couple of single level blur operations. As for the GUI windows, they may be reduced in size, in a single step, to being observable from a safe proximity only. The costs of the respective threat mitigation operations are worked out as: $1 \times [\text{number of affected data items}] + 1$, assuming that each GUI window contains solely one data item (not represented in the figure). The left side of the + operator represents the joint information utility impact while the right side denotes, in a much simplified way, other factors contributing to the action impact. The target path distances to cover are 1, 2 and 2 for containers v, w and u respectively and are represented as red segmented lines next to each of the containers in the figure.

The central point of the Figure 4.8 are the three tables illustrating all the values as computed by the algorithm at every step of its execution. Encircled in red are the final solutions to each of the sub-problems encountered. The sub-scripts represent the ids of the contributing mitigating operations. The optimal cover, as given by the algorithm, consists of a single-level blur at the container u and a resize at the container w. Note that the problem really has four dimensions — one for each of the parameters of the function f. In the figure, three dimensions are encoded in the tables themselves while the fourth is represented by the structure of the containment tree.

To hint at the optimality of the algorithm we contrast it briefly to an alternative, two-step, approach:

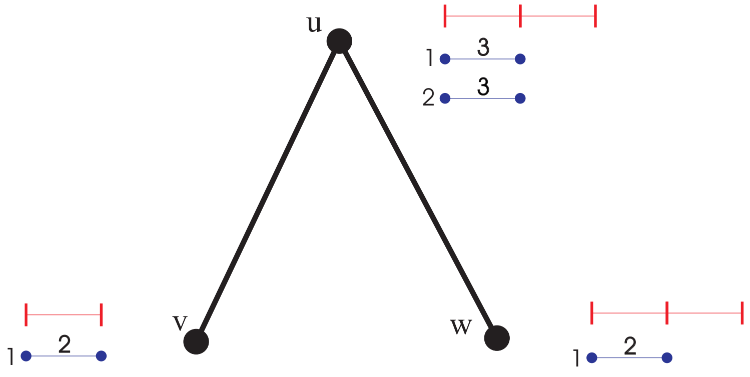
1. First, use the single-container variant of the algorithm from the previous section to find a container-local optimal solution for the LoEs experienced by every container in a containment tree.

2. *Divide and Conquer* bottoms-up and, for every container, compare the cost of its local solution with the sum of the costs of the solutions at all of its first-hop children. Discard whichever is more expensive.

The algorithm terminates at the root of the affected containment tree. For the example in the Figure 4.8, this algorithm would come up with a solution clearly sub-optimal to the OCD. As it would not find any possible local cover for the node w, it would suggest threat mitigation through two level screen blurring at the container u as its final solution. The cost associated with this solution is 6 instead of 5 as obtained from the constrained OCD

i \ (l_c, l_{tot})	0	1	2
(0,0)	0	0	0
(0,1)	$4_{v[1], w[1]}$	$\min[4_{v[1], w[1]}, 3_{u[1]} + 0] = 3_{u[1]}$	$\min[3_{u[1]}, 3_{u[2]} + 0] = 3_{u[1/2]}$
(0,2)	∞	$3_{u[1]} + 2_{w[1]} = 5_{u[1], w[1]}$	$\min[5_{u[1], w[1]}, 3_{u[2]} + 2_{w[1]}] = 5_{u[1/2], w[1]}$
(1,0)	0	0	0
(1,1)	0	0	0
(1,2)	$2_{w[1]}$	$\min[2_{w[1]}, 3_{u[1]}] = 2_{w[1]}$	$\min[2_{w[1]}, 3_{u[2]}] = 2_{w[1]}$
(2,0)	0	0	0
(2,1)	0	0	0
(2,2)	0	0	0

Container u



i \ (l_c, l_{tot})	0	1
(0,0)	0	0
(0,1)	∞	2_1
(1,0)	0	0
(1,1)	0	0

Container v

i \ (l_c, l_{tot})	0	1
(0,0)	0	0
(0,1)	∞	2_1
(0,2)	∞	∞
(1,0)	0	0
(1,1)	0	0
(1,2)	∞	2_1
(2,0)	0	0
(2,1)	0	0
(2,2)	0	0

Container w

Figure 4.8: Common Cheapest Path algorithm example.

algorithm. The excess cost comes from the unnecessary information utility impact caused by the resulting target path distance "overshoot" for the node v.

4.5.7 Discussion

In this section, we discuss the relaxation of the assumption and the constraint made in Section 4.5.5. The general form of the OCD algorithm, accounting for the relaxation of both the constraint and the assumption at the same time, is not presented for the clarity purposes. However, its derivation is straightforward based on the two algorithms presented below.

Generalisation to Multiple Threat Types

The relaxation of the assumption from Section 4.5.5 implies that a data item may, in general, be exposed to threats of multiple primitive threat types simultaneously. Two distinct scenarios may then arise. The first occurs if no compound threat types exist. In this case, the optimal cover can be obtained by independent invocations of the constrained OCD algorithm for each of the primitive threat types experienced within a realm. The other case assumes the existence of threat mitigation operations that impact on multiple threat types simultaneously. Owing to its generality, we focus on the latter scenario.

Following the stick-path analogy, any stick associated with a mitigation operation affecting multiple primitive threat types applies simultaneously to all the individual paths associated with each of the threat types. This is owing to the definition of the compound threat type. The extension of the constrained OCD algorithm to account for the consequences of the assumption relaxation, thus, consists of providing the ability to reason about the exposure of multiple primitive threat types, instead of only one. With reference to the constrained variant of the algorithm, the function f is now redefined to take the following parameters:

$$f(v, i, \{l_{c(1)}, l_{tot(v,1)}\}, \dots, \{l_{c(n)}, l_{tot(v,n)}\})$$

where each of the $\{l_{c(x)}, l_{tot(v,x)}\}$ pairs represents the distance of the path already covered and total distance of the path to cover, respectively, for the node v, as associated with the primitive threat type enumerated as x. The roles of $l_{c(x)}$ and $l_{tot(v,x)}$ are analogous to the roles of l_c and $l_{tot(v)}$ in the constrained OCD variant. The increase in the number of dimensions the algorithm operates over is $2 \times n$, with n being the total number of primitive threat types.

We specify the algorithm as follows. The first and the third case of the constrained OCD algorithm are adapted as:

$$f(\dots) = \begin{cases} 0 & \text{if } \forall x, l_{tot(x)} \leq l_{c(x)} \text{ ;} \\ \dots & \\ \infty & \text{if } i = 0 \text{ \& } is\_leaf(v) \text{ ;} \\ \dots & \end{cases}$$

In the first case, there is no more exposure of any threat type. In other words, the target distances for all the paths have already been covered. The third case remains the same as in the constrained OCD algorithm specification. The second case of the algorithm is specified as:

$$f(\ldots) = \begin{cases} \ldots \\ \sum_{k \in chld(v)} f(k, i_k, \{l_{c(1)}, l_{tot(k,1)}\}, \ldots, \{l_{c(n)}, l_{tot(k,n)}\}) & \text{if } i = 0 \ \& \ !is\_leaf(v) \ ; \\ \ldots \\ \ldots \end{cases}$$

This is a straightforward extension of the second case of the constrained OCD algorithm specification. The condition remains the same — it says that if we run out of sticks associated with the currently visited node v, and the node v is not a leaf-node, then we should look for separate covers for each of the subtrees below v. The final case, where we decide whether to use a stick under consideration (i) or not in the optimal solution, is expanded to:

$$\begin{array}{rcl} \min[& & f(v, i-1, \{l_{c(1)}, l_{tot(1)}\}, \ldots, \{l_{c(n)}, l_{tot(n)}\}), \\ & c_i \ + & f(v, i-1, \{l_{c(1)} + l_i, l_{tot(1)}\}, \ldots, \{l_{c(j)} + l_i, l_{tot(j)}\}, \\ & & \{l_{c(k)}, l_{tot(k)}\}, \ldots, \{l_{c(n)}, l_{tot(n)}\})] \end{array}$$

The condition associated with the rule remains the same as in the constrained OCD algorithm specification. The main point of the above rule is that the length of the stick i, given by l_i, is added solely to the partial solutions corresponding to primitive threat types the stick applies to. Stick i may contribute to the cover of a single path distance, if i represents a mitigation operation affecting threats of a single primitive type, or to the covers of multiple path distances, if the mitigation operations impacts on threats of a compound threat type. The above rule illustrates the latter case, in its second line. The choice of the particular threat types the stick i applies to, in the above rule specification, is arbitrary. Note, also, that we have implied that each threat mitigation operation that affects multiple threat types does so to the same degree. In other words, the l_i is constant across the threat types. This may not hold in the general case. To account for this, we only need to see l_i as a vector of threat type specific stick lengths. The algorithm is then adapted in a straightforward manner.

Non-Additive Transparency Functions

The main consequence of the relaxation of the constraint set in Section 4.5.5 is that threat mitigation operations cannot, in the general case, be represented as fixed length sticks. In other words, the stick-path analogy cannot be used for the problem statement and the algorithm specification as was done in the previous sections. The reason for this is the variability of the threat mitigation operations' relative offset range across the exposure degree scale, as discussed in Section 4.4.

As the first step in generalising the constrained OCD algorithm, we associate, with every threat mitigation operation, a set of pairs of LoEs of the form $(l_{h(igh)}, l_{l(ow)})$, where $l_h \gg l_l$. This is instead of associating the mitigation operations with fixed exposure degree impacts. The semantics of each of the pairs states that the respective mitigation operation reduces the LoE l_h to the LoE l_l, with respect to a single primitive threat type. This effectively defines a relationship on $M \times T \times L^2$, where M is a set of threat mitigation operations, T is a set of threat types and L is a set of LoEs. Secondly, we define the `offsets` mapping as:

$$\texttt{offsets} : M \times T \times L \longrightarrow L$$

Given a threat mitigation operation m, a threat type t and a LoE l_l, the function returns the corresponding LoE l_h. The returned value, l_h, may be the same as, or dominant to, l_l. The former is the case when there is no such l_h that the operation reduces to l_l. Finally, we alter the interpretation of some of the previously used notation. We redefine l_c of the constrained OCD to represent LoE that is mitigated by the current selection of mitigation operations. Also, $l_{tot(v)}$ of the constrained OCD is now taken to denote the LoE active at a container v.

The extension of the constrained OCD algorithm to account for the constraint relaxation can, now, be defined as:

$$f(v, i, l_c, l_{tot(v)}) = \begin{cases} 0 & \text{if } l_c \gg l_{tot(v)} \ ; \\ \sum_{k \in chld(v)} f(k, i_k, l_c, , l_{tot(k)}) & \text{if } i = 0 \ \& \ !is\_leaf(v) \ ; \\ \infty & \text{if } i = 0 \ \& \ is\_leaf(v) \ ; \\ \min[f(v, i-1, l_c, l_{tot(v)}), & \\ \qquad c_i + f(v, i-1, \texttt{offsets(}m\texttt{, } l_c\texttt{)}, l_{tot(v)})] & \text{otherwise} \ ; \end{cases}$$

The interpretation of the four cases of the above algorithm follows from that presented previously for the constrained OCD algorithm and its generalisation for the relaxation of the assumption. The algorithm can be extended to account for simultaneous exposure of multiple primitive threat types in a straightforward manner, analogously to the way in which the same has been done, in the previous section, for the constrained OCD algorithm. This is aided by the definition of the `offsets` mapping which takes explicit account of the threat typification. Thus obtained algorithm would represent the most general variant of the OCD algorithm.

4.6 Summary

In this chapter we have introduced the notion of information security vs. information utility seesaw and its balancing as the goal for the overall information exposure threat mitigation process. We have defined information utility with reference to a wider notion of utility as used in economics and philosophy and outlined a corresponding metric. The importance of information utility maximisation, in general, and its balance with the level of exposure protection provided, in particular, has been emphasised as one of the key factors for the user acceptance of the vision of pervasive computing. Contrasting the notions of risk and exposure degree we have introduced the threat significance level as a first step toward conceptualisation of structured reasoning about threat severity and mitigation. The next step was the definition of a LoE, and its corresponding modelling approach, as the instrument for matching the threat severity with the level of protection provided by mitigation operations. Based on the above concepts and inspired by the dynamic programming solution to the 0-1 Knapsack problem, we developed an algorithm that discovers the optimal, with respect to information utility and system usability impacts, threat mitigation strategy for a holistic realm in the face of a set of threats. Further generalisation of the algorithm is left to Chapter 6. The overall computational characteristics, resource requirements as well as the flexibility of the presented approach to threat mitigation, make it suitable for deployment in pervasive computing environments.

Chapter 5

From CASPEr to μCASPEr: Architecture & Policy Model

5.1 Chapter Overview

The focus of this chapter is on the presentation of μCASPEr, a policy-based specialisation of CASPEr and an adaptation of its architecture targeted at resource constrained devices. We set off with a high-level overview of the generalised CASPEr architecture, in Section 5.2, outlining its structure and describing the roles of its main components. Based on this, in Section 5.3, we introduce the "split" architecture of μCASPEr in which computationally intensive tasks are off-loaded from resource poor to more powerful devices, without hampering the operational independence or autonomy of the μCASPEr hosting platforms. μCASPEr, and its architecture, are founded on a policy model based on a variant of finite state automata called *Finite State Transducers with Tautness Functions and Identities* (TFFST) [BS04] — detailed in Section 5.4. The focus of the section is on the application of the TFFST policy model, in terms of its policy structure, to the CASPEr setting. We also show how a high-level, Deontic, policy language is used for μCASPEr policy specification. Having presented the μCASPEr architecture and the policy model, we follow on to describe in detail, in Section 5.5, the operation of μCASPEr. Specific attention is paid to the per-Client policy formation, including policy translation and conflict resolution, policy deployment, policy evaluation and policy enforcement stages. We round off the chapter by outlining the requirements and a high-level architecture of a data management model for CASPEr.

5.2 CASPEr Architecture Overview

CASPEr architecture is divided into four distinct layers: the *Threat Estimation Layer*, the *Information Exposure Threat Adaptation Layer*, the *Adaptation Enforcement Layer* and the *Realm Containment Modelling Layer*. Each of the layers relies on a number of components to provide for their functionality. Figure 5.1 shows a high-level view of the CASPEr architecture, components of each of the layers and inter- and intra- layer interactions. The figure also illustrates how CASPEr relates to the *Application* and *System* spaces of the traditional computer architecture. We use the term *space* instead of the standard term *layer* for semantic distinction with the notion of *layer* as used in the context of CASPEr. For clarity purposes, the figure does not show the interactions between the application space and the CASPEr layers related to CASPEr administrative tasks. The relative size of the individual layers, and their components, as represented in the figure, is not meant to be indicative of their internal complexity or resource requirements.

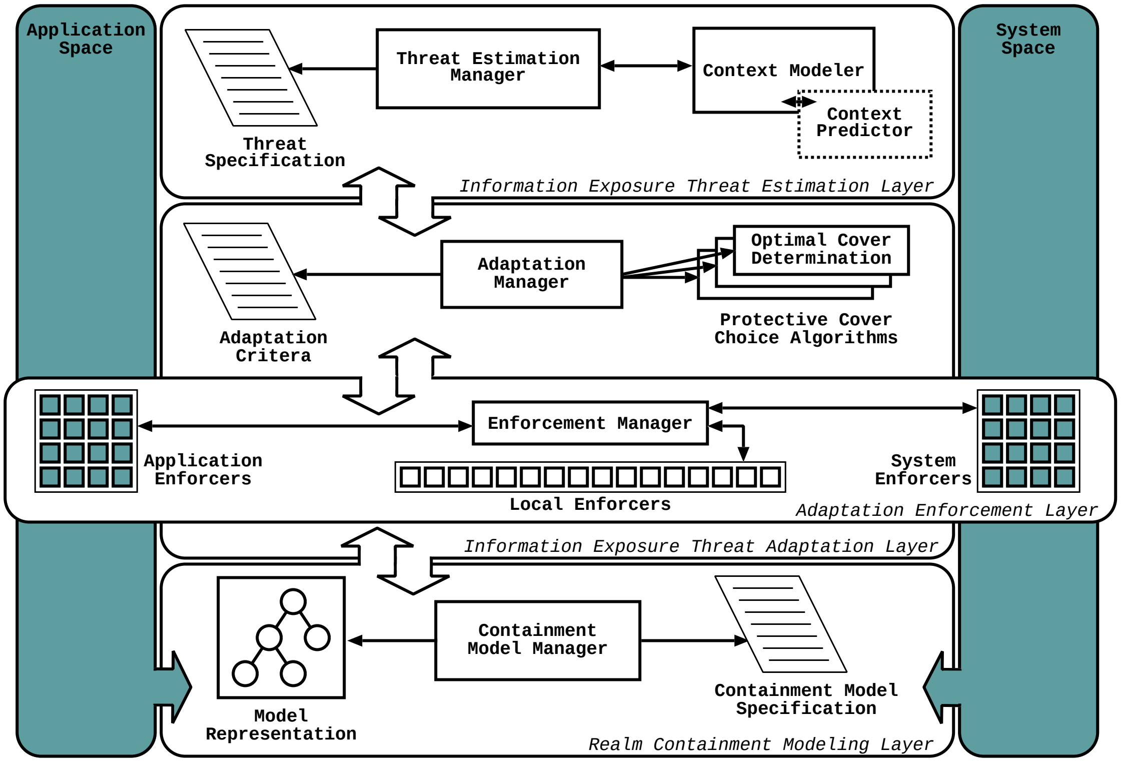

Application Space
Threat Specification
Threat Estimation Manager
Context Modeler
Context Predictor
Information Exposure Threat Estimation Layer
System Space
Adaptation Critera
Adaptation Manager
Optimal Cover Determination
Protective Cover Choice Algorithms
Application Enforcers
Enforcement Manager
Local Enforcers
System Enforcers
Adaptation Enforcement Layer
Information Exposure Threat Adaptation Layer
Model Representation
Containment Model Manager
Containment Model Specification
Realm Containment Modeling Layer

Figure 5.1: High-level representation of the CASPEr architecture.

5.2.1 CASPEr Components

Threat Estimation Layer. The role of the *Threat Estimation Layer* is to establish, through its *Threat Estimation Manager* component, the presence of information exposure threats based on the context sensed by the *Context Model* component. The correlation between contextual states and information exposure threats and their characteristics is specified by the *Threat Specification* component. Thus established threat state is fed to the *Information Exposure Threat Adaptation Layer*. The Threat Estimation Layer supports proactive CASPEr operation through its optional *Context Predictor* component. The role of the Context Predictor is to "foresee" contextual states that will arise in the near future based on current state of context, as captured by the *Context Model*, or on the higher level data such as e.g. mobility profiles, diaries and schedules, explicit user input etc.

Realm Containment Modelling Layer. The main responsibility of the layer's *Containment Model Manager* component is to maintain an accurate state of the containment-based model of the realm, referred to as the *Model Representation* in the figure. The Containment Model Manager handles model update operations coming from the *Application* and *System* space components as well as from the *Information Exposure Threat Adaptation Layer*. The containment model is built and managed abiding by the *Model Specification*. The Realm Containment Modelling Layer also plays an important role in providing the Information Exposure Threat Adaptation Layer with the information and the functionality necessary for the threat propagation and mitigation processes.

Information Exposure Threat Adaptation Layer. For simplicity purposes, we refer to this layer as to the *Adaptation Layer*. The main responsibilities of the Adaptation Layer are the establishment of the LoEs experienced by data items in a realm, choice of the adequate threat mitigation strategy as well as the coordination of the threat mitigation process. The majority of tasks are accomplished by the *Adaptation Manager* component. A number of algorithms for the optimal threat mitigation strategy, such as the OCD presented in Section 4.5, can be supported through the *Protective Cover Choice Algorithms* component. Their operation is guided by the *Adaptation Criteria* — consisting of any information required to support the threat adaptation process, other than that provided by the Realm Containment Modelling Layer. The Adaptation Manager initiates and monitors the actual threat mitigation process carried out by the components of the *Adaptation Enforcement Layer*.

Adaptation Enforcement Layer. The sole role of the *Adaptation Enforcement Layer* is to realise the adaptation strategy decided upon by the Adaptation Layer. The *Enforcement Manager* coordinates the various *Enforcers* in accomplishment of the task. Every Enforcer implements a single adaptation functionality, i.e. a class of information exposure mitigation operations. As containers may originate from both the *Application* and *System* spaces, the Adaptation Enforcement Layer encompasses relevant Enforcers from the two spaces as well as its internal, *Local*, Enforcers.

The diagram in the Figure 5.2 shows an example relationship among traditional system architecture components at the application and operating system levels and the corresponding containment-based model of the world. It also demonstrates the role of the CASPEr aware-

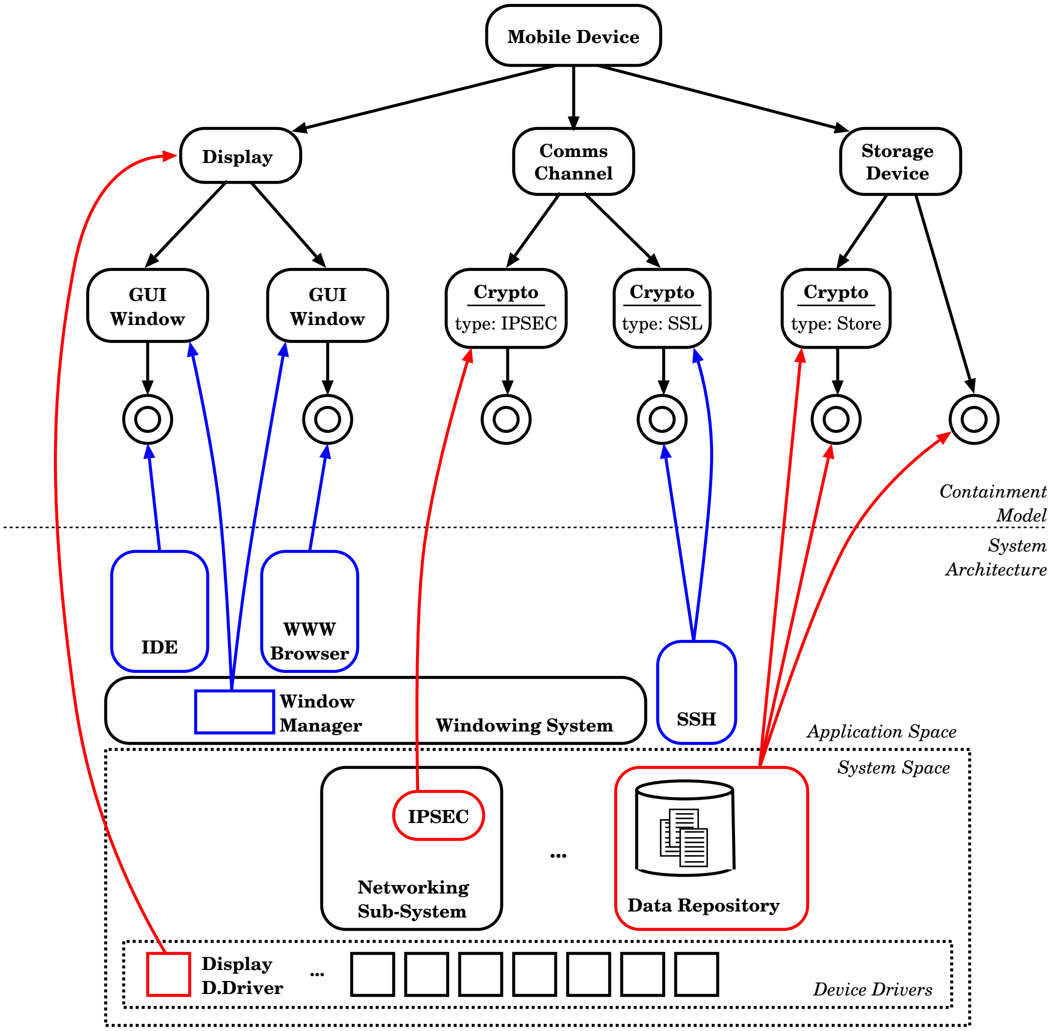
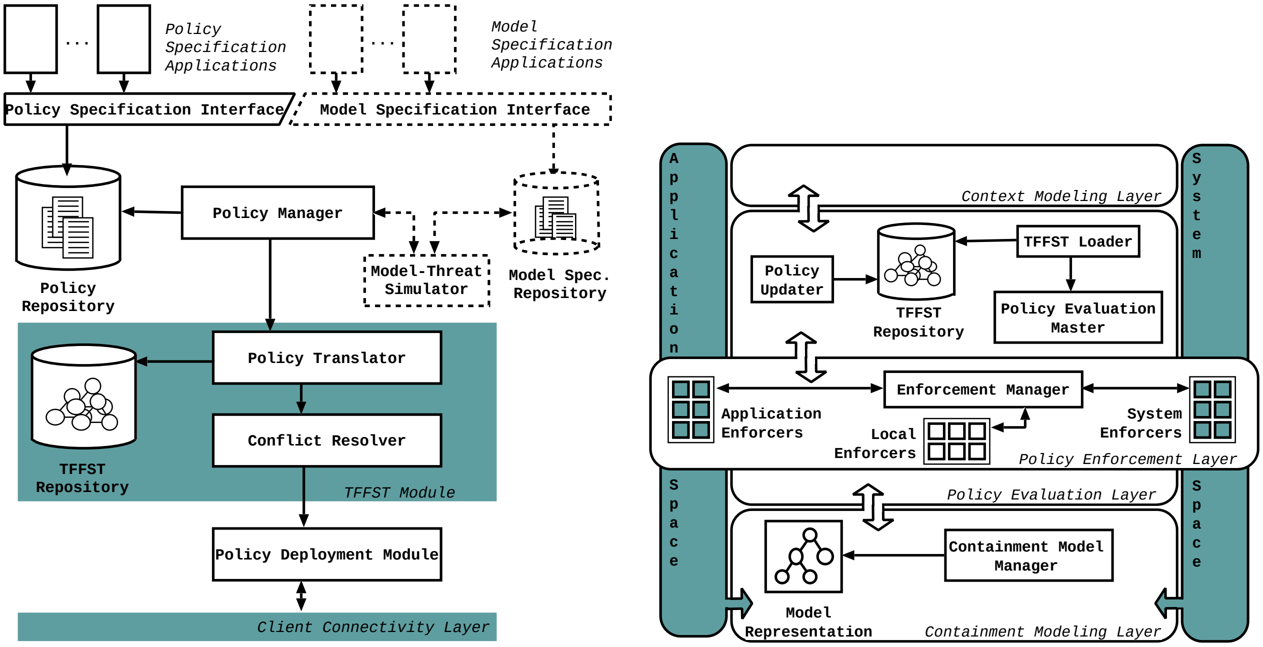


Figure 5.2: Example component authority - container mapping.

ness at both the levels. The lower part of the figure (below the dashed horizontal line) shows a simplistic diagram of a system architecture, while the upper part depicts containment-based representation of the realm. The coloured arrows link individual application and system level components with the containers they represent the component authorities for.

5.3 μCASPEr Architecture

In the previous section we have outlined the generic CASPEr architecture and its most important components. Recognising the likelihood of high degrees of resource poverty among ubiquitous computing devices — ranging from mobile phones and PDAs to tiny sensor platforms, we developed μCASPEr — a policy-based specialisation of CASPEr suitable for deployment on resource constrained devices. In this section we provide a high-level outline of the μCASPEr architecture. The specific roles of the individual components at various layers in the architecture are specific to the policy model we employ. Thus, their detailed description is postponed until section 5.5, after we have introduced the TFFST policy model.



(a) Management side.

(b) Client side.

Figure 5.3: μCASPEr architecture.

Figure 5.3 depicts a high-level view of the "split" system architecture consisting of the *Management* side, Figure 5.3(a), and the *Client* side, Figure 5.3(b). The goal behind the separation was to off-load computationally and resource intensive policy model related tasks from the resource constrained Clients onto the resource powerful Management side. The main challenge in accomplishing this was retaining the independence and operational autonomy of individual Clients — crucial for continuous and efficient information exposure protection in pervasive computing environments. The fundamental role in meeting the challenge was the particular choice of the policy model — based on *Finite State Transducers with Tautness Functions and Identities* (TFFST), and the manner in which it is deployed in μCASPEr. In the TFFST model, it is the policy pre-processing stage that incurs the highest, and possibly prohibitive, computational complexity and resource consumption. Consequently, these tasks are assigned to the unconstrained Management side where they are accomplished *a priori* to the actual policy deployment on the Client side. Not only does this save on the resources consumed at the Clients but it also fully honors their operational autonomy.

Namely, the Client side is where the policy decision making and enforcement take place while the role of the Management side components begins with policy specification in a high-level language and ends with the delivery to the Client of a deterministic, conflict-free TFFST representation of the policy.

5.3.1 The Management Side

Components of the Management side, Figure 5.3(a), handle policy management tasks that are detached from runtime policy evaluation and enforcement and would otherwise, due to the associated resource requirements, seriously hamper Client operation.

Policy Editor and Policy Specification Interface. To explicitly specify policy rules in a high-level policy language — Ponder [DDLS01] in our case, the policy administrator[1] uses a policy editor application. The Ponder distribution itself [Pon] provides for both a dedicated editor as well as a policy compiler into Java [Micb]. We envisage other high level applications would be developed to support policy specification in a user-friendly manner acceptable to an average user (e.g. graphical). The obtained high-level policy rule specifications are stored in the *Policy Repository* via the *Policy Specification Interface* (PSI).

Policy Manager (PM). The role of the *Policy Manager* (PM) is to formulate a specific policy for every μCASPEr Client in an on-demand fashion. The per-Client policy consists of a set of policy rules applicable to the containment configuration as supported by the Client's software and hardware platform. The individual policy rules are either obtained from the policy repository, having been explicitly specified by the policy administrator, or are generated in an automatic fashion by the *Model-Threat Simulator* component based on the containment model specification and the Client's *profile*. The latter process consists of simulating the threats which can affect the Client (based on its profile) and generating the corresponding policy rules accordingly. The related components are depicted dash-lined as they are considered optional.

TFFST Module. The components of the TFFST module[2] are responsible for producing a conflict-free, deterministic TFFST policy representation of the per-Client policy. The *Policy Translator* transforms the high-level policy rules specification into their TFFST form while the *Conflict Resolver* (CR) ensures that they are deterministic. Thus obtained TFFSTs are archived in the *TFFST Policy Repository* for reference (e.g. audit, reuse, etc.). Inherent in the process of building a TFFST and its determinization is conflict resolution (Section 5.5.3)

[1] Generic term to refer to a person in charge of the policy specification and maintenance.

[2] The components of the TFFST module are shared with the work presented in [VBS$^+$05] due to the common policy model deployment. This is acknowledged in the declaration accompanying this thesis and also in [DBVC06] which represents the result of the collaborative work on the topic.

— thus the name of the CR component. The algorithms involved in the conflict resolution process are of high complexity and incur considerable computational costs — thus their offloading onto the Management side. TFFST evaluation by the Clients, on the other hand, is particularly lightweight, as shown later in the chapter and further discussed in Chapter 6.

Policy Deployment Module (PDM). PDM serves as the interface component between the Management and the Client sides. It accomplishes the tasks of mutual authentication of the two sides, initiation of the policy formation process at the PM and download of the TFFST per-Client policy to the Client.

5.3.2 The Client Side

The main role of the Client side (Figure 5.3(b)) is the policy evaluation and enforcement. In other words, the information exposure threat evaluation and adaptation processes. The layering in the μCASPEr Client side architecture follows the generic CASPEr architecture (Figure 5.1). However, each of the layers offers a scaled down functionality, in terms of components' roles and responsibilities, due to the envisaged Clients' resource constraints and adapted to the μCASPEr policy model.

Context Modelling Layer (CML). The role of the CML is to establish the presence of contextual states relevant for the threat model analysis and provide this information to the *Policy Evaluation Layer*. For the purpose, the CML may leverage any adequate, trusted services available in the environment. Unlike the corresponding layer in the CASPEr architecture, CML lacks the explicit threat estimation process — in fact, the process does not occur at all on the Client side. The mapping of contextual states to the threats is implicit in the per-Client policy itself.

Policy Evaluation Layer. Components of the *Policy Evaluation Layer* are responsible for the evaluation and enforcement of policies driving the information exposure adaptation process of a ubiquitous computing device. The *Policy Evaluation Master* (PEM) serves the role of the *Policy Decision Point* (PDP) as defined by the Policy Core Information Model (PCIM) [MESW01]. It receives events denoting information exposure threat relevant contextual states from the CML, both synchronously and asynchronously, and evaluates the relevant TFFSTs to decide on threat mitigation operations to be executed by the *Policy Enforcement Layer*. The *TFFST Repository* stores the per-Client policy, obtained from the Management side by the *Policy Updater*. The role of the *TFFST Loader* is explained in Section 5.5.

Policy Enforcement Layer. The *Policy Enforcement Layer* acts as the *Policy Enforcement Point* in the PCIM. The functionality provided by the layer and roles of the *En-*

forcement Manager (EM), as well as individual *Executors*, are analogous to their CASPEr counterparts.

Containment Modelling Layer. The *Containment Modelling Layer* consists of the *Containment Model Representation* data structure and the *Containment Model Manager* (CMM) active component. As the policy evaluation is fully TFFST driven, the model representation itself is rudimentary, consisting of containment path-like expressions, supporting only event triggering at PEM on the model state updates. The events initiate threat model reevaluation by PEM as well as the TFFST policy management operations detailed in Section 5.5. The overall role of the layer is analogous to the corresponding one in the full CASPEr architecture but tailored to the poorer containment model representation.

5.4 The Policy Model

To be feasible, a policy model intended for ubiquitous computing systems has to take into account constraints imposed by enforcement platform heterogeneity in terms of the variability in its resources and capabilities. Thus, the policy model needs to provide for:

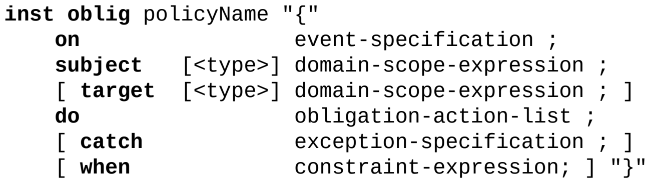
- Deployment of policies tailored to the resources, capabilities and usage patterns of the policy enforcement ubiquitous device.
- Formulation of a policy in a way that leads to unambiguous decisions in the shortest possible time and with the least computational complexity.
- Specification of a policy in compact and portable manner.

The policy model presented in this section is specific to the μCASPEr and the requirements posed by its target deployment environment. Its expressive power is thus confined to the concept of *obligation*, as detailed below. Consequently, the policy model is not directly generalizable to the wider CASPEr setting where substantially greater expressiveness is needed.

5.4.1 Ponder as a Deontic Policy Language

The policy model we propose uses Ponder [DDLS01] as a high level language for security and management policy specification. Ponder is a declarative, object oriented, policy language based on Deontic concepts. Deontic logic is a modal logic used to describe and reason about the concepts of *obligation* and *permission*. Although other languages founded on Deontic concepts have been developed, such as e.g. Rei [KFJ03], we opted for Ponder owing to the wide availability of open source tools for policy management, syntactic and semantic policy analysis and checking and direct policy transformations into XML and translations into Java. The source code availability was required as we needed to adapt Ponder slightly to the target application, as described in Section 5.5.

Following the Deontic concepts, the two fundamental policy types provided by Ponder are *authorisation* and *obligation* policies. In this work we make use of the latter only. Obligation policies specify events and actions that must be performed on the events occurrence. The syntax of obligation policies in Ponder is shown in Figure 5.4.



```
inst oblig policyName "{"
    on                  event-specification ;
    subject   [<type>] domain-scope-expression ;
    [ target  [<type>] domain-scope-expression ; ]
    do                  obligation-action-list ;
    [ catch             exception-specification ; ]
    [ when              constraint-expression; ] "}"
```

Figure 5.4: Ponder obligation syntax.

Language keywords are in **bold**. Optional elements are enclosed in `[]` and repetition is specified within {}. The policy specifies actions, denoted by the `do` statement, that a `subject` in a domain must do, specifying the `target` domain unless actions are internal, `on` an event occurrence. Constraints, specified by the `when` keyword, are optional and may be used to limit the applicability of a policy. Policies are not element order sensitive. The optional `catch` clause specifies exceptions to be thrown in case the specified actions fail to execute. Concurrency operators may be used in action specification in cases multiple actions are to be performed sequentially (→) or in parallel (||). Composite events may also be specified using event composition operators. Examples are to be presented shortly.

5.4.2 Policy Specification

Our policy specification model follows the *Event–Condition–Action* (ECA) paradigm in which an overall policy is broken into multiple fragments. Each fragment represents a rule specifying an action to be taken in response to a predefined set of conditions, triggered by an event or a set of events. We refer to the policy fragments as the policy rules. We use Ponder [DDLS01] obligation policies to represent individual rules. The following is a sample policy.

Rule 1:

```
inst oblig PolicyRuleNo001 {
  on context.OwnerAway;
  subject s = PolicyEvaluationMaster;
  target t = ContainmentModelManager;
  do s.activateLoE(LoE.Physical.3, */storage/data_item:sclass=S);
  when t.isActive(*/storage/data_item:sclass=S);
}
```

Rule 1 specifies that on occurrence of the contextual state denoted as `OwnerAway`, the `PolicyEvaluationMaster` (the PEM μCASPEr component) is to `activate` the particular Level of Exposure (`LoE.Physical.3`) for all data items of the particular security classification (`sclass=S` — Secret) contained on every storage device within the realm. In practise, a realm represents a single ubiquitous computing device. Note that a single contextual effect may imply multiple threat types and also multiple LoEs for each of the threat types as experienced by data items in different containments. Each of these may be specified in a separate policy rule analogous to the Rule 1. The condition (`when`) specifies that the rule should be executed only for the matching containments that are instantiated, i.e. are *active*, in a realm at a point in time — as determined by the policy `target`, the `ContainmentModelManager` (CMM in the μCASPEr architecture). Notice the use of containment path expression to match multiple actual data items and their respective containments. Effectively, the above policy rule serves the role of initiating the adaptation process on the target containment (see Rule 2 as an example follow-up policy rule).

Rule 1 is a representative of the rules conditioned on the occurrence of a threat. The other category of policy rules as deployed in CASPEr is the ones conditioned on the activation of a LoE, as triggered by the CMM. Examples are shown in Rule 2 and Rule 3 below:

Rule 2:

```
inst oblig PolicyRuleNo002 {
  on LoE.Physical.3;
  subject s = EnforcementModule.Storage;
  do s.destroy(*/storage/data_item:sclass=S);
}
```

Rule 2 specifies that on the activation of the LoE `LoE.Physcal.3` all data items of security class `S` contained within any storage device in the realm must be destroyed. This is to be accomplished by the `Storage` EM component registered as the provider of the respective threat mitigation operation for the particular containment. Note that the destruction operation for different storage devices may be provided by different Enforcers. Rule 2 is the followup of the Rule 1, should it result in the actual activation of the LoE.

Rule 3:

```
inst oblig PolicyRuleNo003 {
  on LoE.Optical.2;
  subject s = EnforcementModule.WindowManager;
  target t = Container(*/display/gui_window:size=MEDIUM);
  do t.shrink(SMALL);
  when t.contains(data_item:sclass=S); }
```

Rule 3, as a further example, demonstrates the use of the `target` clause. It specifies that on LoE denoted as `LoE.Optical.2` the EM is to, via its `WindowManager` component, shrink all *GUI Window* containers of size `MEDIUM` that contain a data item of security class `S`.

5.4.3 An Evaluation Model Based on Finite State Transducers

Finite State Automata (FSA) are classical computational devices used in a variety of large-scale applications. A Finite State Transducer (FST) is a FSA whose transitions are labelled with both an input and an output label. FSTs are exploited for efficient, time critical, decision making in a wide range of fields — especially where the expressed rules exhibit inherent ambiguities, for their conflict resolution properties. Perhaps the most notable example is in the area of speech processing, where they are typically used to reason about highly ambiguous grammatical rules in a timely manner. Furthermore, FSM based solutions are typically lightweight, representationally as well as computationally: they can be implemented as arrays of states, transitions and pointers among them without requiring heavy management structures.

To represent policies we are using the FST-based model presented in [BS04] called *Finite State Transducer with Tautness Functions and Identities* (TFFST). In particular, we leverage *Deterministic Transducers* (DT), a category of transducers free from ambiguities. For a given class of symbols and a state those transducers have only one outgoing edge with a matching label. DTs are computationally interesting as their evaluation time does not depend on the size of a transducer, but on the length of the input. This is as the evaluation process consists solely of following the only possible path corresponding to the input and writing consecutive output labels as encountered along the path [RS97].

The TFFST model is further based on a modification of predicate augmented FSTs [vNG01], replacing predicates by a metric representing the distance between a policy and a given event. A policy condition defines a delimiting region specifying where a complying event can lie. When an event happens to be inside two or more such overlapping regions a *modality* conflict arises. To resolve modality conflicts, rather than reasoning how far from a region's border the event lies, the concern is how *tautly* the region's defining condition fits to an event. The result of the conflict resolution process, thus, is the policy defining the condition that is the most *taut* around the event.

To quantitatively represent the aforementioned *tautness*, a metric called *Tautness Function* (TF) is used. TF is a real number in the interval $[-1, 1]$. The more taut a condition is around an event the closer its TF is to zero.

Definition. *A Tautness Function associated with a condition c, denoted τ_c, establishes a mapping from $E \times C$ to the real interval $[-1, 1]$ where:*

- *E is the set of possible events or attempted actions,*
- *C is the set of policy conditions,*

- $\tau_c(e) \in [-1, 0) \Leftrightarrow e\, does\, not\, satisfy\, c,$

- $\tau_c(e) \in (0, 1] \Leftrightarrow e\, satisfies\, c,$

- $\tau_c(e) = 1 \Leftrightarrow \forall f \in E, f\, satisfies\, c,$

When the TF is modelling the condition part of the rule, the subject or any other property of the condition, such as temporal constraints, are included in the condition c. In the same manner, when the TF is modelling the action part of a rule, the condition c includes the target or any property of the action.

Even though the *tautness* in the range of $[-1, 0)$ denotes that event does not satisfy the condition, its fine-grained expression is required for certain FST manipulation operations, such as complementation. The TF of 1 denotes all possible events. As an example, consider two policy rules: the first is applicable, as defined by its condition, to all containers of class *Storage* while the second applies to removable storage devices only. The LoE specified in the rules is activated only for the removable storage devices when, given a set of events, the TF evaluation for the second rule yields a positive value closer to 0 than the value obtained for the first rule.

The concept of tautness functions represents a sort of abstraction layer over the application specific issues that actually determine how TFs are implemented and how they map events and conditions to real numbers. The most significant advantage of using TFs in μCASPEr is the ability to define a different way to model each decision making attribute, such as containment configuration or information utility measure, and combine them using the algebra for TFs. The algebra defines basic logic operators: *disjunction*, *conjunction* and *negation*; plus two new operators called *tauter-than* ($\rightarrow_\tau$) and *as-taut-as* ($\rightleftarrows_\tau$) specially formulated to express the concept of distance as used in TFs. Due to the space constraints for detailed presentation of the above concepts please refer to [BS04].

The main idea of the TFFST policy model is to replace the classic input and output symbol labels on FSTs with TF labels expressed in the form of the TF algebra expressions. Thus, instead of matching an incoming symbol with a label on a transition, a transducer will evaluate the corresponding TF. Furthermore, following a transition it will produce an event with a positive (or zero) value for the TF in the outgoing label. TFFSTs are formally defined as:

Definition. *A Finite State Transducer with Tautness Functions and Identities (TFFST) is a tuple* (Q, E, T, Π, S, F) *where:*

- Q *is a finite set of states,*
- E *is a set of symbols,*
- T *is a set of tautness functions over* E*.*
- Π *is a finite set of transitions* $Q \times (T \cup \{\epsilon\}) \times (T \cup \{\epsilon\}) \times Q \times \{-1, 0, 1\}$[3]*.*
- $S \subseteq Q$ *is a set of start states.*
- $F \subseteq Q$ *is a set of final states.*
- *For all transitions* $(p, d, r, q, 1)$ *it must be the case that* $d = r \neq \epsilon$*.*

In the implementation, we use an extension of the above definition to let the transducer deal with strings of events and actions in each transition. μCASPEr policy rules are modeled using TFFSTs, in which the incoming label represents the condition and the outgoing label the action (see Figure 5.5).

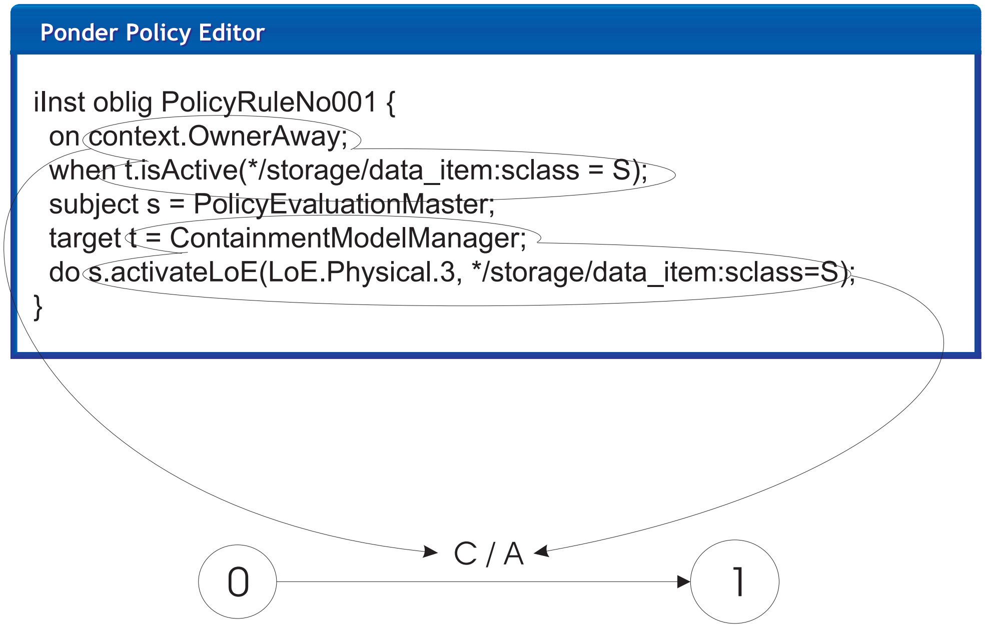
5.4.4 Modelling Policies with TFFSTs

To clarify the use of previously introduced concepts for policy modelling we show how *obligations* and *constraints* are expressed. TFFSTs may also be used to model *authorisations*, *prohibitions* and *dispensations* [DDLS01], however, these policy rule types are beyond the current CASPEr requirements.

Obligations

An *obligation* is a rule expressing that when an event matches a particular condition, a specified action must be executed. Typically, the input will report occurrence of an event, or a set of events, while the output denotes the execution of an action or set of actions

[3] The final component of a transition is an “identity flag” used to indicate when an incoming event must be replicated in the output.



```
iInst oblig PolicyRuleNo001 {
  on context.OwnerAway;
  when t.isActive(*/storage/data_item:sclass = S);
  subject s = PolicyEvaluationMaster;
  target t = ContainmentModelManager;
  do s.activateLoE(LoE.Physical.3, */storage/data_item:sclass=S);
}
```

Figure 5.5: Policy translation process example.

specified by a policy rule. However, other combinations are also possible, for instance, to be *unobtrusive* (as defined by [CLN03]), the input can be replicated in the output.

A clear view of the links between objects in a Ponder policy rule specification and the elements of a TFFST is shown in the Figure 5.5. As shown in the figure, the *obligation* is represented by a transducer that consumes events and produces actions. The transducer has two states and a single transition between them. The label on the transitions is divided into the left and the right part by the delimiter `/`. The former represents the condition of the obligation (`on` and `when` clauses) while the latter corresponds to the action to be executed (`target` and `do` clauses).

Rule 4 is an example of a μCASPEr policy rule in which an action is conditioned on the occurrence of more than a single event.

Rule 4:

```
inst oblig PolicyRuleNo004 {
  on context.OwnerAway → TimerOver(delay);
  subject s = PolicyEvaluationMaster;
  target t = ContainmentManager;
  do s.activateLoE(LoE.Physical.3, */storage/data_item:sclass=S)
  when t.isActive(*/storage/data_item:sclass=S)
}
```

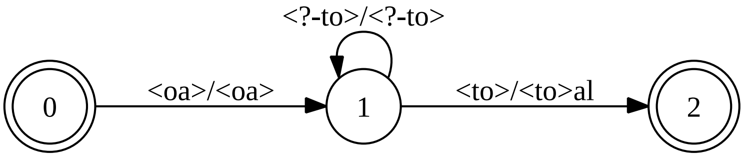


Figure 5.6: Rule 4 TFFST representation.

The rule specifies that the LoE (`LoE.Physical.3`) is to be activated only after the contextual state — the device owner not within a pre-defined proximity, remains valid for the specific time period (`delay`). Figure 5.6 shows a TFFST representation of the Rule 4. As the policy rule effectively postpones the threat mitigation, the delay has to be chosen so that it does not provide for a credible window-of-opportunity for information exposure with respect to the particular threat type.

The TFFST notation as used in the figures throughout this chapter is as follows. The symbol "?" substitutes *any event* while the term `epsilon` stands for the *null event* — a dummy event executed instantaneously. The symbol "–" represents set substitution operator. Symbols "<" and ">", enclosing the labels, denote input-output TF identity. The labels typically represent event and action names in an abbreviated, two-letter, form. For example, the *context.OwnerAway* event is denoted by the label "oa", the *TimerOver* is abbreviated "to" while the "al" label corresponds to the *activateLoE* action. By convention, the state enumerated as "0" is the *initial*, i.e. starting, state while the double circled states are *final*, i.e. FSA termination, states.

Constraints

Constraints are built into a TFFST policy model using the FST *composition* operation, as defined in [BS04]. TFFST composition is an analogue operation to the standard function composition. A constrained TFFST policy is obtained by composing the single TFFST representing all obligation policy rules with the TFFSTs corresponding to the constraint policy rules.

To demonstrate how the constraints work consider the Rule 5 below. In cases of frequent interleaved occurrences of events that lead to mutually exclusive states, denoting a fluctuating presence and a variable degree of a threat, the system may suffer from a "ping-pong" effect. For example, referring back to the Rule 4, the ping-pong effect would be caused by a mobile device's owner repeatedly walking away from and coming back into the vicinity of the device. The ping-pong effect has undesirable consequences for resource availability as well as for the μCASPEr threat mitigation efficiency. One possibility for avoiding the effect is through specifying a constraint on the "opposing" events, as shown in Rule 5:

Rule 5:

```
inst oblig PingPongConstraint{
```

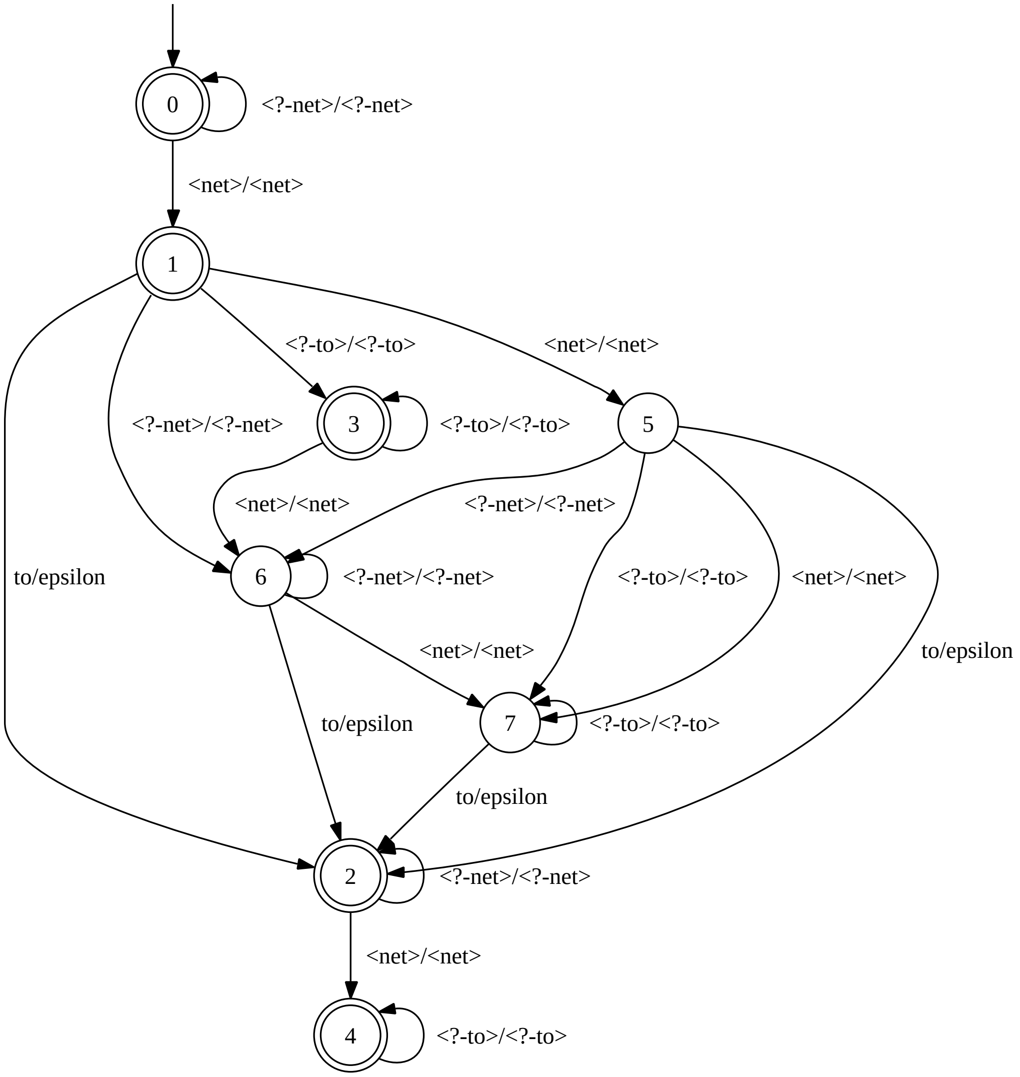


Figure 5.7: Rule 5 (Constraint) TFFST.

```
  on context.NoExplicitThreat;
  subject s = PolicyEvaluationMaster;
  do s.ignoreEvent(context.NoExplicitThreat);
  when s.pingPongPeriod(time);
}
```

The NoExplicitThreat event denotes absence of any information exposure threat. The

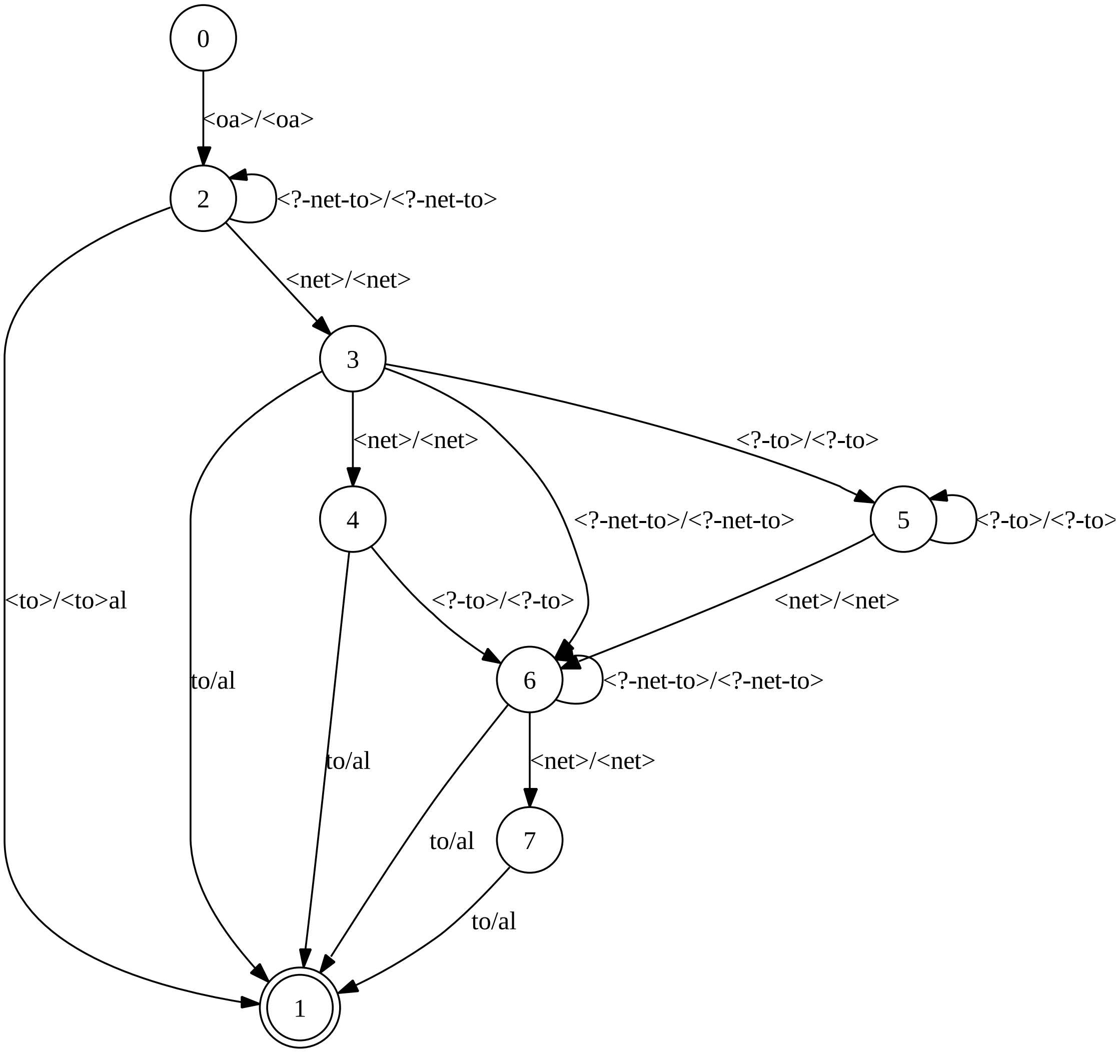


Figure 5.8: Rule 4 and Rule 5 composition TFFST.

effect of the event is lowering the LoEs to their “default” values as set by a policy model. For an example specification of a complete μCASPEr policy and a set of corresponding TFFSTs please refer to Appendix A. If the default is not defined, all LoEs for all data items in the realm are reduced to the *NULL* LoE. The TFFST in Figure 5.7 represents the above (Rule 5) constraint while the Figure 5.8 shows the result of its composition with the TFFST from Figure 5.6.

5.4.5 Per-Client Policy Structure

In the above section we provided a general description of the TFFST policy model. Going one step further, here we present the specific structure in which the policy model is deployed on per μCASPEr Client basis.

The particular μCASPEr per-Client policy structure is influenced mainly by the following two factors: i) the developed protection model is fully information-centric — i.e. the threat adaptation process is driven by the LoEs as experienced by the individual data items on the matching granularity; and ii) the LoEs as experienced by the individual data items due to the same set of exposure threats may differ subject to their respective containment configurations. Considering these two factors, we model the per-Client policy as a set of independent sub-policies specified on per data item containment basis. A data item containment is a sequence of nested containers starting at a containment tree root and ending at a container of class *Data Item*. In other words, every data item containment supported by Client's software and hardware platform has a single TFFST associated with it. The overall policy for a Client is a collection of all such TFFSTs. Individual TFFSTs can be characterized as stand-alone for they allow a fully independent threat reasoning and adaptation of the data item containments they are associated with.

The set of TFFSTs corresponding to the data item containments instantiated in a realm at a point in time is referred to as the *active* policy set. It is only the TFFSTs from the active policy set that get evaluated by PEM as detailed in Section 5.5. The maintenance of the active policy set is accomplished by the *TFFST Loader* component of the Client's Policy Evaluation Layer as a response to the containment model state update events generated by the CMM.

Figure 5.9 depicts an example active policy structure for a ubiquitous computing device. The symbols representing the individual TFFST policies are positioned below the data items whose containments they correspond to. Furthermore, the TFFSTs are labeled by the path expressions matching the data item containments they are associated with.

The adopted approach of per-Client policy structuring has a number of advantages over the alternative of expressing the per-Client policy as a single TFFST. Firstly, it represents a clear tradeoff between the number of TFFSTs required to express the per-Client policy and the individual TFFST complexity. As discussed in Section 6, the simplification to the structure of individual TFFSTs in the adopted approach by far outweighs any overheads brought about by their increased quantity (in terms of size and evaluation complexity). Secondly, the adopted approach is far more flexible as the support for new container classes can be added in a modular fashion solely by obtaining the corresponding TFFSTs.

Appendix A presents an example of how a per data item containment policy is derived from an overall μCASPEr policy specification. A selection of corresponding TFFSTs is also shown.

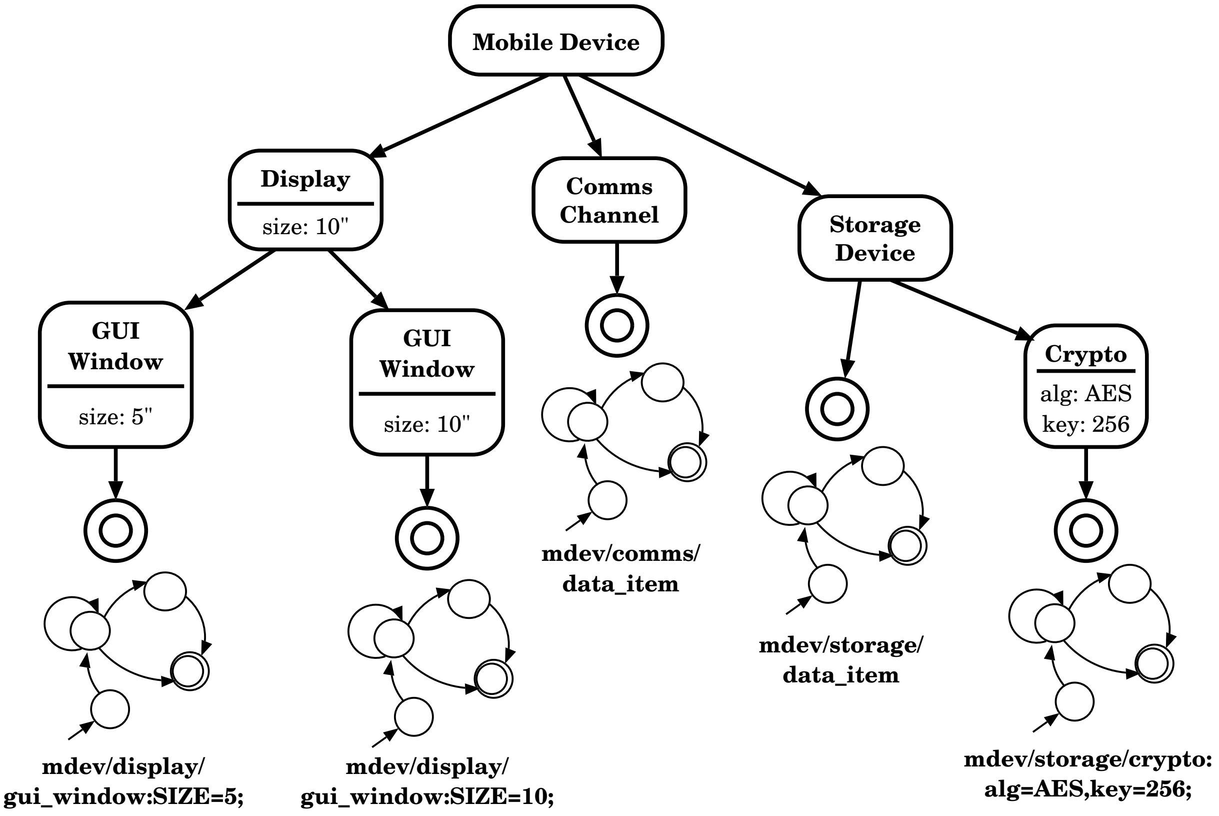


Figure 5.9: Example active Client TFFST policy structure.

5.5 μCASPEr Operation

In this section we describe the processes related to the μCASPEr operation. They accomplish the tasks required to generate, deploy, manage and evaluate the TFFSTs comprising the μCASPEr per-Client policy. The processes are divided between the Management and the Client side (Figure 5.3). The latter provides for the policy specification and generation (Section 5.5.1), policy translation (Section 5.5.2) and conflict detection and resolution (Section 5.5.3) which precede the client policy deployment (Section 5.5.4). The latter caters for the runtime policy evaluation (Section 5.5.5), including relevant TFs computation (Section 5.5.6) and policy enforcement (Section 5.5.7).

5.5.1 Policy Specification and Generation

μCASPEr provides for two types of per Client policy definition: explicit policy specification and automatic policy generation through simulation.

Explicit Policy specification assumes user input of individual policy rules forming a policy. The specification process may be direct or indirect. The former assumes explicit policy rule specification in Ponder high-level policy specification language. Although the explicit policy specification approach offers increased flexibility and expressiveness the disadvantage is the requirement of familiarity with Ponder and the related technical issues. We envisage dedicated applications to be developed on top of the μCASPEr Policy Specification Interface to provide for intuitive ways of policy specification trading off flexibility for user friendliness. This would allow for μCASPEr policies to be specified and administered by the "average" user of the target deployment platforms. Figure 5.3 depicts where in the μCASPEr architecture these policy specification applications fit. Both approaches leverage the PSI to store specified Ponder policies in the Policy Repository.

Per Client policy may also be obtained through simulation by the PM Simulator component (Figure 5.3). Leveraging individual container and threat specifications, as defined in Chapter 3, PM can simulate effects of all contextual states that a Client is capable of modelling for all supported containments. The relevant Client's capabilities are provided in the form of a profile in the policy deployment phase, as explained in Section 5.5.4. The result of the simulation process is a set of Ponder policy rules forming a Client's policy. The μCASPEr architecture provides for the containment model and threat model specification through a dedicated interface — the Model Specification Interface. Both the interface and the related policy generation components in the Figure 5.3 are depicted using dashed lines. This is to denote that the components are optional. We expect that in the majority of μCASPEr deployments direct policy specification will be used exclusively, for simplicity reasons.

Explicit policy specification and policy generation through simulation, however, are not mutually exclusive processes. Most notable example of the symbiosis is when the former is is used for specifying a policy default while the latter is used to define behaviour relative to the default. The symbiosis gives rise to a type of static policy conflicts easily resolvable by the PM. The conflict occurs when an explicitly specified policy rule matches with a simulation generated policy rule on the containment they apply to. The resolution is based on a simple rule stating that explicitly specified policy rules always have precedence over the automatically generated ones. Explicit user arbitration may be considered in cases of exceptional ambiguity or for error elimination. This type of conflict is resolved prior to the TFFST policy model conflict resolution stage, described in Section 5.5.3.

5.5.2 Policy Translation

Policy translation from high-level languages into internal policy evaluation models, in our case from Ponder specifications into TFFSTs, can be a complex task that needs to be kept simple and ad-hoc. The translation process follows the principles outlined in Section 5.4. Example of the mapping between objects of a Ponder policy and corresponding TFFST components has been shown in Figure 5.5.

One of the challenges of the design was the association of TFs to Ponder policies. To accomplish this, and support the translation process, the Ponder distribution [Pon] had to be altered slightly. Abiding by the object oriented methodology used by Ponder we leverage the methods specified in the `do` clause, associated with `subject` and `target` objects of policy rules, as hooks for TFs. Thus, when `subject` or `target` methods are invoked to check the `when` clause, a corresponding method is executed at the same time to assign a TF value instead of the boolean value that Ponder assigns to the condition. The result of the TF computation is assigned to the policy rule instead of the boolean value that Ponder originally evaluates the condition to. For conditions represented by logic combinations of simple conditions the TF algebra remains valid.

One of the advantages of the above approach is that the methods implementing TFs can be developed explicitly and provided to the PEM externally. This effectively enables PEM behaviour to be customised in a flexible and lightweight manner. For example, TF computation based on different parameters may be used to provide for different policy rule prioritisings on per realm basis.

5.5.3 Conflict Resolution

An advantage of using transducers to model policies is the rich set of operations available. Transducers can be joined, intersected, complemented, composed and determinised under certain conditions. To build a TFFST that models a set of positive obligation policy rules the *union* operation is used to join together individual TFFSTs encoding each of the policy rules separately. However, the union of TFFSTs maintains policy ambiguities and contradictions. To resolve these *determinisation* and *composition* operations are performed [BS04] in a stage following the translation of the policy rules into their TFFST representation. Note that these conflicts arise at a different level from the conflicts between the pre-defined and generated policy rules (Section 5.5.1) which are resolved by the Management side's PM prior to engaging in the policy translation process.

Determinisation. Determinisation operation transforms a TFFST into its deterministic and unambiguous version. In the process it also eliminates static conflicts between individual policy rules. Static policy conflicts are policy conflicts that can be detected at compile-time through static policy analysis — they are obvious from the policy specification, without the need to execute the policy rules.

A TFFST M is said to be *deterministic* if:

- M has a single starting state.

- There are no states $p, q \in Q$ such that $(p, \epsilon, x, q, i) \in \Pi$.

- For every state p and event e there is at most one transition (p, τ_d, x, q, i) such that $\tau_d(e)$ is positive.

Given a deterministic TFFST the complexity of computing the output for a given input string ω is linear in ω and independent of the overall size of the TFFST. The linearity is owing to the fact that to compute the output we need a single path through the TFFST matching the particular input. As a direct consequence, determinism of a TFFST is a guarantee that policy evaluation can be implemented in an efficient way.

Static policy conflicts in μCASPEr arise due to multiple policy rules matching the same containment. These conflicts may arise in two distinct forms: i) where the containments in the conflicting rules are of different "specificness", e.g. `pda/802.11/data_item`, `pda/wireless/data_item` and `pda/comms_channel/*`; and ii) where the containments in the conflicting rules are of the same "specificness". In the former case, the rule with the most specific containment match wins while, in the letter case, the most dominant LoE is the conflict resolution criteria.

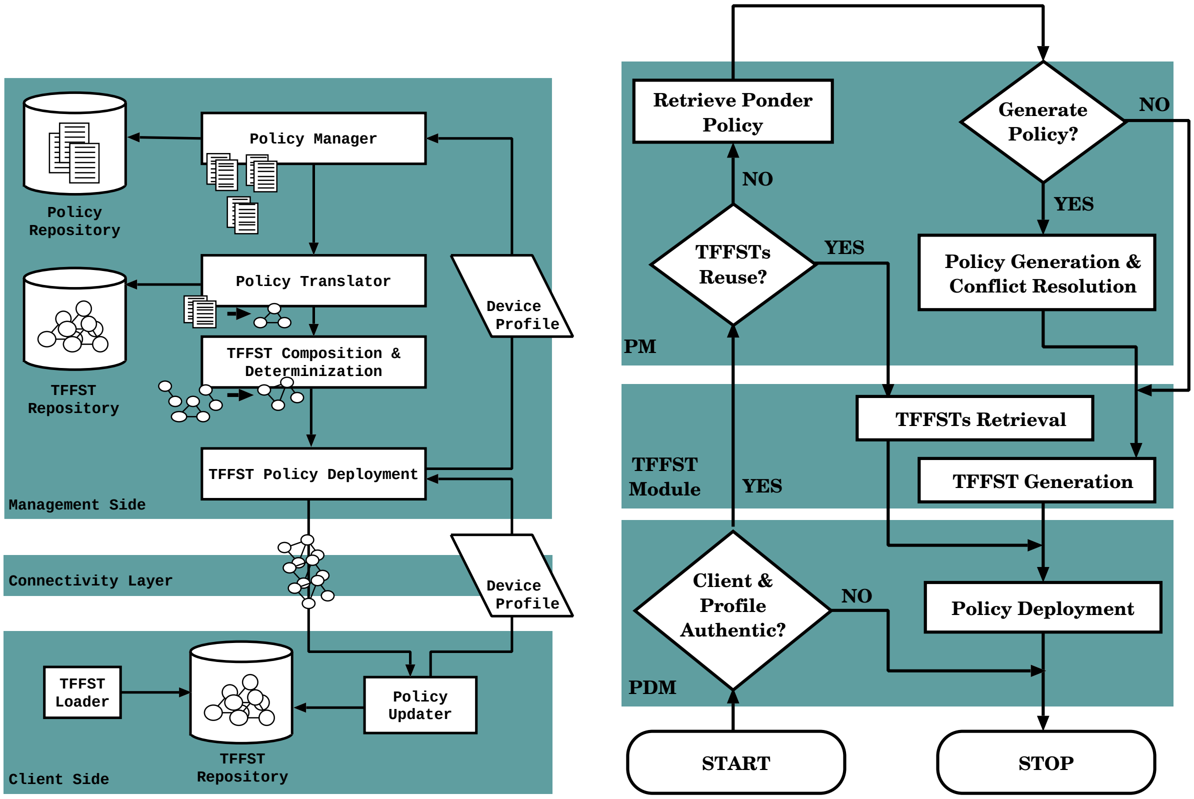
Composition. Composition operation eliminates semantic contradictions in a policy. Semantic policy contradictions are also referred to as *dynamic conflicts* and represent conflicts between "actions" in a policy. Dynamic conflicts, as opposed to the static conflicts, are not obvious from the policy specification. Although they can be predicted, their actual occurrence cannot be detected using the static policy analysis. Dynamic conflicts arise at runtime, due to individual policy rule interactions caused by occurrence of a particular set of contextual states in a specific realm configuration.

Dynamic policy conflicts in μCASPEr arise when multiple policy rules applicable to the same containment specify different actions to be performed on the occurrence of the same LoE. The conflict is a μCASPEr incarnation of the more general protective cover choice problem from Section 4.5. Dynamic policy conflicts also arise when multiple LoEs corresponding to the same threat type are activated simultaneously, due to a single set of contextual states, for the same data item. Dynamic policy conflicts are resolved runtime, as a result of the TF evaluation process, described in Section 5.5.6.

TFFST composition is equivalent to the binary relation composition: $R_1 \circ R_2 = \{(x, z) \mid (x, y) \in R_1, (y, z) \in R_2\}$. The process can be seen as "chaining" TFFSTs so that the output of an input of a transducer in a chain is the output of the preceding container, except for the first TFFST in the sequence.

As shown above, TFFST determinisation and composition operations not only build a transducer that models a policy but also eliminate ambiguities and contradictions within and among the policy rules. In other words, conflict resolution is intrinsic to the policy modelling approach. This is one of the fundamental advantages for leveraging TFFSTs for policy modelling in μCASPEr. Performing the policy conflict detection and resolution process at the Management side has clear advantages with respect to resource constraints at the Client side. This is particularly true for the resolution of the dynamic policy conflicts for which all contextual states that may possibly occur and be modelled by a Client have to be accounted for.

5.5.4 Policy Deployment



(a) Policy deployment scheme.

(b) Policy deployment steps.

Figure 5.10: Policy deployment process.

The policy deployment process subsumes policy translation, conflict resolution, TFFST generation and, optionally, policy generation processes. In other words, it represents a union of the processes involved in the production of a Client specific, TFFST-based, μCASPEr policy. The overall policy deployment process is coordinated by the Policy Deployment Module component of the Management side. The Figure 5.10 illustrates the process. The Figure 5.10(a) represents μCASPEr architecture (Management side) components involved in the process as well as the symbolic representation of their inputs and outputs. Figure 5.10(b) depicts, in an algorithmic fashion, steps of the deployment process as performed by the Management side's components.

To initiate the policy deployment process a Client connects to the Management side's PDM through the Client Connectivity Layer (CCL) and passes it the *Device Profile*. Device profile specifies policy relevant resources and capabilities of the Client. These consist of a mandatory and an optional part. The former specifies:

1. Context, i.e. threat, modelling capabilities of the Client.
2. Container classes and their instances supported by the Client's platform.
3. Threat mitigating operations as well as the Client's components providing them.

We envisage cases in which the Client suffers from resource poverty to the extent at which the runtime computation of TFs to resolve dynamic policy conflicts is unfeasible. Thus, the device profile allows for the specification of TF computation relevant parameters, such as user preferences (Section 4.2) and mitigating operation action impact values, in order to carry out the dynamic policy conflict detection and resolution at the Management side. In this case, a fully static policy is loaded into the Client at the end of the policy deployment process. We also envisage cases in which Clients' platforms are standardised across an organisation and in which the profile may consist only of an identifier — serving solely as a reference for the Management side. Device profiles need to use a standardised, flexible and portable format. The two standards most widely exploited for hardware and software resource description are the CC/PP [w3c04] and the UAProf [oma03].

Once the profile is passed to PM by PDM, PM carries out checks with the TFFST module on whether a policy for the same or a device of same profile has already been generated. If so, and if it has not been made outdated, e.g. by a more recent explicit policy rule specification, it can be immediately retrieved and handed over to the Client where the deployment process ends. Otherwise, PM retrieves, from the Policy Repository, applicable Ponder policy rules and checks their completeness as required for the Client. If incomplete, the simulation process is carried out and the conflicts between pre-defined and generated policies are resolved. The remaining generated policy rules are saved into the separate compartment of the policy repository for possible audit, and reuse, purposes. If policy generation process is not supported, and the existing policy rules are incomplete, either a partial per-Client TFFST policy is produced (for a subset of the supported set of data-item containments) or the input from the policy administrator is solicited. PM then passes the policy rules to the TFFST module for translation and conflict resolution in per data item containment sets. The TFFST policies are optionally saved in the repository for reuse purposes. In the final step, the PDM installs the policies into the Client's Policy Module through the CCL using the Policy Update Interface. The policy is thus deployed and ready to be used at the Client.

5.5.5 Policy Evaluation

Policy evaluation in μCASPEr, as previously stated, follows the *Event-Condition-Action* paradigm. Two main classes of events are the threat occurrences and subsequent LoE activations, as shown in previously presented (Section 5.4) policy Rules 1,2 and 3. Note that during the TFFST generation process the two types of policy rules we distinguished between in Section 5.4 are collapsed. TFFST evaluation is performed by the PEM. At any point in time PEM operates on a set of TFFSTs consisting of a single TFFST per active data item containment. To restate, a data item containment is *active* if it is actually instantiated in a realm. Each active containment may correspond to multiple matching actual containments on the Client's host platform. For the set of active TFFSTs operated on by PEM to reflect the true state of the realm, i.e. its data item containment configuration, each update operation at CMM generates an event that triggers relevant TFFST load/unload to/from the PEM's active policy set by the TFFST Loader.

A fundamental characteristic of the policy evaluation process, supported by the respective TFFST structure, is that at each evaluation of a TFFST, PEM starts at the single starting state and always ends at a termination state of the TFFST. In other words, PEM is fully stateless across TFFST evaluations — it never stops the evaluation in a non-terminating state of a TFFST. The rationale is outlined below.

Policy evaluation process is initiated by the PEM's reception of information exposure threat relevant contextual state description from the Context Modelling Layer. The contextual state description consists of a list of all threat relevant contextual fragments[4]. It is provided to PEM upon occurrence, or ceasing to exist, of a relevant contextual fragment or on explicit request by PEM. The latter is prompted by loading of a TFFST at PEM to adjust the protection level for a newly instantiated data item containment appropriately.

Rather than providing PEM solely with information about single contextual fragments as they occur, CML reports the list of all contextual fragments that hold at the point in time. This is required as all threats essentially map to the same LoE space and to determine the appropriate LoE to activate for each of the threat types the complete contextual state needs to be considered. Otherwise, threat underestimation may occur due to incomplete information about the state of context determining the overall threat model at a point in time.

Alternatively, PEM and TFFSTs would have to support state preservation across TFFST evaluations. In addition, CML events denoting absence of a contextual fragment would have to be supported as well. The main disadvantage of this approach is the explosion in the individual TFFST complexity that it would cause. Thus, this alternative is considered to be beyond feasibility taking into account the resource poverty expected in the μCASPEr target setting. For analysis of TFFST complexity please refer to Section 6.3.

The suggested stateless PEM design keeps the overall μCASPEr complexity at bay. An exception to the design principle of not accounting for the "absence of contextual fragment" events is the *NoExplicitThreat* contextual state referred to throughout the Section 5.4. The exception is made to enable support for the default μCASPEr policy which is conditioned on the absence of all threats.

[4] A contextual state is determined by a set of contextual fragments. An example of a relevant contextual fragment is the *OwnerAway* as used in Rule 1, Section 5.4.2.

Every threat occurrence, in general, may cause LoE triggering for multiple data item containments. This means that PEM needs to evaluate a number of TFFSTs. In general, the overall per-Client policy evaluation process consists of multiple TFFST evaluations. Although, in theory, we can say that the evaluation happens in parallel, PEM actually needs to impose an ordering among the TFFSTs. The ordering may be considered important in cases where the evaluation delay is seen as increasing the likelihood of information exposure. Thus, PEM supports ordering by data item sensitivity level and container class containment structure. For example, we can specify that data items of higher sensitivity are always first to be addressed starting with the ones contained within *Display* containers. The output of both the evaluations of individual TFFSTs and the overall evaluation process is a set of none or more alternative threat mitigation operations that need to be performed to adapt to the experienced threat levels. To discriminate between the alternatives the *TF Computation* process (next section) is leveraged.

For situations in which frequent contextual state changes, and thus the corresponding CML raised events, are envisaged we define the notion of *epoch*. An epoch is a time frame by which the events indicating threat model changes are delayed, at the CML, to reduce the number of policy evaluations through the event grouping. Care has to be taken that the delay does not result in a credible “window-of-opportunity” for information exposure.

5.5.6 Dynamic Policy Conflict Resolution - TF Computation

The output of each transition of a TFFST traversed by the PM consists of a TF algebra expression (Section 5.4). For examples please refer to the Appendix A. The expressions are in the form of strings of symbols linked together by the TF algebra’s logic and tautness operators (Section 5.4). The symbols represent either LoEs to be activated or threat mitigation operations to be performed. The expressions themselves describe the dynamic policy conflicts. To resolve the conflicts, and disambiguate a TFFST policy, the TF expressions are evaluated by TFs. The TF computation results in a single symbol, unambiguous, TFFST output that guides the subsequent policy enforcement.

Probabilistic Prioritising

One of the possible criteria for TF implementation is probabilistic prioritising. In this approach, TFs ponder each conditioning expression, specified by the Ponder `when` clause of the conflicting policy rules, in terms of the probability that they evaluate to true. The lower the probability the *stronger* the condition is considered to be. The *stronger* the condition is the higher priority it is assigned, should it evaluate to true. As an example, consider the following policy rule:

Rule 6:

```
inst oblig PolicyRuleNo006 {
  on LoE.Emanations.3;
  subject s = EnforcementModule.NetworkHandOffManager;
  do s.handOff(*/802.11:nic=A;)
  when s.isAvailable(*/802.11:nic=A;) &&
       s.signalStrength(*/802.11:nic=A;) <
       s.signalStrength(*/bluetooth:nic=F);
}
```

To illustrate the probabilistic prioritising we focus on the condition expression, given in the italic typeface above:

```
...
  when ...
     s.signalStrength(*/802.11:nic=A;) <
     s.signalStrength(*/bluetooth:nic=F);
...
```

The setting of the Rule 6 is sensitive data transmission over a link whose signal permeates the physical container which it originates from, e.g. an office, into an area outside the secure perimeter. We assume that the sensitivity class of the transmitted data implies the data is exposed outside the secure perimeter even if encrypted. To mitigate the threat the connection needs to be handed off to a non-exposed link. In other words, to a communications technology whose signal is, or can be, confined within the secure perimeter. Thus, the main criteria for choosing the target containment for the connection migration, in this case, is the signal strength.

Considering the inherent characteristics of the *802.11* and *Bluetooth* technologies the probability that the former will offer a lower signal strength in realistic conditions is low. Thus, the policy manager must have had an important reason[5] to explicitly specify the condition. On these grounds the condition is considered as very strong and the policy rule is given high priority. Nevertheless, each TF value can be additionally pondered, according to additional criteria, such as user preferences — UPQoS, as presented in Section 4.2, communications link QoS etc.

Inter-TFFST Dynamic Policy Conflicts

The above outlines how the dynamic policy conflicts are resolved during the evaluation of a single TFFST. However, the PM is likely to evaluate multiple TFFSTs as a response to a single contextual state change. Inter-TFFST dynamic policy conflicts arise if the evaluation of multiple TFFSTs results in different threat mitigation operations to be applied to containers in the intersection of the respective data item containments. This may be due

[5] Discounting for the erroneous policy specification.

to different LoEs, associated with overlapping threat types, being experienced by the data items or due to the criteria involved in the alternative mitigation operation discrimination. This problem has been already identified, albeit in a more general context, as the protective cover choice problem (Section 4.5). The OCD algorithm has been presented as the solution. We identify two main alternative approaches to resolving the inter-TFFST dynamic policy conflicts in the context of μCASPEr.

The first approach, suitable for resource poor Clients, is to eliminate any potential conflicts at the policy specification phase. The easiest way to accomplish this is not to allow the policy rules to specify mitigation operations on containers, or portions of a containment tree, which may be in the intersection of multiple data item containments — as determined by a particular container classification and the containment relation definition. This essentially implies providing for mitigation operations only at containers that may not, by definition, contain more than a single data item. For example, considering containments matched by the path expression `*/display/gui_window/data_item`, mitigation operations would only be available for *GUI Window* and *Data Item* containers. This is assuming that the former may contain solely one data item. The second approach, suitable for sufficiently resource capable Clients, requires a TF that implements the OCD algorithm of Section 4.5. The TF is used to process a TF algebra expression derived from the outputs of all TFFSTs evaluated by the PM as a response to a single threat model change. Note that this approach implies effective delay in the individual TFFST policy enforcement until all relevant TFFSTs have been processed.

5.5.7 Policy Enforcement

Policy enforcement in μCASPEr, as in CASPEr, is carried out by the Enforcer components of its Policy Enforcement layer. The policy enforcement process is coordinated by the Enforcement Manager as instructed by the PEM upon the policy evaluation. As the containers themselves represent entities that may originate from, and the management responsibility for which is located within, both the application and the operating system layer components, the corresponding Enforcer components are expected to be provided by the individual component authorities (Section 3.4)themselves. In addition to this, we envisage purpose-specific, dedicated Enforcer modules to be developed as part of the μCASPEr. The various origins of the Enforcer components are depicted in Figure 5.3 — illustrating the manner in which the Policy Enforcement Layer spans the traditional application and system levels. Therefore, the full awareness at both the application and operating system levels is not required only for notifying CMM of updates to the state of the world but also for threat mitigation operation provisioning. The absence of such impacts on the level of granularity at which threat analysis and mitigation may be provided.

Upon a notification by the PEM, the Enforcement Manager invokes the required Enforcers through an asynchronous call-back mechanism. The call-backs are defined in terms of threat mitigation operation *classes*. The classes signify the pre-estimated impact that the corresponding threat mitigation operations have on the respective threats in terms of LoEs. Applications and operating system components register the call-backs (to their own implementation of the mitigation operation classes) with the Enforcement Manager prior to or upon a corresponding container comes under their authority, by creation or otherwise.

Due to the fixed nature of the TFFST policies, the availability of threat mitigation operations at a Client, in terms of classes, has to be known in the per-Client policy formation phase at the Management side. This implies that, unlike in CASPEr, support for novel threat mitigation operation classes cannot be added to a Client without recourse to the Management side. This represents a clear tradeoff between the degree of flexibility and the level of Clients' autonomy provided by μCASPEr versus the associated computational complexity and resource overheads. This is further discussed in Section 6.3.

5.6 CASPEr Data Model: An Outline

In this section we briefly outline a data storage model design in accordance with CASPEr requirements. The design provides for data type specific information manipulation operations (Section 3.5) and the relevant meta data association. Information manipulation actions are performed on per data item basis. They have a generic description (threat mitigation operation class) and data item type specific implementation. To avoid imposing otherwise unnecessary overheads and functionality requirements on the application space and burdening application designers with additional complexity we propose offloading the support for information manipulation actions into the data storage system.

5.6.1 Requirements and Implications

The above suggested approach causes a shift in roles and responsibilities of both the applications and the data repository:

- **Data Repository.** For a data repository to support the information manipulation actions it needs to be able to recognise, at a fine-grained level, application specific data types and respective internal data layouts. By internal data layout we assume the data item substructure of data objects used in the data model, such as files. Understanding of the data layouts also ensures the compatibility of the transformed, i.e. manipulated, data. Means of associating individual data items with sensitivity levels, with respect to a security policy, need to exist. Identification of data items for information manipulation purposes is done on information sensitivity level basis. The data repository, further, has to support a mechanism for provision of data type specific implementations of threat mitigation operation classes and their respective mappings.

- **Applications.** Upon a shift in the threat model, applications should simply be able to retrieve, from the data repository, an "updated" version of any of the threat affected data they are operating on. An event-based system to accomplish this is, however, outside the scope of this thesis. Each application-specific data type should include enough information to enable the correct operation of the data repository with respect to the policy enforcement. For this, application awareness is required. An exception is applications that do not alter data in a way which violates its layout, e.g. data viewers such as browsers, PDF readers etc. These can remain utterly unaware of the underlying data model.

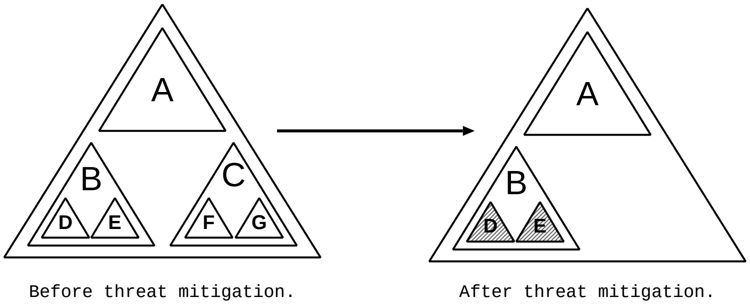


Figure 5.11: A data transformation scenario.

Figure 5.11 illustrates an example information manipulation process at the data repository level. It represents the data item structure of a document prior to and post a set of information manipulation actions, depicted on the left and on the right of the arrow respectively. To address a particular information exposure, as instructed by the EM, the data repository adapts the document by: fully omitting data entity labelled *C*, including data items *F* and *G*, and reducing the information content of data items labelled *D* and *E*.

5.6.2 Data Layout Specification

To address the issue of data layout specification we leverage the capabilities of XML [XML04] data representation. XML allows for applications and standards bodies to produce data layout descriptions which can be interpreted by the data repository in a systematic fashion. XML data representations are seen as matching our requirements for several reasons. Firstly, they facilitate efficient inspection and identification of data content, both automatic and manual, aiding the choice and design of applicable information manipulation actions. Secondly, the XML approach enables easy protective actions validation through XML schema to ensure application specific format compliance of the transformed data. Thirdly, the XML file format being built upon a set of standards implies the ability to reuse data type specific information manipulation actions for all application data that uses the standardised data types. Finally, we have at hand a number of standard XML tools that greatly facilitate generic data access.

5.6.3 Storage Subsystem Architecture

Figure 5.12 shows the component structure of the storage subsystem. It is comprised of a *Generic Interface*, *Data Analyser*, pluggable *Type Wardens* and the *Data Store*. The *Enforcement Module* (EM) CASPEr component initiates the information exposure adaptation process by interacting with the storage subsystem through invoking the generic calls. The generic calls correspond to the threat mitigation operation classes, as mentioned previously. They are internally mapped to data-type specific implementations that obey the pre-defined data structuring rules. These are provided by pluggable modules known as *Type Wardens*. The *Data Analyser* coordinates the execution of information manipulation operations performed by the *Type Wardens*.

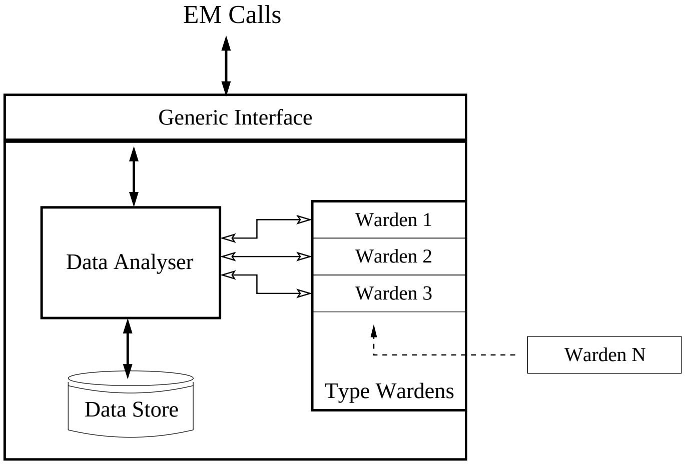


Figure 5.12: Storage enforcement system architecture.

Generic Interface. Generic interface provides for a set of generic calls, representing information manipulation operation classes, as invoked by the PEM. An example of the interface is a family of calls of the form:

```
degrade_levelX(cpath target_data_items)
```

where `target_data_items` is a containment path expression that matches a set of data items stored at the repository. The `levelX` component of a generic function call name denotes the *Information Loss Factor* (Section 4.2) associated with the operation. As the target data items are identified based on their sensitivity, the path expression has to be of the form:

expression`/data_item:sclass=Y[,...];`

Data Analyser. The role of the Data Analyser is coordination of the information manipulation process. Firstly, the Data Analyser identifies target data items and retrieves them from the *Data Store*. This assumes that the data items are actually stored in the local data repository. Data Analyser also supports explicit provision of data items to undergo information manipulation through a separate set of generic calls. Secondly, it identifies and invokes the appropriate operations for each of the data items based on their type. Note that, as the information manipulation service provided by the Data Repository is pre-registered with the EM, it is invoked only for data items of the type supported by the repository through the *Type Wardens*. Finally, it either: i) stores the data back into the *Data Store*; or ii) passes the data to the EM for explicit application update.

Type Wardens. The role of Type Wardens is to provide data-type specific implementations of the generic information manipulation operations. The data type specific actions can be either provided along the type definition by standards' bodies, or application designers, or they may be custom-made by security administrators. The latter is facilitated by the existence of standard XML tools (i.e. DOM, SAX, XPath) that can be used as data manipulation interface to application-specific data types. The ability to provide for type wardens in an "on-demand" fashion significantly facilitates the deployment of the storage system on resource constrained ubiquitous devices.

Data Store. For the purposes of the proposed system the data store is used solely as a persistent data repository.

5.7 Summary

The core of this Chapter has focused on the presentation of μCASPEr, a policy based instantiation of CASPEr. The architecture of μCASPEr was developed as a specialisation of a more general CASPEr architecture to the policy model. Apart from being comprised of dedicated components, we have illustrated how the CASPEr, and thus μCASPEr, architecture spans application and system components of a target platform — hinting at the importance of wider CASPEr awareness. The policy model chosen for μCASPEr was the Finite State Transducer with Tautness Functions and Identities (TFFST) model, based on a variant of Finite State Automata [BS04]. The key advantage of the model is that it is characterised by lightweight policy evaluation while the inherently resource intensive tasks can be off-loaded to computationally capable devices and accomplished *a priori* to the actual policy deployment. Furthermore, tautness functions facilitate autonomy of target platforms. We have shown how Ponder policy specification language, as a representative of Deontic policy languages, can be utilised for high-level expression of obligation and constraint policy rules. The high-level policy rules are subsequently translated into TFFST forms and determinised into their unambiguous forms. We have defined a specific policy structure which, as shall be seen in the next chapter, greatly facilitates the overall policy model flexibility, operational independence of target platforms and also decreases the policy evaluation complexities. We have detailed the operation of μCASPEr in terms of the policy model related processes specifying the roles of individual architectural components. Although policy based systems exhibit, in general, considerable degrees of rigidness, the specific way in which the TFFST model is applied in μCASPEr offers high degrees of flexibility with respect to the constraints imposed by the target deployment platforms themselves. Furthermore, it retains all the fundamental CASPEr principles seen as the contribution of our work. Finally, we have outlined a high-level architecture of CASPEr-aware data model.

Chapter 6

Discussion and Evaluation

6.1 Chapter Overview

In this chapter we argue for the general feasibility of CASPEr and μCASPEr deployment in the target environment and their completeness with respect to information exposure threat analysis, mitigation and the coverage of the envisaged use cases and scenarios. This is accomplished through a discussion of the concepts and mechanisms presented in the previous chapters and their theoretical and empirical, example driven, evaluation, as suitable. The chapter also aims at clarifying the contribution of the thesis through the delineation, by means of a qualitative comparison, of the role CASPEr concepts play in the information security big picture painted by the related work overviewed in Chapter 2.

Section 6.2 focuses on CASPEr as a theoretical framework and the high-level concepts it rests on. Firstly, we discuss the LoE modelling granularity — a quality that influences a number of aspects determining the efficiency and effectiveness of CASPEr, and identify the contributing factors. We proceed to argue for CASPEr feasibility through a discussion and analysis of the resource overheads involved in containment modelling and CASPEr operation, supported by a realistic example. Finally, we demonstrate how communications channels, naturally spanning multiple containers, fit the single containment paradigm and describe the role of modelling input devices as containers. In Section 6.3 we evaluate μCASPEr with respect to the TFFST policy model overheads incurred at the Client side. Per-Client TFFST policy size as well as its evaluation overheads are analysed in detail, both theoretically and through an example. The policy model scalability is also discussed. In Section 6.4 we generalise the OCD algorithm by showing how it can be extended to account for non-composability of information security mechanisms. We then present the algorithm's termination and complexity analysis. The final section of the chapter (Section 6.5) focuses on the delineation of CASPEr from the related work and crystallisation of its place within the information security big picture. We present a qualitative analysis of CASPEr and its qualitative comparison to the related work of Chapter 2 based on 19 criteria. Subsequently, we attempt at visualisation of CASPEr within the big picture through detailed, conceptual, identification of its distinguishing characteristics from the point of view of information flow control, dissemination control and access control.

We conclude that CASPEr (and μCASPEr) deployment in the envisaged target settings is feasible and that it plays a unique role, with respect to several aspects, in information security.

6.2 CASPEr Discussion

6.2.1 A Comment on LoE Modelling Granularity

The granularity of a Levels of Exposure model has a two-fold impact on CASPEr:

- The coarser grained the LoE model the less accuracy in matching a threat mitigation operation to an experienced exposure degree. Consequently, the less ability to balance the information security vs. utility seesaw.
- The finer grained the LoE model the higher the frequency of the adaptation process — encompassing protective cover discrimination and its enforcement, and thus higher the related resource overheads[1].

While the former depicts the impact LoE model granularity has on CASPEr effectiveness the latter hints at the related operational resource overheads.

LoE model granularity, as shown in Chapter 4, depends on context capturing and modelling capabilities, context-threat correlation certainty and confidence, and availability of matching threat mitigation operations provided by a CASPEr deployment platform. Research in context awareness has, so far, resulted in availability of high precision, high confidence contextual information in instrumented environments. This is especially so with respect to location sensing [HSK04].

What we see as the fundamental factor influencing LoE model granularity is the estimation how likely a threat is to materialise in a context and also to what extent individual threat mitigation operations impact on the likelihood. At the very core of the problem lies the necessity of profiling, with a certain confidence, human "attackers" as potentially present in the encountered contexts. Fine grained attacker profiling is especially difficult for its subjective side, involving a number of psychological and physiological factors, as well as the necessity to estimate the motivation, skills, knowledge and resources of the attacker. Further blow to the LoE modelling granularity is delivered by the fact that the defence has to be aligned with the most severe threat that may arise in a context. For context-threat correlation, thus, we have little alternative than to rely on instruments such as expert opinion, historical evidence or, due to the sensitive nature of information exposure, highly constrained experiments, such as in [TC03].

[1] Note that this is true for environments with dynamically changing context.

Consequently, we expect the LoE modelling granularity to be dominated by the above, threat estimation, issues rather than context capturing and modelling capabilities or the availability of threat mitigation operations. Although the decrease in the LoE modelling granularity has a direct negative effect on the efforts to maximise information utility, even in cases where a LoE model collapses to binary (exposed/not-exposed) a number of benefits provided by CASPEr remain unaffected, such as context-adaptive, pro-active, temporally and spatially continuous information exposure protection.

6.2.2 Containment Modelling Size Overheads

A question that naturally arises in the context of the containment-based approach to modelling the world is the potential size overhead of the model representation. An upper bound on the size of a realm representation can be expressed in terms of the maximum representational size of a single container and the maximum number of containers in a single realm.

Nominally, every container has to encapsulate at least a set of attributes required for threat estimation, as determined by container classification, and methods, or references to methods, implementing relevant transparency functions and threat mitigation operations. Representational size of such a container is highly implementation specific. Different programming paradigms, target CASPEr deployment software and hardware platforms and implementation design decisions will yield different overall container sizes.

To provide a “feel” for a realm size in terms of the number of containers we consider software and hardware platform configuration of a typical general purpose laptop computer in conjunction with the moderately detailed container classification presented previously in the Figure 3.4, Section 3.3.3. Figure 6.1 shows a graph depicting example definition of the containable relationship for the classification. Observe that we abstract away data type specific classes and consider individual cryptographic algorithms as of a same relevant strength.

The maximum possible number of containers in a realm corresponds to a containment tree where every container from the classification is instantiated, for each combination of its attribute values and for every container it may be contained within, according to the containable relationship definition. For the example in Figure 6.1, this gives the maximum containment tree depth of 4 levels and the maximum number of containers to be represented as 73. Note that any externally attachable devices, e.g. displays, storage or audio, are not accounted for in the example.

For example, with respect to *Storage Device* container class, in a single containment tree we can have at most two instances of the class corresponding to removable and non-removable storage devices respectively. Each of the instances can further contain an instance of *Data Item* container for each of the three sensitivity classes in both unencrypted and encrypted form. According to this, every instance of a *Storage Device* accounts for at most 8 containers.

In short, the above back-of-the-envelope model size estimation gives us confidence that memory capacity overheads incurred by a containment model representation would not be

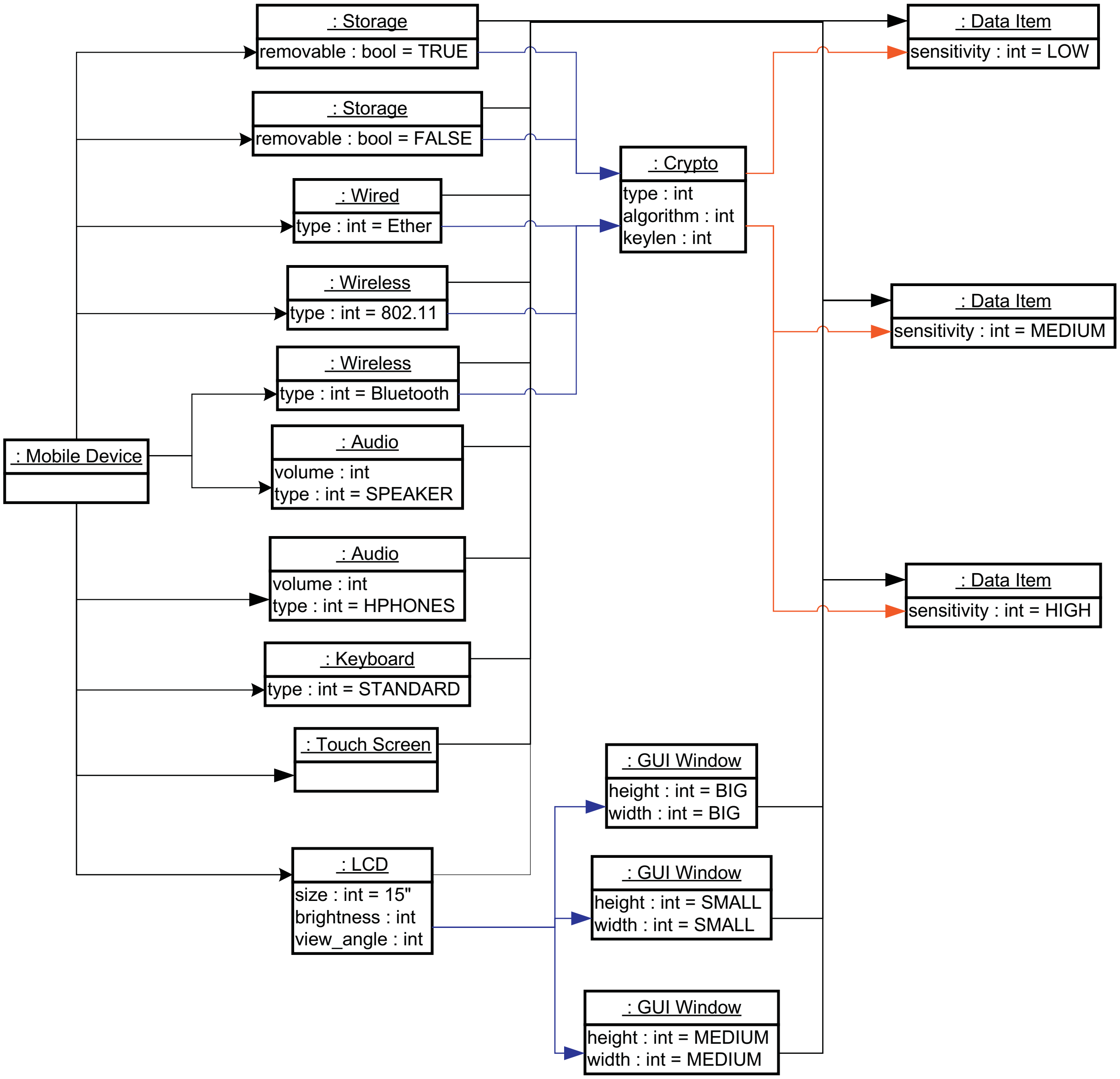


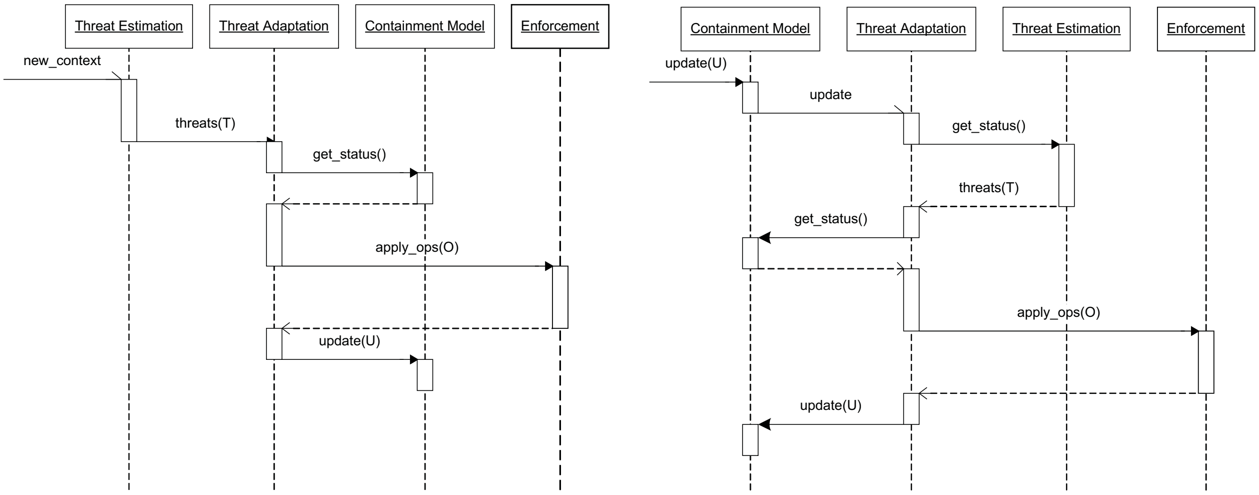
Figure 6.1: Example containable relationship specification.

substantial irrespective of the actual implementation approach and at any realistic modelling granularity. The favourable result is owing to the fact that every container in a containment model realm representation corresponds to all instantiated real world entities of the respective class that are in the same state.

6.2.3 Computational Overheads

CASPEr processes that we can identify as contributing to CASPEr computational overheads, at the architectural level of abstraction, are:

- Context sensing and modelling.
- Contextual state - information exposure threat correlation.
- Threat propagation — LoE establishment.
- Protective cover discrimination — OCD algorithm.
- Threat mitigation operations execution.
- Containment model update.



(a) Threat model change scenario.

(b) Model update scenario.

Figure 6.2: Main CASPEr operation scenarios.

Figures 6.2(a) and 6.2(b) show UML sequence diagrams representing two main CASPEr operation scenarios at a high level of abstraction. The former depicts a sequence of actions following a shift in threat model. The latter scenario occurs upon execution of a containment model update operation. The entities in the sequence diagrams correspond to the layers of the CASPEr architecture as described, together with their roles, in Chapter 5.

Context and Threat Modelling

Resource overheads involved in context sensing and modelling, as well as in the threat correlation processes, are highly implementation and platform specific and their assessment is beyond the scope of CASPEr. We expect them to be tailored to the capabilities of the target platform. Observe, however, that the expressiveness of a context model directly influences information exposure threat correlation granularity.

Threat Analysis and Mitigation

Threat propagation involves computation of dedicated transparency functions at each container as encountered during threat propagation down a containment tree. In the worst case, the number of individual transparency function invocations is equal to the number of containers in a containment tree affected by a threat — should the threat propagate from the tree root to all the leaf nodes. We expect transparency functions to be implemented as low-cost, mainly arithmetic, operations.

Threat mitigation operations are implementation and enforcement platform specific. Their complexity may range from negligible, such as display blanking or file erasure, to considerable, such as network data migration or resource intensive cryptographic operations. We expect availability of mitigation operations to be tailored to target platform capabilities.

Containment Model Update

Containment model update operations (Section 3.4.6) consist of simple alterations to the corresponding data structures. The order of their execution can be expressed as O(1) in terms of comparisons required to locate individual target containers and containments. This is owing to the utilisation of the containment path expressions. After the target containment has been located, the realm update consists of low-cost data structure (n-ary tree) updates [CSRL01]. Each of the update operations may cause, as a side-effect, trigger (Section 3.4.6) propagation up the containment tree and the execution of the relevant trigger-matching operations at each of the encountered nodes. The upper bound on the number of thus triggered operations is in the order of containment model tree depth.

The OCD Algorithm

The OCD algorithm may potentially be executed following every containment update operation and every contextual state change event, i.e. it has a role in both of the above scenarios. The complexity of the algorithm is pseudo-polynomial as shown in Section 4.5.6 and revisited in Section 6.4 below — and thus not seen as a significant resource overhead contributing factor.

Apart from resource overheads caused by any of the individual processes we are also interested in potential frequency of their executions. In other words, in the frequency at which scenarios in the Figures 6.2(a) and 6.2(b) may occur.

The occurrence frequency of the scenario in the Figure 6.2(a) is given by the threat model change frequency. This depends on the context modelling and information exposure threat correlation granularity as well as context dynamism. The former is reflected in the LoE modelling granularity. For the reasons discussed in Section 6.2.1 we do not see the scenario as of prohibitively high frequency. However, where threat model changes are frequent, the notion of *epoch* can be leveraged to reduce the overall number of executions of the respective processes through introducing a delay in the corresponding event reporting, as specified for μCASPEr in Section 5.5.5.

The frequency of containment update operations, triggering scenario in the Figure 6.2(b), depends on the level of detail and the level of abstraction at which a realm is modelled. Most of the update operations as presented throughout this thesis are direct or indirect results of explicit user actions — hinting at their low frequency from the computational point of view. Otherwise, the notion of *epoch* can, again, be leveraged to keep resource consumption at bay.

In summary, we do not expect any aspect of the CASPEr operation to impose prohibitive computational overheads. This is partly due to the CASPEr design and partly due to the ability to tailor the individual mechanisms supporting the CASPEr operation to the capabilities of the target deployment platforms.

6.2.4 Modelling Communications Channels

The tree structure of the containment based model of the world presented in Section 3.4 rests on the assumption that containers are always nested fully within each other. And so do the presented approaches to information exposure threat mitigation. We call this *single*, as opposed to *multiple* containment. Single containment does not apply straightforwardly to containers of *Communications Channel* class. The very purpose of a communications channel is to span physical containers — at least the ones representing devices interfacing to it. Thus, in the general case, they cannot be assumed to be confined within physical boundaries of a container nor even a realm — seemingly breaking the single containment concept. The more so as we already model the *Emanation* threat type based on this characteristic of communications channels.

On one hand, representing a communications channel within a single containment is correct from the point of the role and responsibility of the realm's authority — such as threat mitigation for the data items within the channel. On the other hand, as a data item propagates between source and destination realms, along a communications channel it is contained within, it is likely to be exposed to information exposure threats different to the threats experienced by the local, source, realm. This means that the standard threat propagation process cannot be utilised to establish threat model for data items contained within a channel. However, the local authority still has to fully account for threat mitigation for data items contained, i.e. to be transmitted, within a communications channel.

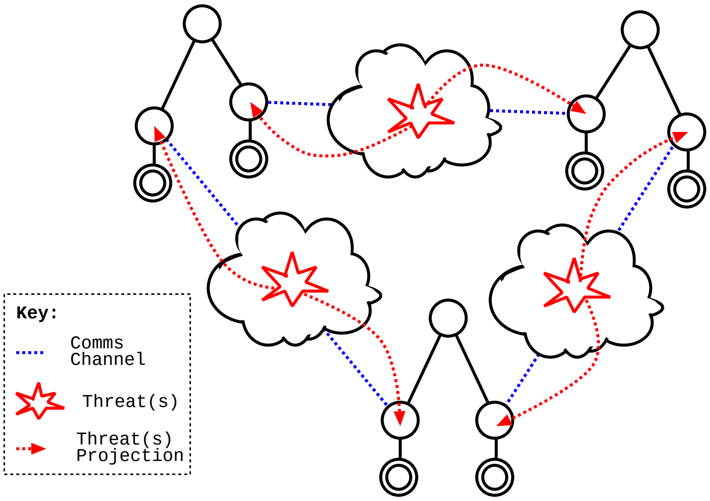


Figure 6.3: Communications channels modelling example.

Figure 6.3 depicts three simplistic realms interconnected by communications channels. The "clouds" along the communications channels represent environments they span that are characterised by different information exposure threat models.

Introducing the concept of a container (Section 3.3), we stated that context can be either internal or external to a container. To be able to account for all information exposure threats encountered by data item on its propagation across a communications channel, and still retain the single containment paradigm, we define the internal context of containers of class *Communications Channel* to be the union of all threat relevant contextual states as present along the channel. The arched arrows in the Figure 6.3 depict this. In this manner, the local authority is able to protect outgoing data adequately in the standard fashion.

We use the same approach to provide protection for data items contained within hot-pluggable devices that may leave a realm without prior notice or are immutable. Examples are USB sticks, memory cards, CDs and DVDs etc. We model these by asserting a constantly high-level of information exposure threats for their contents originating in their interior.

6.2.5 Modelling Input Devices

In the example container taxonomic hierarchy of Section 3.3 we have specified *Input Device* as one of the container classes. Referring back to the container classification criteria two questions naturally pose themselves:

- How do input devices, such as keyboards, touch-pads etc. represent data item containers?

- How can a piece of information be exposed by being contained “within” an input device?

To answer the first question, we say that a data item representing a piece of information is being contained within an input device during the process of data input. As soon as the data representing a single data item, e.g. a Personal Identification Number (PIN) or a password, has been input it is no longer considered to be contained within the input device. Thus, input devices, akin communications channels, represent transient data item containers. To actually be able to model data item containment within an input device container applications initiating user input need to acknowledge the process by updating the containment model accordingly — through requesting the creation of a data item of appropriate sensitivity within the appropriate input device container. This implies application awareness not only of CASPEr but also of the information security classification, implying the ability to parse the relevant meta-data.

To address the second question above we focus on *Optical* and *Acoustic* information exposure threat types. A recent work by Zhuang et al. [ZZT05] has shown that by analysing solely the sounds of keyboard keystrokes it is possible to recover, with staggeringly high confidence, the data being input. Unlike some of the previous research, Zhuang et al. obtained the results in non-constrained environments, with no prior knowledge on the “typist” or any training data, for random as well as non-random input character sequences and even for “silent” keyboards. They also suggest that similar results may be obtained from outside the room by using parabolic microphones.

Depending on their physical size and layout and the method of data input supported, input devices may facilitate variable degree of input data recovery through analysis of observed, or recorded, data input process. In other words, through *Optical* information exposure threat type. In a recent experiment [Zal05], Zalewski has been able to recover short sequences of keystrokes using thermal imaging — making safes and Automatic Teller Machines (ATMs) particularly vulnerable. Although we were unable to find any published study on the direct observation of data input sequences we believe that results of an experiment like [TC03] would clearly point at existence of the threat and its severity.

The inherent transparency of input devices for the threats of *Optical* information exposure may be accounted for by careful design of software data input interfaces. For example, an interface that randomises physical key to input symbol mappings may be a venue to explore for highly sensitive data. A commercial vendor, Pointsec[2], offers a related product called *PicturePIN.* To be able to model the impact of data input interfaces on information exposure threats, we introduced the *Input Interface* virtual container class in Figure 3.4 of Section 3.3.

In the example μCASPEr policy in Appendix A we account for the information exposure threats associated with input devices. Through modelling input devices and controlling input

[2] www.pointsec.com

of sensitive data, CASPEr allows for limited information integrity protection. For example, we could effectively protect the integrity of a mobile phone's PIN by specifying that the data of the respective sensitivity is under threat if input in the device owner's absence.

6.3 μCASPEr Evaluation

To transform the generic CASPEr architecture, where all processes are executed on a single host, to a "split" architecture as proposed for the μCASPEr represents a clear trade-off between the level of flexibility and autonomy in the information exposure threat adaptation process at the Client side and the complexities and resource requirements of supporting and enabling the adaptation process itself.

Two aspects inherent in the TFFST model itself [BS04] pose a possible disadvantage with respect to the feasibility of the policy deployment process at the resource deprived Client side. Those are the high complexity of the algorithms involved in TFFST generation and the size of resulting TFFST. Apart from having possibly prohibitive storage requirements, the latter also impacts on policy evaluation computational overheads.

In order to deploy a policy based solution in an environment likely to exhibit severe levels of resource poverty it is essential to shift processes with high overheads away from the constrained target devices. The fundamental role in accomplishing this was played by the choice of the TFFST model for the policy representation in μCASPEr. The most resource and computationally intensive tasks related to the TFFST policy modelling turn out to be, at the same time, not the tasks required to be performed runtime for policy evaluation or enforcement at the resource poor Client side. Consequently, they lend themselves to straightforward offloading to resource unconstrained Management side. There, they are accomplished *a priori* to the actual policy deployment at the Client side. These tasks include, most notably, the process of generating TFFST policy model representation and the related conflict resolution. The detailed overview of the processes taking place at the Management side was given in Chapter 5.

In this section we focus on evaluating μCASPEr through analysing the complexities and resource overheads incurred by the TFFST policy model related processes carried out at the Client side. At this stage of μCASPEr development we assume the Management side to possess ample resources to carry out the tasks assigned to it. Furthermore, it is the processes taking place at the Client side that are critical for adequate and timely information exposure mitigation — i.e. for the μCASPEr efficiency and effectiveness.

6.3.1 μCASPEr Client Side Overheads: An Overview

Considering their impact on μCASPEr effectiveness and efficiency, we distinguish between tasks that are on the threat mitigation critical path and all other, supporting tasks. The former, namely, are policy evaluation, performed by the PEM, and policy enforcement, coordinated by the EM and accomplished by the Policy Enforcers 5.3. The latter are context modelling, including the relevant event generation and delivery, containment model management, and management of the active TFFSTs set.

In the context of μCASPEr, unlike for CASPEr, we do not see the containment model representation as a considerable factor. We expect it to be of a rudimentary form using only containment path-expression like strings with a sole function of supporting the TFFST Loader. The resource overheads of maintaining an active policy set, consisting of simple TFFST load/unload operations, are seen as negligible and dominated by the storage and memory space requirements of the respective transducers. As in the case of CASPEr, the context modelling and related overheads are outside the current scope of our work.

One of the inherent "features" of TFFSTs, trading off against their low cost of evaluation, is their potentially prohibitive space complexity, i.e. their physical size. This is widely recognised in the relevant areas of research [BS04]. We leverage a number of heuristics in μCASPEr, drawing back to per-Client policy structure design (Section 5.4), to keep the internal TFFST policy model complexity and size at bay. In the following two sections we focus in detail on the space complexity of the μCASPEr TFFST policy model and related computational overheads, processing time in particular, incurred by the policy evaluation and enforcement processes. We also point out how certain μCASPEr design decisions enable lowering the complexities.

6.3.2 Policy Size as a Inhibiting Factor

In Natural Language Processing and other fields where FSTs are leveraged for making time critical decisions they are frequently characterised as "lightweight". This refers to the favourable evaluation processing time overheads rather than to their physical size. For deployment in μCASPEr the potential physical size of a per-Client TFFST policy represents an equally inhibiting factor as does the policy evaluation complexity for the efficiency and effectiveness of information exposure threat mitigation.

We firstly discuss the size of each individual, per data item containment (Section 5.4.5), TFFST and then, based on the findings, proceed to comment on the potential size of the overall, per-Client, TFFST policy.

Comments on TFFST size

A standard way to express size of a FST, and thus TFFST, is in terms of the number of transitions. This is as the transitions embed all the necessary information for TFFST evaluation. The states usually represent only nominal entities, associated with no functionality.

The main factors contributing to a TFFST size are:

1. The number of symbols in the alphabet used for expressing TFFST input.

2. Allowed degree of repetition of the symbols in the input string.

3. Significance of the order in which the symbols appear in the input string.

The alphabet symbols in the FST terminology correspond to distinct events denoting information exposure threat relevant contextual states in μCASPEr. Any repetition of symbols in the input string does not, thus, carry any more information about information exposure threat presence than a single symbol occurrence does. Consequently, we discount for the second factor above and define that no input symbol repetitions are allowed, as a first step to constraining the potential TFFST size.

In cases where the ordering of the symbols in TFFST input string is seen as important the maximum number of possible strings can be expressed as:

$$\sum_{i=1}^{m} P_i^m$$

where m is the number of symbols in the alphabet and P denotes permutations without repetitions. As without any symbol repetitions the maximum input string length is m, the upper bound on the number of transitions in a TFFST is given by:

$$m \times \sum_{i=1}^{m} P_i^m$$

Should the symbol repetitions be accounted for, both the m and P would escalate resulting in a significant, effectively unbounded, increase in the potential TFFST size.

However, in μCASPEr the order in which threat estimation relevant contextual states arise does not in any way influence the mitigation process — ultimately, all of the threats need to be mitigated. In other words, the ordering of input symbols is seen as unimportant. Consequently, the maximum number of strings generated from an alphabet of m symbols is given by:

$$\sum_{i=1}^{m} C_i^m$$

where C denotes combinations without repetitions. Compared to the permutations, each iteration of the above summation lowers the maximum number of possible input strings by a factor of $i!$. This is by the definition of permutations ($P_i^m = \frac{m!}{(m-i)!}$) and combinations ($C_i^m = \frac{m!}{(m-i)! \times i!)}$). In analogy to the previous case, the upper bound on the number of transitions in a TFFST is now given by:

$$m \times \sum_{i=1}^{m} C_i^m$$

The fact that the ordering of the symbols in the input string is not important means that we can impose an explicit ordering, prior to the TFFST evaluation stage, with no consequences for the efficiency and effectiveness of μCASPEr threat mitigation process as such. The ordering results in further tightening of the upper bound on the TFFST size, as given by:

$$\sum_{i=1}^{m} C_i^m \times i$$

As a proof-of-concept we implemented TFFSTs as simple Java data structures resulting in each of the individual transitions accounting for approximately 40 bytes. In Appendix A we specify a moderately simple, but effective, example CASPEr policy and present a selection of TFFSTs. Figure A.11 shows the most complex of the policy TFFSTs — comprised 87 transitions among 31 states. Judging by the above size of the individual transitions, the size of the TFFST can be estimated at less than 3.5 kilobytes. Note that the other TFFSTs of the example policy are significantly smaller. It should also be taken into account that the exact size of a transition depends on the programming language, implementation paradigm and data structures used for the actual TFFST realisation. Thus, the numbers stated here should interpreted as a realistic approximation, rather than an exact estimator or a bound.

Comments on TFFST per-Client Policy Model Size

A per-Client TFFST policy model is comprised of multiple TFFSTs. Each of the TFFSTs corresponds to a single data item containment as supported by Client's software and hardware platforms (Section 5.4.5). Similarly, the maximum number of input events, i.e. the symbols in the alphabet as accepted by each of the TFFSTs, depends on Client's context modelling and threat estimation capabilities.

To demonstrate that it is highly unlikely that the overall TFFST policy size for a Client will become the inhibiting factor for the policy deployment we provide for an example estimation. The estimation is based on the container classification in the Figure 3.4, Section 3.3.3, the definition of the containable relationship from the Figure 6.1 and the approximation of individual TFFST size presented above. As derived from the Figure 6.1, the total number of data item containments, in this case, is 51. For illustration, there are 6 data item containments involving each of the *Storage Device* containers — a single one per *Data Item* container of different sensitivity, in encrypted and unencrypted form alike. The size of the overall TFFST policy for the example can be estimated at, approximately, $3.5 \times 51 = 178.5$ kilobytes. From another point of view, this means that a typical mobile phone, as sold on the market nowadays, with 32MB of memory could easily store approximately 90 distinct TFFSTs within only a 1% fraction of its storage capacity. This demonstrates that, despite the variability in a number of contributing factors, the size of the per-Client μCASPEr policy is unlikely to inhibit μCASPEr deployment in its target setting.

Scalability-wise, adding support for a single container increases the overall TFFST policy size by the size of a set of TFFSTs corresponding to the new data item containments thus introduced. The number of additional TFFSTs is equivalent to the number of leaf nodes

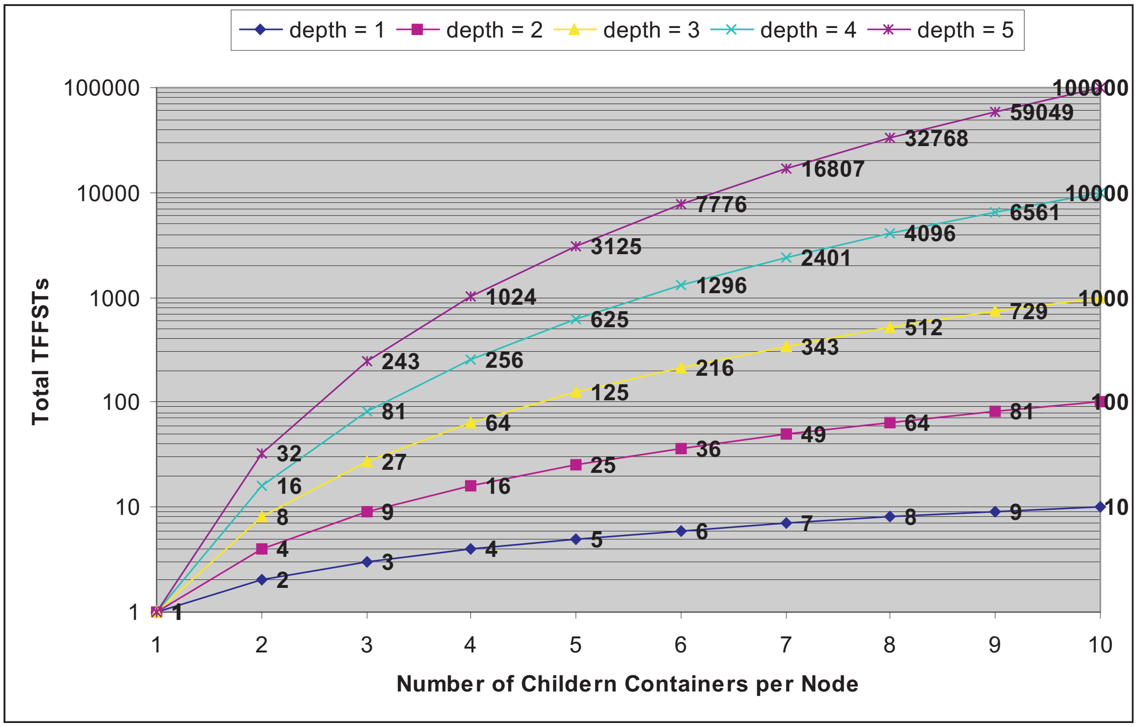


Figure 6.4: Per-Client TFFST policy scalability.

below all instances of the newly introduced container in the containment tree of the maximum size for the particular container classification and containable relationship definition. An exception are leaf nodes themselves, whose contribution is one TFFST per instance. Note that the number of data item containments in a realm is strictly less than the number of containers as each containment is comprised of one or more containers. The graph in Figure 6.4 shows how, in the worst case, the number of TFFSTs in a per-Client policy grows with the increase in the number of distinct containers to be represented at each level in a containment tree, for tree depths ranging from 1 to 5 (with the root node at the depth 0). Each of the five plots in the graph corresponds to a single tree depth. The X axis represents the number of first-hop children for each of the nodes in the tree, except for the leaf nodes, rather than the absolute number of nodes in the tree. This implies the general uniformity of containment trees, as was the case in Section 4.5.2, which is highly unlikely to arise in practise, both in terms of the nodes' out-degree and the balance of the tree. Thus the worst case characterisation of the plots.

In summary, considering that the trends in memory capacity of mobile devices exceed tens, hundreds and even thousands of megabytes, we do not expect the potential TFFST policy size to be an obstacle for μCASPEr applicability to the target setting.

6.3.3 Policy Evaluation Complexities

Computational complexity, and thus the processing delay, incurred by a FST evaluation does not depend directly on transducer's overall size but on the length of the input. This is owing to the fact that in every deterministic transducer there is, at most, a single path that matches a particular input. The transducer is processed by following the single path and evaluating the respective TFs along the way (Section 5.4.3). For the correct path to be followed, a transition matching the relevant input symbol has to be selected, through label comparison, at every state of a DT encountered in the evaluation. The number of comparisons to be made impacts on the processing delay. Therefore, the out-degree of transducer's states is seen as the most important factor in characterising the computational complexity of a TFFST evaluation.

An upper bound on the evaluation order for a TFFST, given a set of distinct events E as the input, is given by:

$$|E| \times Max_i(O_i) \times Max_t(T_t)$$

where $|E|$ is the number of events in the input, O_i is out-degree of a state i and T_t is time to evaluate a TF associated with transition t. $Max_i(O_i)$ represents the maximum out-degree among the states in a TFFST while $Max_i(O_i)$ is the evaluation time for the most complex TF. In practise, TFs are assumed to be chosen to match Client's capabilities. The overall, per-Client, policy evaluation time is linear in the number of TFFSTs to be evaluated.

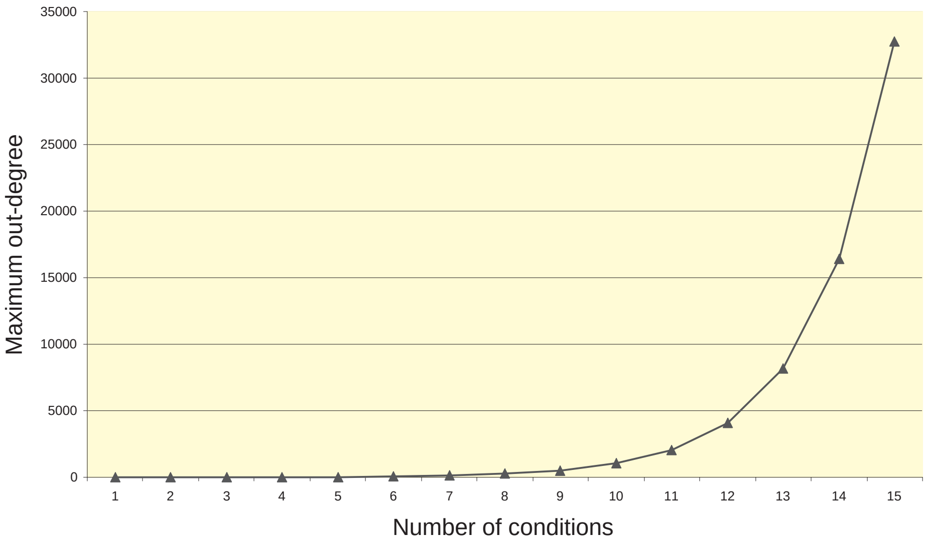


Figure 6.5: Number of conditions vs. TFFST out-degree.

Out-degree of a TFFST[3] is directly affected by the number of conditions in a policy

[3] Out-degree of TFFST is equal to the maximum out-degree among its states.

[BS04]. This is as after applying the determinisation algorithm every combination of mutually exclusive conditions has to be accounted for. Policy conditions in Ponder are given by the `when` clauses (for examples please refer to policy rules in Section 5.4). Figure 6.5 shows how the number of conditions in a policy affects the TFFST out-degree. More formally, the upper-bound on out-degree of a single TFFST state is given by:

$$O\left(\sum_{i=1}^{n} C_i^n\right)$$

where n is the number of conditions fulfilled by a single event. The lower bound is linear in the number of conditions:

$$O\left(n\right)$$

In μCASPEr, the number of policy conditions was significantly reduced by embracing the per data item containment, rather than a per-Client, TFFST policy modelling approach. Other than any static conflicts, we expect TFFSTs to be inherently highly deterministic. Consequently, out-degrees of TFFSTs representing μCASPEr policies are brought close to $O\left(n\right)$ in the average case.

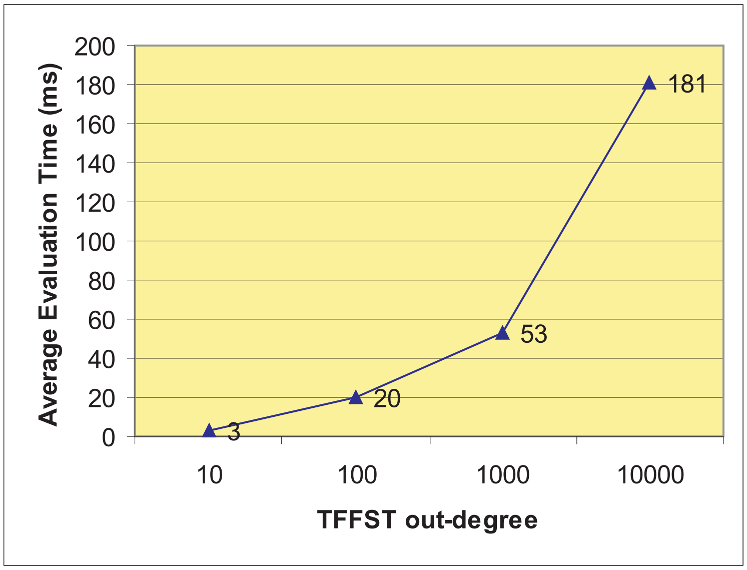


Figure 6.6: Average TFFFST evaluation time.

The graph in Figure 6.6 plots average evaluation time for four TFFSTs with out-degree values 10, 100, 1000 and 10000 respectively. Figure 6.7 depicts the general form of the TFFSTs that were evaluated. The number of states implies the input string of 10 symbols — equivalent to 10 co-occurring information exposure threats of different types. Transitions were labelled sequentially, from 0 to $n-1$ with n being the out-degree, as shown in the

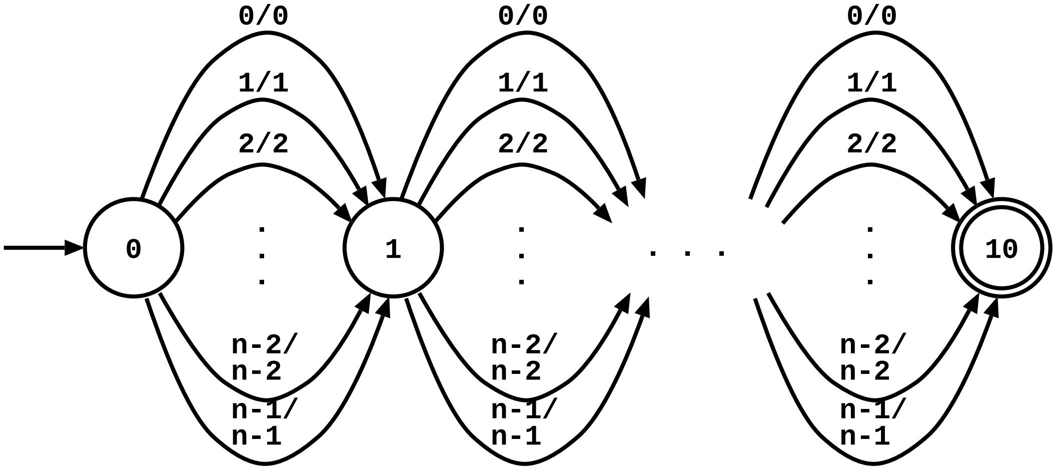


Figure 6.7: General form of the evaluated TFFSTs.

figure. To obtain the average evaluation time the input strings were always formed to target the middle transition among the states. Each experiment was repeated 100 times and the mean was calculated. Standard deviation was below the timing granularity, i.e. equal to 0, except for the out-degrees of 1000 and 10000 where the results deviated by 1 and 2 ms respectively. The evaluation platform was a Toshiba Tecra 9100 laptop with Intel Mobile Pentium 4 clocked at 1.7GHz and 512MB RAM, under an average load. To put the obtained results into perspective of a real world policy, consider that the most complex TFFST of the example policy presented in Appendix A, shown in the Figure A.11, has the maximum out-degree of only 6 (for the state 0).

As for the policy enforcement, the resource overheads incurred by specific threat mitigation operations are application and Client's platform specific and we have no grounds to assess them in this thesis. The operations may incur next to no cost at all, e.g. screen blanking, but may also exhibit significant costs, e.g. public key cryptography related operations (assessed in [GG01] [AVTO03]). We envisage availability of threat mitigation operations at the target platforms to be tailored to their resource capabilities.

6.3.4 Summary

Two most important factors affecting μCASPEr applicability in the target environment are identified as the number of transitions in a TFFST and its out-degree. The μCASPEr design goal was keep the two as low as possible. This was largely accomplished by opting for the per data item containment, rather than for the per-Client, TFFST policy model approach. Policy definition on per data item containment basis allowed for more specific policy rules, substantially lowering the number of conditions and thus impacting on both of the resource overhead critical TFFST characteristics. For devices with split storage space and operating memory, the approach allows for optimisation of the memory capacity used by the active policy set. On-demand loading and unloading TFFSTs from the active TFFST policy set can be seen as Client hot-programming.

With respect to scalability, we are not concerned about the number of ubiquitous computing devices supporting μCASPEr as they act fully independently. Focusing on a single device, the model may scale by adding support for new containers and by enhancing ability to recognise new events. The impact of the former is in the overall policy size, as determined by respective container classification and containable relationship definition. The latter causes a potential increase in TFFST input strings length and thus contributes to the TFFST evaluation time. Based on the presented evaluation results we are confident that scalability in any of the directions will not represent an inhibiting factor for μCASPEr applicability.

In conclusion, we consider μCASPEr deployment as feasible on highly resource constrained ubiquitous computing devices.

6.4 OCD Algorithm Generalisation and Evaluation

6.4.1 Constraints on Threat Mitigation Operation Composition

A well known problem in the area of theoretical computer security research is composition of security mechanisms [And01]. The essence of the problem is that if we compose two or more security mechanisms the *strength*[4] of the obtained composition is not in general equal to the sum of strengths of the individual security mechanisms, but less. Furthermore, the composition process itself may lead to sensitive information leakage. Strong evidence of this has been provided for ciphers, security protocols and secure components composition [And01].

The information exposure threat mitigation approach proposed for CASPEr relies heavily on composition, i.e. “stacking”, of relevant security mechanisms represented as the transparency of individual containers. This is due to the structure of the containment-based model of the world and the way in which threat propagation is modelled. Thus, the concern is whether it is safe to say, in general, that transparency of a containment is equal to the “sum” of individual transparencies of the containers that comprise it. So far, in the exposition of the OCD algorithm (Section 4.5) we have assumed so.

Potential constraints on threat mitigation operations composition can be modelled in two manners:

1. By embedding any constraints in the container classification and containable relationship specification.

[4] The extent to which the mechanism is able to withstand the specific attack in was designed to protect against.

2. By explicitly defining non-composable mitigation operation groups globally or on per container class basis.

The former approach effectively ensures that in all well-structured containment trees no offending mitigation operations can ever be composed, through container stacking. Effectively, this means isolating non-composable transparency characteristics into separate containers and ensuring they may never contain each other in an offending manner. An example would be defining the containable relationship so that a container of class *Crypto* may never, directly or indirectly, contain another container of class *Crypto*. The advantage of the approach is that no explicit checks, apart from ensuring that a realm is well structured, need to be carried out at the run-time. The main problem with this approach, however, is that it may not be possible, in practise, to classify containers in such manner while abiding by the container classification criteria and retaining container semantics (Section 3.3).

The latter approach implies the resolution of the non-composability problem at the threat mitigation stage — run-time at the Client side. More precisely, in the OCD algorithm execution phase. To accomplish this, we assign each group of non-composable operations a single colour in what we call threat mitigation operation *colouring* process. The semantics of the colouring is that operations of different colours are mutually exclusive. The colouring may be uniform across all of the containers in a container classification or it may be container class specific. The former implies that within a single containment tree no operations of different colours can be performed irrespective of the container they are applied to. The latter signifies that operations of differing colours may be performed if and only if they are not applied to the same container instance.

The two approaches are complementary. To account for either the OCD algorithm needs to be adjusted, as presented next.

Colouring Aware OCD

For clarity purposes we present only the extension of the constrained version of the OCD algorithm (Section 4.5.6). All changes are analogously applicable to the OCD extensions presented in Section 4.5.7.

Firstly, we consider the case of globally uniform container colouring. The OCD extension is specified straightforwardly as:

$$g(v, C, l_{tot}) = \mathrm{MIN}_{c \in C}\, f(v, c, n_v, 0, l_{tot})$$

where the original OCD definition of the function f has been extended by a single argument, c, denoting the colour of sticks, i.e. operations, to consider. Otherwise, the definition of the function f remains as presented in Section 4.5.6. Note that the above represents the initial call to the algorithm. Thus modified OCD, given by the function g above, effectively finds optimal solutions for all the colours and then chooses the one of the minimum cost among them.

In the case where the colourings are container class specific the required adjustment of OCD algorithm is slightly more complex. We define the function g in a manner analogous to the above, however being specific about which node's colouring scheme we are referring to each time around:

$$g(v, C_v, l_c, l_{tot}) = \mathrm{MIN}_{c \in C_v} \, f(v, c, n_v, l_c, l_{tot})$$

In addition to this we also need to alter the definition of the function f slightly to account for the actual independence of colouring schemes among containers:

$$f(v, i, c, l_c, l_{tot(v)}) = \begin{cases} 0 & \text{if } l_{tot(v)} \leq l_c \ ; \\ \sum_{k \in chld(v)} g(k, C_k, n_k, l_c, l_{tot(k)}) & \text{if } i = 0 \ \& \ !is\_leaf(v) \ ; \\ \infty & \text{if } i = 0 \ \& \ is\_leaf(v) \ ; \\ \min[f(v, i-1, l_c, l_{tot(v)}), & \\ \qquad c_i + f(v, i-1, l_c + l_i, l_{tot(v)})] & \text{otherwise ;} \end{cases}$$

The difference to the constrained OCD is in the call to the function g rather than a direct recursive call to f in the second case above.

The OCD algorithm awareness of the colouring scheme, in both the above cases, results in addition of another dimension to the OCD supporting, dynamic programming, data structure used for sub-problem results memorisation. The complexity, however, remains pseudo polynomial.

6.4.2 Termination Analysis

On a number of previous occasions we have stated that the OCD algorithm operates over multiple dimensions. The dimensions correspond to each of the parameters of the function f, and g above, that get iterated/recursed over.

Figure 6.8 depicts dimensions relevant for the termination analysis of the fully extended variant of the OCD algorithm. The (l_c, l_{tot}) pairs are depicted as one dimension with the red dot denoting l_{tot} and the axis denoting the progress toward it (l_c).

The recursive definition of the OCD algorithm, the constrained version as well as the extensions, is strictly monotonic with respect to the dimensions. Firstly, the algorithm always proceeds down a containment tree, starting with the initial node, never revisiting a container. This is given by the second base case of the constrained OCD definition. Secondly, the algorithm always decreases the number of sticks, i.e. threat mitigation operations, under consideration, starting with the full set. Thirdly, never does the algorithm backtrack on a stick that it made a decision about. In other words, it makes monotonic progress toward the l_{tot}. Finally, where a colouring scheme exists never is the same colour re-considered, relative to its scope.

Each of the dimensions is inherently finite and has a clearly defined minimum and maximum. The monotonicity of the algorithm in conjunction with the dimensions finiteness

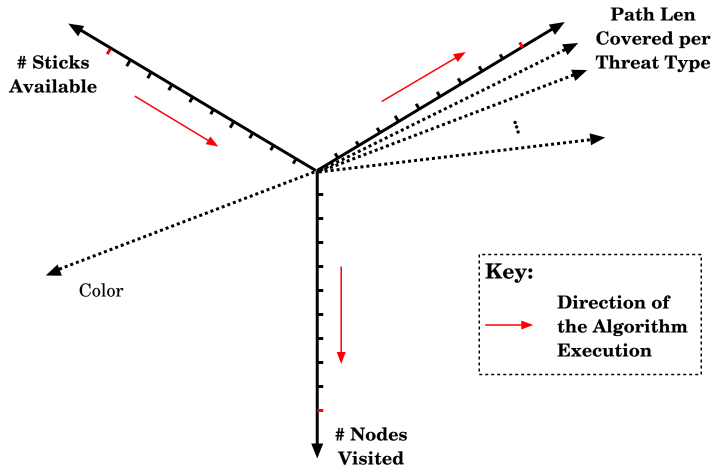

Path Len Covered per Threat Type
# Sticks Available
Color
Key:
Direction of the Algorithm Execution
# Nodes Visited

Figure 6.8: OCD operating dimensions.

implies that either the minimum or the maximum in each of the dimensions is reached after a finite number of steps. An exception is the path length dimension whose maximum does not have to be reached for the algorithm to terminate. The algorithm terminates when the available sticks dimension reaches the minimum and the tree depth dimension reaches its maximum (for every colour, if a mitigation operation colouring scheme is in place).

For the above reasons we are able to state that all of the presented variants of the OCD algorithm are guaranteed to terminate.

6.4.3 Comments on Complexity: Theory and Practise

In analogy to the complexity of the dynamic programming approach to solving the integer constrained 0-1 Knapsack problem (Section 4.5.4) the complexity of all presented variants of the OCD algorithm is pseudo polynomial. The order of the constrained OCD algorithm is given by:

$$\mathrm{O}(I \times N \times L^2)$$

where I is the number of sticks, N is number of nodes in a containment tree and L is the length of the longest path to cover.

Once extended to account for the simultaneous exposures of multiple threat types, in conjunction to the existence of compound threat types (Section 4.5.7), the OCD algorithm exhibits complexity in the order of:

$$\mathrm{O}(I \times N \times L_1^2 \times \ldots \times L_n^2)$$

where $L_1, \ldots, L_n$ represent lengths of the longest paths to cover for each of the co-occurring threat types enumerated as $1, \ldots, n$.

Further, the constrained OCD variant extended to account for threat mitigation operation non-composability, as presented above, exhibits complexity characterised by:

$$\mathrm{O}(I \times V \times L^2 \times C)$$

where C represents the number of different colours. The order applies to the case where colouring is uniform across a containment tree. Container specific colouring would contribute factors representing the number of colours to the above complexity.

The dynamic programming approach provided us with a pseudo polynomial solution to a constrained problem of what is in the general case an NP hard problem. In theory, this means that the algorithm is feasible — a result of significant importance. In practical terms, the resource overheads of the OCD algorithm come from:

- The exact size of the data structure supporting the memorisation of the solutions to sub-problems encountered during algorithm execution.

- The processing delays incurred by simple arithmetic operations and comparisons carried out for each of the cells of the data structure.

- Impact of the potential recursion in the algorithm implementation on the stack capacity — depending on the containment tree depth.

The dimensions of the supporting data structure are exactly the above orders of the algorithm variants. In Section 6.2 we have seen that for a reasonable grained container classification a realm corresponding to a general purpose laptop computer would be represented by a containment tree with the maximum number of containers of 73 and the maximum depth of 4. Further, we would be optimistic to envisage more than a one digit number of mitigating operations available on per container basis, including container insertion and migration operations. For the reasons presented in Section 6.2, the same applies to the total number of LoEs on per information exposure threat type.

Consequently, we do not envisage any of the above factors to cause any significant, or even non-negligible, resource overheads for general purpose computing platforms or mobile computing devices available on the market nowadays. For highly constrained platforms we propose the specialised μCASPEr specialisation of CASPEr, presented in Chapter 5.

6.5 Qualitative Analysis and Comparison

In Chapter 2, we provided an outline of the related research in the area of information security, focusing on information confidentiality protection. The current state-of-art in pervasive computing information security consists mainly of adaptations, and enhancements, of the paradigms presented in Chapter 2, developed for traditional computing environments, to the specific characteristics and usage scenarios envisaged for the novel setting. To best delineate the concepts comprising CASPEr, introduced in the previous chapters, and clarify the contribution, in this section we present a qualitative analysis of CASPEr and its comparison to the related work outlined in Chapter 2. The criteria used has been selected to best depict conceptual and functional gaps in information security, with an emphasis on pervasive computing environments, filled by CASPEr.

Although the connotation of a qualitative comparison is usually along the lines of mutual exclusivity of the compared solutions, with the intention of emphasising the contrast between them for the purpose of choosing one instead of another, in the case of CASPEr it is not so. Toward the end of the section we show how CASPEr complements, and is orthogonal to, most major information security paradigms. This is as CASPEr addresses a specific set of information confidentiality issues, brought into foreground by the vision of pervasive computing, rather than extending the existing mechanisms and paradigms developed for traditional computing environments.

Prior to delving into the core of the section we briefly discuss how an implementation may be seen from the industry perspective. For this purpose, we found it interesting to refer to the *Common Criteria*.

6.5.1 Common Criteria and CASPEr

The Common Criteria for Information Technology Security Evaluation [cc:b] [cc:a], or *Common Criteria* (CC) for short, aims at being a world-wide de-facto standard for evaluation of Information Technology (IT) products security. CC originates from three distinct criteria: European *Information Technology Security Evaluation Criteria* (ITSEC), United States *Trusted Computer System Evaluation Criteria* (TCSEC) — widely recognised as the "Orange Book" and *Canadian Trusted Computer Product Evaluation Criteria* (CTCPEC). Behind CC stand relevant EU, US and Canadian governing and sponsoring bodies.

The CC specifies requirements for the IT security of a product through a number of *functional requirements* [cc:05b] and *assurance requirements* [cc:05a]. Individual requirements of both kinds are grouped into distinct categories, and further into families. While the CC functional requirements specify security relevant behaviour of a product, assurance requirements serve as the basis for ensuring that the claimed security functionality is effective within a product.

Prospective consumers or developers can specify their security needs in terms of security requirements and objectives, leveraging CC functionality requirements, in the form of a CC *Protection Profile* (PP). A PP is intended to be reusable and to define requirements which are known to be effective and efficient with respect to a set of objectives, much like design patterns in software engineering. Thus, a PP describes a category of products or systems which have similar security relevant behaviour. Products are subsequently evaluated and certified against PPs. PPs have been developed for firewalls, databases, access control devices, smart cards etc.

Among all PPs registered at the CC portal [cc:b] only[5] the *Discretionary Information Flow Control (MU)* PP [LNSV02] specifies functionality that would partially describe a CASPEr implementation. The common grounds between CASPEr and the PP is in that it describes a product that protects information flows in a friendly user setting, can be considered as an add-on to well-established security concepts and can be integrated with a host system in a variety of ways. Although not an exact match for the CASPEr functionality, the PP shows how a system-wide information flow control security functionality can be described in the CC terminology.

Analysing the PP and the CC functionality requirements document [cc:05b] we have identified the CC functionality requirement categories that would be applicable to a CASPEr implementation as *User Data Protection* (FDP), *Security Management* (FMT), *Resource Utilisation* (FRU) and *Cryptographic Support* (FCS). However, we were unable to identify a straightforward way to leverage the CC functional requirements to express any form of context-aware or context-adaptive, proactive as well as reactive, security functionality. Moreover, we failed to describe, in CC terminology, any of the concepts novel to CASPEr as presented throughout the thesis and summarised in the quantitative comparison below.

The CC groups evaluated and certified products into categories according mainly to their *Target of Protection*. Among them is the *Data Protection* category. Several products in the category are targeted at data protection on mobile computing environments. Thus, they are best specimens from the commercial world comparable, to an extent, with CASPEr. However, all of them provide solely for indiscriminate, mandatory and inflexible storage level data protection. Nonetheless, for completeness reasons, we include them in the qualitative comparison under the collective name of *Static Data Protection* mechanisms. Examples[6] [cc:b] are *Pointsec PC Version 4.3*, *Encryption Plus Hard Disk 7.0*, *Protect Drive V7.0.3*, etc.

6.5.2 Qualitative Comparison

Comparison Criteria

As stated above, the criteria chosen for the CASPEr qualitative analysis and comparison were not meant to characterise the related work exhaustively but to best delineate CASPEr concepts. The comparison is based on 19 individual criteria grouped into 7 categories — as represented by the columns in the Figure 6.9. Grouped by the categories the criteria are as follows:

- **Protection Goal.** Protection goal of an information security mechanism can be one or more of the well-known *Confidentiality*, *Integrity* or *Availability* information security facets.

[5] Correct at the time of writing of this thesis.

[6] A number of non-CC certified products exist on the market that provide for the same or closely related functionality.

- **User Profile.** User profile characterises the envisaged attitude of information custodian, i.e. a person in a direct possession of information, toward preservation of information security. A custodian is said to be *hostile* if they are envisaged to make an active effort to violate information security and/or not make an active effort to protect information when aware of a potential compromise. Otherwise, a custodian is considered *friendly*.

- **Decision Criteria.** Decision criteria describes factors considered by individual mechanisms in their respective information security relevant decision making process.

  - *Subject Attributes* capture any information associated with a subject initiating an operation on, or being in command of, a piece of data. For example, subject attributes may incorporate subject's role(s) as in RBAC [FK92] [SCFY96], trust assigned to a subject as in [DBE$^+$04] and organisation [CM03] or a team [TS97] the corresponding user belongs to.
  - *Object Attributes* consist of any relevant meta-data associated with a piece of data, or information deducible from the data itself.
  - *Context* assumes reference to any information relating to the notion of context strictly as interpreted in this thesis (Section 2.2). Some researchers [BEM03] also refer to this as the state of *dynamic environment*.
  - *Side-Effects* criterion denotes explicit consideration of any non-information security related impact each particular decision may have. This criterion is targeted especially at characterising CASPEr with respect to its information utility (Section 4.2) maximisation.
  - *Sys/Apps State* criterion is introduced to disambiguate between the interpretation of the term context as used in this thesis and where it denotes internal state of an application, a system or where it names and identifies any of their sections as in [BEM03] [TS97] [BMY02].
  - *Other* criterion accounts for criteria not assumed by any of the above. For example, conditions and obligations in UCON (Section 2.8).
  - *Static* criterion characterises information security protection mechanisms with no support for dynamic, run-time, decision making based on any of the above criteria. Static decision making process characterises mandatory, indiscriminate, inflexible and unadaptable systems. This criterion is mutually exclusive to any of the above decision making criteria.

- **Binary Decision Model.** Binary decision model denotes a strictly two state result, e.g. *on/off* or *allow/deny*, of the respective decision making process.

- **Temporal Continuity.** Temporal continuity captures the temporal nature of the decision process and, more importantly, decision enforcement:

  - *Point-of-Request* denotes decision making and enforcement processes that occur, and are triggered by, explicit operational request on an object on behalf of a subject. They can be also described as one-off and on-demand. A typical example is traditional access control (Section 2.7), triggered by any form of access request. Point-of-Request is typical for passive[7] information security mechanisms overviewed in Chapter 2.
  - *Usage Bound.* We say that an active security mechanism is temporally usage bound if the relevant information security decisions are re-evaluated continuously during the period a piece of information is being used and enforced promptly. As opposed to Point-of-Request, the decision making process and subsequent decision enforcement are triggered by shifts in the state of the relevant decision criteria, such as e.g. context in the case of CASPEr.
  - *Life Long.* A mechanism provides life long information security if it re-evaluates its decisions and enforces them promptly throughout information life-time. In other words, from the point in time at which a piece of information enters a system under jurisdiction of the mechanism, by creation or otherwise, to the point at which it exits the system — not being bound to the usage period of the target piece of information.

- **Spatial Continuity.** Spatial continuity represents an analogy to the temporal continuity with respect to a particular whereabouts of a piece of information, e.g. storage device, display, communications channel etc. We say that an information security mechanism fulfils this criterion if: i) it is aware of and distinguishes between whereabouts of a piece of information throughout its life-time in a system; and ii) if its decision making process takes explicit account of the information whereabouts.

- **Platform Support.** By *Applications* support we mean that a mechanism requires, fully or partially, awareness from applications not related to the mechanism itself in order to provide for its functionality. If a mechanism requires *Hardware* support it means that it cannot function without a set of dedicated hardware components or functions thereof.

The Taxonomy

The Table in Figure 6.9 shows the quantitative comparison of CASPEr and the related work according to the above criteria. The columns of the table represent individual criteria while the rows are associated with the related work as outlined in Chapter 2. Rather than representing every individual piece of related work separately, we group them into higher-level paradigms, where applicable, with no loss of specificness with respect to the comparison

[7] The terms *passive* and *active* security mechanism originate from [TS97] and were outlined in Section 2.7.

		Protection Goal			User Profile		Decision Criteria							Binary Decision	Temporal Continuity			Spatial Continuity	Platform Support	
		Confidentiality	Integrity	Availability	Hostile Custodian	Friendly Custodian	Subject Attr.	Object Attr.	Context	Side-effects	Sys/App State	Other	Static		Point-of-Request	Usage Bound	Life-Long		Applications	Hardware
Flow Control		Y	N	N	Y	Y	N	Y	N	N	N	N	N	Y	Y	N	N	N	Y	Y
Dissemination Control (& DRM)		Y	Y	N	Y	N	Y	Y	N	N	Y	Y	Y*	Y	N	Y	Y*	N	Y	Y
Access Control	*DAC, MAC & $RBAC_{0,1}$*	Y	Y	N	Y	N	Y	Y	N	N	N	N	N	Y	Y	N	N	N	N	N
	Context-Aware AC	Y	Y	N	Y	N	Y	Y	Y	N	Y	N	N	Y	Y	N	N	N	N	N
	Active Access Management	Y	Y	N	Y	N	Y	Y	N	N	Y	N	N	Y	Y	Y*	N	N	N	N
Access Flow [Sto81]		Y	Y	N	Y	N	N	Y	Y	N	N	N	N	Y	N	N	Y	N	N	Y
Usage Control [PS02b, SP03]		Y	Y	N	Y	N	Y	Y	Y	N	Y	Y	Y*	Y	N	Y	Y*	N	N	N
Extended ZIA [CN03]		Y	Y	N	Y	N	Y	Y	N	N	N	N	N	Y	Y	N	N	N	N	N
Static Data Protection		Y	Y	N	Y	N	N	N	N	N	N	N	Y	Y	Y	Y	N	N	N	N
CASPEr		Y	Y*	Y	N	Y	N	Y	Y	Y	Y*	N	N	N	N	N	Y	Y	Y*	N

Figure 6.9: CASPEr qualitative analysis and comparison.

criteria. The categorisation obtained follows the general layout of the Chapter 2. The table rows corresponding to the higher-level paradigms are labelled in *italic*. The rows labelled in non-italic specify individual contributions, together with the appropriate references.

The only category of related work not self-explanatory from the Chapter 2 is labelled as *Static Data Protection*. The category represents mechanisms whose decision process can be characterised as static. In other words, the category encompasses mechanisms that are indiscriminate with respect to the data they are applied to and, in general, lack any dynamically adaptive behaviour according to the criteria from the Figure 6.9. Although the most notable examples are the data storage encryption solutions referred to in Sections 2.9 and 6.5.1, other data-wise indiscriminate solutions, such as network firewalls or mobile device functionality constraining policies (Section 2.9), fall under this category as well.

Every cell of the table contains either `Y`(es) or `N`(o) denoting whether a particular mechanism marking the corresponding row fulfils the criterion referred to by the column or not respectively. The symbol `Y` in a cell associated with a higher-level information security paradigm denotes that there exist realisations of the paradigm which fulfil the particular criterion. The `Y`* symbol is used where the fulfilment of a criterion is ambiguous, i.e. open to interpretation. In the Figure 6.9 this is true in several cases:

- *Dissemination Control and Usage Control.* In Chapter 2 we have stated that both DCON and UCON rely on the concept of a digital container, a cryptographic envelope, which contains the data to be subjected to controls according to the associated control set. While the dissemination and usage control, in DCON and UCON respectively, are essentially usage bound, the cryptographic mechanisms forming the digital container ensure the protection of data while it is not being actively accessed. Thus, we say that both DCON and UCON provide static data protection part of the information lifetime in the system. This, in turn, means that information is effectively protected throughout its lifetime in both DCON and UCON.

- *Active Access Management.* The ambiguity concerning active access management arises due to the level of abstraction at which the core models are specified in the literature. The main characteristic of the active access management paradigm, as originally defined in [TS97] and referred to in the subsequent work (Section 2.7), is the continuity of the actual decision making process — i.e. decision re-evaluation as the relevant conditions change, during the period of time in which the data is being actively used. However, what all the relevant literature referred to in Section 2.7 remains silent about is when the actual decision enforcement happens — leaving the issue to the implementation. For example, in both [TS97] and [BMY02] permission revocation occurs instantly while the enforcement happens on the subsequent authorisation decision point. In Figure 6.9, however, we acknowledge the possibility of the actual decision enforcement, alike decision evaluation, being usage bound as opposed to occurring only at the point-of-request.

- *CASPEr.* Qualitative analysis of CASPEr has resulted in three ambiguities. Firstly, although CASPEr focuses on information confidentiality protection, its features may be utilised to provide for information integrity in a limited number of cases. This is outlined in Section 6.2.5 and exploited in the example μCASPEr policy in Appendix A. Secondly, CASPEr can be considered as being *Sys/Apps State* aware in cases where the state is reflected on the respective containment configuration, e.g. size of a GUI window, deployment of cryptographic mechanisms etc. Finally, despite CASPEr having been designed to operate with variable degrees of application awareness, the lack of such does incur substantial impact on its effectiveness through the inability to model application-level containments. This impacts on the granularity at which both threat effects are modelled and mitigation strategies are devised. Furthermore, no application awareness would cause CASPEr to suffer from a problem analogous to the well known issue of the tendency of information security labels to accumulate over time in MLS systems in which fine-grained application awareness does not exist [And01].

With respect to the last point above, as CASPEr represents a framework, application support can be built into the host system in an incremental fashion, trading off information exposure protection granularity, the ability to maximise information utility and the spatial and temporal continuity against the effort and the time required to provide for the necessary levels of application awareness. We see the common starting point as protection of data on storage devices or otherwise when not in use by an application — where no application awareness is required.

6.5.3 Visualising CASPEr in the Big Picture

Further to the CASPEr qualitative analysis and the comparison presented in Figure 6.9, in this section we try to illustrate visually CASPEr placement within the big picture created by the major paradigms as included in the qualitative comparison. We do so from three distinct points of view — effectively dissecting the overall qualitative comparison from three different angles.

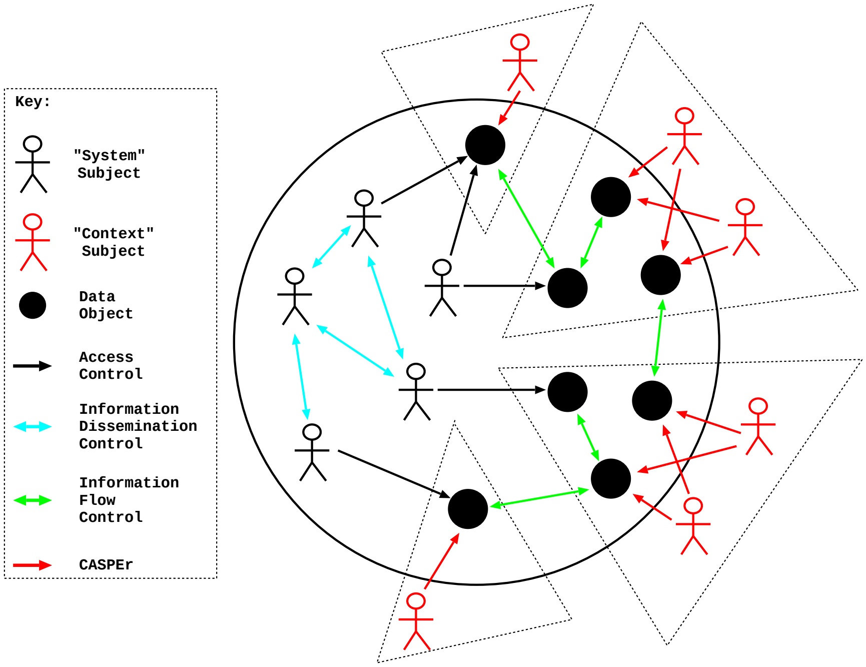


Figure 6.10: CASPEr in the information security big picture.

We start by visualising the CASPEr relation to the general concepts of information flow control, information dissemination control and access control as shown in Figure 6.10. The large circle in the centre of the figure represents a single operating platform or a system in its most abstract sense. It can be an isolated system but it can also be a distributed system of any kind, such as e.g. a peer-to-peer system or a specialised content distribution infrastructure. We use the term *system subjects* to refer to active entities within such a system which manipulate or otherwise make use of the data objects. For example, system subjects in an operating system denote the processes while in a content distribution system they may denote the individual computing platforms as a whole. The latter occurs, for example, in DRM systems in which a digital container is bound to the identity of the target hardware/software platform configuration. Data objects can be interpreted as data items in the CASPEr terminology.

Dash-lined triangles represent distinct, information exposure threat model relevant, contexts that affect the individual objects existing within them. The triangles effectively comprise a different, intersecting, dimension to the rest of the diagram. Subjects existing in the contexts represent active entities that give rise to information exposure threats — thus they are of fundamentally different nature to the subjects within the system, in spite of the same symbol used in the figure. We refer to them as *context subjects*. Ultimately, the context subjects may be thought of simply as human "attackers".

The arrows in the figure denote data flows, among the respective entities, that are points of focus of, i.e. are regulated and/or mediated by, the individual information security paradigms as included in the qualitative comparison in Figure 6.9. The arrows are colour coded as given in the figure key. Note that, for clarity reasons, the diagram does not depict all possible arrows of the same kind among the entities. Furthermore, although we do realise that from the point of view of information flow control system subjects can be regarded as objects we include no arrows that denote so, with no consequences for the argument. The diagram clearly delineates CASPEr functionality as laying in a different dimension to the one hosting information flow control, information dissemination control and access control — filling in a previously unoccupied place in the big picture.

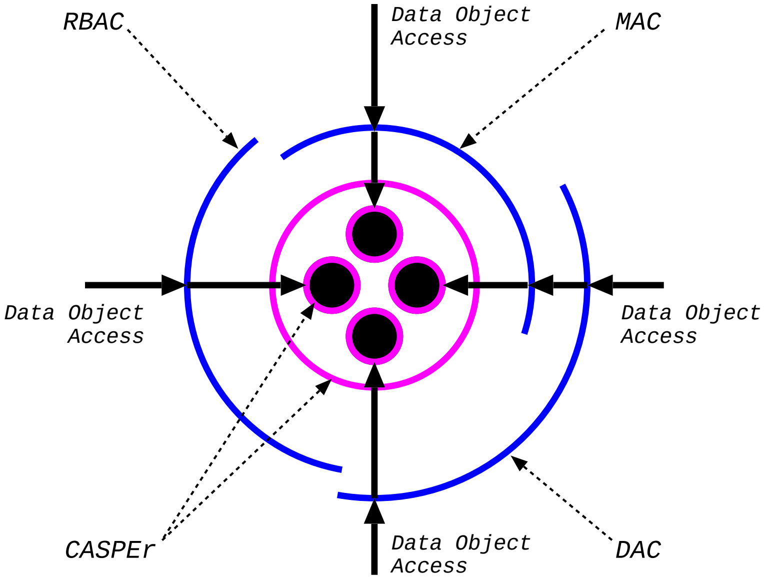


Figure 6.11: CASPEr in relation to DAC, MAC and RBAC.

Focusing solely on the access control paradigm, the diagram in the Figure 6.11 shows how CASPEr relates to the concepts of DAC, MAC and RBAC, including their decision-criteria enhanced variants (Section 2.7), such as e.g. context-aware access control. The diagram depicts, in a data object centric fashion, four different ways in which DAC, MAC and RBAC can be layered in a single system. We leave out the uncommon case in which all three access control models coexist in a system, although it has been suggested in the literature [SCFY96]. CASPEr itself is positioned the closest to the data objects, forming a wrapper that illustrates the containment paradigm. For each of the four layerings, the figure plots the data object access authorisation process, represented by the solid arrows. The corresponding access requests originate at higher levels. The manner in which CASPEr is represented also signifies its continuous role, not in the authorisation process itself, but in determining data information handling and management procedures employed and the form in which data is available pre-, in- and post- access. CASPEr plays no role whatsoever in determining rights of a subject to access an object but can be considered, in the context of the Figure 6.11, as a monitor mediating the manner in which information is accessed.

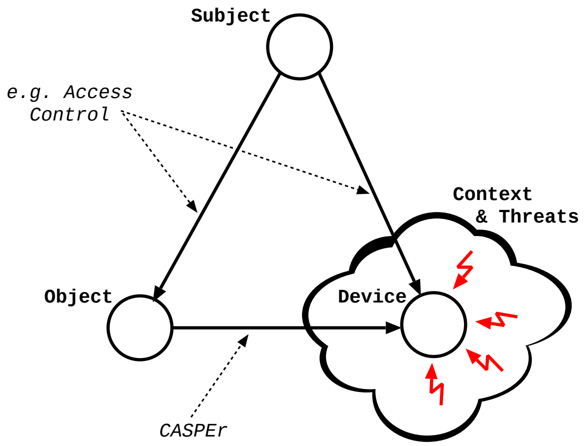


Figure 6.12: The Subject-Object-Device triangle.

Figure 6.12 shows, what we call, a *Subject-Object-Device* triangle. The term subject denotes a system, rather than context, subject while the notion of object refers to data objects. In the context of the figure, we refer to a *device* not as a ubiquitous computing platform, as throughout the thesis, but to denote the spatial whereabouts of a piece of information. In CASPEr terminology this corresponds to the notion of data item containment, including any relevant data management and handling procedures. More straightforwardly, the notion of a *device* can also be thought of as a single component of a computing platform, such as a display, a storage device or a communications channel.

A device itself is always embedded in a context which determines the threat model affecting a piece of information contained within the device at any point of time — as depicted by a cloud and the arrows in the figure respectively. Note that we refer to a threat model as describing information exposure threats. The arrows forming the triangle denote the relationships between the entities to which information security paradigms can apply. For example, as indicated on the diagram, access control mediates access requests made by subjects to both data objects and devices. CASPEr, on the other hand, influences the relationship between data objects and the devices they are contained within. The point that we would like to emphasise in particular is that, to the best of our knowledge again, no other information security paradigm applies to the object - device link as shown in the figure. The level of indirection created by linking object and device nodes via the subject node results in the failure to capture, and realise, one of the core concepts of CASPEr — information security protection at the data item granularity based solely on threats as present in the data item's environment.

6.5.4 Summary - Comments on CASPEr Uniqueness

Comparing CASPEr qualitatively to the related work, as given in the Figure 6.9, we can conclude that CASPEr is unique, to the best of our knowledge, from several points of view.

Firstly, CASPEr represents the only work in which contextual information is used for explicit threat modelling as presented in the thesis. Although other work in the area, as referred to in the Figure 6.9, does account for various aspects of contextual information we are not aware of any attempts at explicit threat modelling, reasoning and mitigation in a manner as accomplished by CASPEr for information exposure threats. The contribution is further reinforced by the definition of the information exposure threats themselves — representing a consequence of the data handling and management procedures as employed in a particular context.

Secondly, CASPEr is unique with respect to explicit consideration, in the decision making process, of the side-effects each alternative decision has. The more so as the side-effects taken into account are related to issues traditionally not considered security relevant, such as information utility or system usability. We consider the ability to provide "good-enough" security [San03], with respect to information utility and system usability criteria, as one of the most important factors for potential CASPEr acceptance and, more generally, for the realisation of the information omnipresence aspect of the ubiquitous computing vision.

Thirdly, one of the enabling factors of the above contributions is the departure from the binary decision model. We are not aware of any other work that offers similar degree of flexibility and adaptability in the decision making process in a similar fashion like proposed for CASPEr. This is in part owing to seeing a number of standard data management and handling procedures as potentially contributing to information exposure threat mitigation, as a side-effect of their primary functionality.

Referring back to the previous section, how CASPEr complements information flow control, information dissemination control and access control paradigms by filling in an information security gap brought into the spotlight by the ubiquitous computing vision and not previously analysed and addressed in a systematic and holistic manner. The place CASPEr occupies within the big picture represents an advance toward in-depth information security in highly dynamic environments — such as ubiquitous computing.

Chapter 7

Conclusions

In this dissertation we have introduced CASPEr, a theoretical framework for context-adaptive, data item centric, temporally and spatially continuous information exposure protection targeted at, but not confined to, pervasive computing environments. The motivation for CASPEr was the identification of information exposure threats, a sub-class of information leakage threats which gain substantial importance in ubiquitous computing environments. Through addressing information exposure, CASPEr fills a gap in the information security big picture not previously addressed in a systematic manner by traditional mechanisms and paradigms as well as their adaptations and extensions to the requirements of the novel environment. The manner in which we designed CASPEr caters for the platform heterogeneity envisaged for pervasive computing settings and facilitates complete operational autonomy of the target deployment devices. In this dissertation we have not aimed at introducing novel mechanisms to address specific instances of information security and privacy threats but have rather developed an original approach, the CASPEr framework, which enables leveraging of the existing ones, in conjunction with standard information handling and management procedures, for mitigation of the newly systematised class of information security threats.

In this chapter we summarise thesis contributions and comment on the potential avenues for future research. Overall, we consider the two main contributions of this thesis to be: i) identification and systematisation of the particular class of threats to information security; and ii) the approach, CASPEr, proposed for addressing the threats. The latter contribution is in itself comprised of further three major and several smaller contributions as presented in Chapters 3, 4 and 5 and summarised below.

7.1 Thesis Summary

In Chapter 1 we presented the motivation for this work. We described a set of scenarios that provide an intuitive feel for the nature of information exposure threats, their potential severity as well as the intended CASPEr behaviour in the face of the threats. We proceeded to formally introduce the concept of information exposure and the notion information exposure threats — the founding contribution of this dissertation. Data from real world security incident surveys in the mobile computing arena was used to reinforce the suggested severity magnitude and importance of the newly identified class of threats. We highlighted the challenges for information security posed by the vision of ubiquitous computing relevant to the thesis and pointed at the shortcomings of the current approaches in addressing them, further motivating our work.

The material presented in Chapter 2 served a two-fold purpose. Firstly, we outlined the general deployment setting for CASPEr and the constraints implied by it. Secondly, we portrayed the information security big picture to which this thesis contributes. The latter was accomplished by outlining four major information security paradigms and defining their scope in terms of the type of threats they address as well as the key actors and entities involved — the criteria that best delineates the contribution of this thesis. To complete the picture, we also highlighted efforts specific to mobile device information security — still immature in addressing the shift in threat models from the traditional computing environments appropriately. The detailed delineation of CASPEr with the related work was postponed until Chapter 6.

In Chapter 3 we present a major (sub-)contribution of this dissertation: a fine-grained, spatially and temporally continuous information exposure threat analysis method. The method rests on the containment based, data item centric approach to modelling the world, utilising the concept of a container, a protective enclosure, as its main building block — in themselves integral contributions of this chapter. Apart from providing for the threat analysis, we have shown how the model facilitates deployment in heterogeneous ubiquitous computing environment in a highly flexible and distributed fashion without requiring compromises on information security. Furthermore, we have described how the model inherently supports reasoning about strictly localised, focused and confined threat mitigation strategies. We introduced a number of formalisms, used throughout the dissertation, both to support the containment based model of the world and the threat analysis process itself. The threat analysis method was conceived to fully honour the autonomy of the target deployment platforms.

In Chapter 4 we focused on another major contribution of this dissertation: a fine-grained, localised, information exposure threat mitigation that maximises information omnipresence. We introduced information utility as a measure of information omnipresence and specified striking the balance of the information utility vs. level of protection seesaw as one of the main objectives for the threat mitigation. The concept of a Level of Exposure and the corresponding formal model served as the main instruments in achieving this goal. An important contribution of the chapter is the utilisation of standard information management and handling procedures for threat mitigation — enabled by the nature of threats themselves and the fine-grained threat analysis method. Marrying the concepts introduced in this and the previous chapter, we developed a dynamic programming algorithm for discovering the optimal mitigation strategy for a realm in the face of a set of information exposure threats

— one of the key contributions. The overall threat mitigation approach was conceived to fully honour the autonomy of the target deployment platforms.

In Chapter 5 we presented an instantiation of CASPEr concepts in a policy based system, referred to as μCASPEr — a further major contribution of the dissertation. The main contribution of the chapter lies within the CASPEr and μCASPEr architectures and the specific manner in which the TFFST policy model has been applied, including the policy structure and the related processes that we detailed. The inherent characteristics of the TFFST policy model allowed us to create the "split" system architecture in which resource intensive tasks are off-loaded to resource capable entities and accomplished *a priori* to the actual policy deployment. Furthermore, we have shown how the policy structure that we defined contributes to the flexibility of the overall model and the autonomy of the target platforms. As we showed in Chapter 6, the specific policy structure also had important consequences on the reduction of complexity at the Client side. Finally, we outlined a requirements and an architecture for a compliant data model.

In Chapter 6 we presented a discussion and (mostly) theoretical evaluation of the dissertation contributions. Firstly, we argued for the general feasibility of CASPEr through identifying, outlining and analysing factors contributing to its complexity and resource overheads at the conceptual level of abstraction. We addressed the issue of CASPEr generality through demonstrating its application to two scenarios which do not obviously fit the proposed threat analysis approach. Secondly, we provided a theoretical and practical, example driven, evaluation of μCASPEr in terms of the TFFST policy model related overheads. We identified the number of transitions in a TFFST and the out-degree of a TFFST as the main factors influencing the policy storage and evaluation overheads at the Client side. The evaluation suggested the general feasibility of μCASPEr deployment in the target setting. An accent was placed on showing the importance of the devised policy structure in minimising the overheads. Thirdly, we focused on the OCD algorithm — generalising it further, analysing its termination characteristics and showing its favourable, theoretical and practical, computational complexity. Finally, we presented a qualitative comparison that contrasts CASPEr with the related work overviewed in Chapter 2 — crystallising the distinct, contributing, position that CASPEr occupies within the information security big picture.

In conclusion, in this thesis we aimed at addressing the gap in the information security big picture corresponding to the information exposure threats and brought into the spotlight with the advent of the vision of ubiquitous computing. To accomplish this we developed CASPEr — a theoretical framework for fine-grained, information-centric, spatially and temporally continuous information exposure threat protection in an autonomic fashion for pervasive computing environments. We have shown CASPEr to be feasible at the commensurate, conceptual, level of abstraction and to represent a contribution to the field of information security — especially in the area of ubiquitous computing. Under the umbrella of the overall contribution we have presented a number of sub-contributions some of which may also be exploited independently of CASPEr and beyond the information security field.

These are, most notably, related to explicit contextual effect modelling and the containment based approach to modelling the world, presented in Chapter 3.

7.2 Comments on the Future Research

The ideas presented in this dissertation can be extended and built upon in a number of ways. Moreover, for the introduced, or similar, concepts and ideas to be captured fully in the form of a real-world system and evaluated in practise a large body of work, spanning a number of research areas, still remains to be done. We, thus, recognise that the dissertation gives rise to many more questions than it provides answers for. However, we also believe that this is inherent for any research that identifies a problem rooted as broadly and deeply as the one we attempted to address is. The above consideration is captured by acknowledging CASPEr as a theoretical framework rather than a ready-to-be-deployed solution.

In the light of the above, the research we intend to conduct in the near future is within the following three domains:

- *Threat Modelling and Reasoning.* Before we can benefit fully from the threat analysis approach introduced in Chapter 3, methods for context-threat correlation as well as container transparency modelling have to be investigated. We envisage these to draw from risk modelling (such as e.g. CORAS [FKG$^+$02] or OCTAVE [oct]), probabilistic estimation, history (experience) based techniques, etc. The feasibility and scope of controlled experimentation is also to be investigated. Another important and interesting aspect to determine is how the non-technical factors outlined in Section 6.2 influence LoE modelling granularity for different threat types in different environments. Without delving into the intricacies of contextual awareness, a further venue for exploration is the manner in which fusion of information from various levels of abstraction (from sensors to diaries and mobility models) and sources can be accomplished for threat model prediction and pro-active threat mitigation.

- *Systems Research.* The manner in which CASPEr is defined opens up a myriad of challenges to be resolved at the systems and architecture levels. We have just barely hinted at these in Chapter 5. One of the most notable, for example, is the issue of containment awareness within application level and operating system level components. This implies the ability to track at the run-time the flows of data both within the components and across their boundaries, system-wide, maintaining the association with relevant meta-data. Another challenge is within the threat mitigation and encompasses the registration, referencing to and the cross-layer, cross-component invocation of the relevant operations. Further, an event-based system is required for the threat notification. These example challenges only hint at the tip of the iceberg that will have to be explored to provide an overall implementation of the concepts proposed in the dissertation.

- *Usability Research.* At several points in this dissertation we have stressed the importance of usability — particularly emphasised in the context of ubiquitous computing. The usability challenge within CASPEr is, roughly speaking, two-fold: the first relates to administration of various parameters and policy definition while the second is concerned with the end-user experience. Both need to be addressed adequately. For the former we envisage a range of solutions: more complex ones requiring specific technical knowledge but providing for finer granularity and increased flexibility — deployable in larger organisations with dedicated staff; and intuitive ones, with user interfaces based on intuitive graphical analogies — for cases in which the role of administrator intersects that of the average user. Adequate models, methods and languages will have to be developed. With respect to the user experience, the challenge is to strike the right balance between seamlessness and seamfulness (i.e. behavioural transparency) to justify the effects of the adaptation decisions, in order to increase confidence in the system through clarifying its behaviour in ambiguous situations.

One of the benefits of the manner in which CASPEr was introduced, and the structure of the contributions, is that its practical deployment can proceed gradually — with each increment providing additional, stand-alone, functionality — working towards the holistic, unified, vision. For example, without any research on fine-grained context threat modelling, assuming solely a binary coarse-grained approach, and ignoring the information utility trade-off, an implementation of the other concepts, most notably the containment based model of the world, would still enable spatially and temporally continuous information exposure threat protection; provision of containment-awareness only at the operating system level, with no application support, would have a substantial effect on the granularity at which the protection could be provided, but would similarly retain the benefits of continuous protection; limiting the container awareness and modelling to solely a single container class, e.g. storage device — in fact, fully avoiding containment modelling, and providing solely for a single protection alternative, e.g. encryption, would still retain the benefits of context-adaptive, pro-active, behaviour. The last example effectively shows how CASPEr encompasses some of the current efforts in mobile device storage protection, such as e.g. [CN03].

We are committed to pursuing the presented line of work in the future as outlined above. At the time of writing of this text there exists an active collaboration among the authors of [DBVC06] in building and evaluating μCASPEr and, more notably, merging it with the system for autonomic mobility support in 4G systems [VBS^{+}05] — providing for an integrated, autonomic, solution for secure mobile device operation and communication in 4G environments.

Appendix A

μCASPER Policy Example

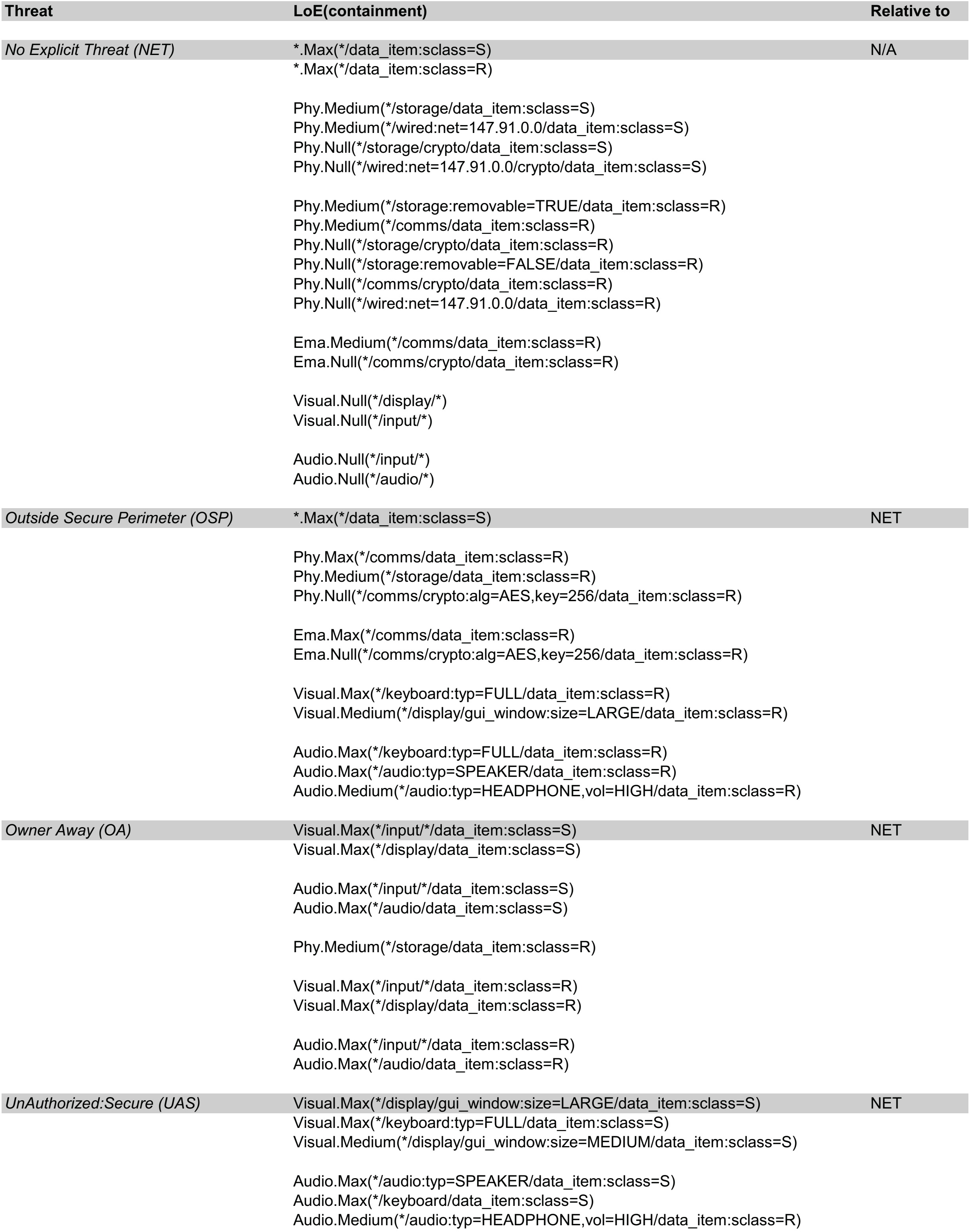
The purpose of this appendix is to present a simple μCASPEr policy in order to illustrate the concepts presented throughout the thesis, especially in Chapter 5, and complement the covered material. Furthermore, the example policy hints at complexities, or lack thereof, involved in specifying a complete, functional, μCASPEr policy for a ubiquitous computing device. For clarity reasons and space limitations we do not present individual policy rules in Ponder policy specification language. The description of the policy that we provide instead is straightforward to translate to Ponder following the analogy with the examples from Chapter 5. Having defined the policy, we provide three sets of TFFSTs corresponding to a selection of data item containments as supported by the hardware and software platform of the example device. The TFFSTs have been obtained by translation from Ponder as explained in Section 5.5.

A.1 Example Policy — Threat Model View

We present the example policy from two points of view: the threat model view — this section, and the containment view — the following section. The threat model view of the example policy is specified in the form of a table, split between the Figures A.1 and A.2. The first column of the table contains description of contextual state leading to a information exposure threat while the second specifies, for each of the threats, LoEs that should be activated for any potentially exposed data item containment. To preserve space, policy for each of the threats is specified relative to the policy as indicated by the third column — `NET` in the majority of cases. The example device for which the policy is specified corresponds to the containable relationship definition from Figure 6.1 (Section 6.2) and can be thought of as a PDA.

In the example policy we account for 7 distinct threats, as follows:

- *Outside Secure Perimeter* (OSP): the device is outside the physically secure perimeter.



Threat	LoE(containment)	Relative to
No Explicit Threat (NET)	*.Max(*/data_item:sclass=S)	N/A
	.Max(/data_item:sclass=R)	
	Phy.Medium(*/storage/data_item:sclass=S)	
	Phy.Medium(*/wired:net=147.91.0.0/data_item:sclass=S)	
	Phy.Null(*/storage/crypto/data_item:sclass=S)	
	Phy.Null(*/wired:net=147.91.0.0/crypto/data_item:sclass=S)	
	Phy.Medium(*/storage:removable=TRUE/data_item:sclass=R)	
	Phy.Medium(*/comms/data_item:sclass=R)	
	Phy.Null(*/storage/crypto/data_item:sclass=R)	
	Phy.Null(*/storage:removable=FALSE/data_item:sclass=R)	
	Phy.Null(*/comms/crypto/data_item:sclass=R)	
	Phy.Null(*/wired:net=147.91.0.0/data_item:sclass=R)	
	Ema.Medium(*/comms/data_item:sclass=R)	
	Ema.Null(*/comms/crypto/data_item:sclass=R)	
	Visual.Null(*/display/*)	
	Visual.Null(*/input/*)	
	Audio.Null(*/input/*)	
	Audio.Null(*/audio/*)	
Outside Secure Perimeter (OSP)	*.Max(*/data_item:sclass=S)	NET
	Phy.Max(*/comms/data_item:sclass=R)	
	Phy.Medium(*/storage/data_item:sclass=R)	
	Phy.Null(*/comms/crypto:alg=AES,key=256/data_item:sclass=R)	
	Ema.Max(*/comms/data_item:sclass=R)	
	Ema.Null(*/comms/crypto:alg=AES,key=256/data_item:sclass=R)	
	Visual.Max(*/keyboard:typ=FULL/data_item:sclass=R)	
	Visual.Medium(*/display/gui_window:size=LARGE/data_item:sclass=R)	
	Audio.Max(*/keyboard:typ=FULL/data_item:sclass=R)	
	Audio.Max(*/audio:typ=SPEAKER/data_item:sclass=R)	
	Audio.Medium(*/audio:typ=HEADPHONE,vol=HIGH/data_item:sclass=R)	
Owner Away (OA)	Visual.Max(*/input/*/data_item:sclass=S)	NET
	Visual.Max(*/display/data_item:sclass=S)	
	Audio.Max(*/input/*/data_item:sclass=S)	
	Audio.Max(*/audio/data_item:sclass=S)	
	Phy.Medium(*/storage/data_item:sclass=R)	
	Visual.Max(*/input/*/data_item:sclass=R)	
	Visual.Max(*/display/data_item:sclass=R)	
	Audio.Max(*/input/*/data_item:sclass=R)	
	Audio.Max(*/audio/data_item:sclass=R)	
UnAuthorized:Secure (UAS)	Visual.Max(*/display/gui_window:size=LARGE/data_item:sclass=S)	NET
	Visual.Max(*/keyboard:typ=FULL/data_item:sclass=S)	
	Visual.Medium(*/display/gui_window:size=MEDIUM/data_item:sclass=S)	
	Audio.Max(*/audio:typ=SPEAKER/data_item:sclass=S)	
	Audio.Max(*/keyboard/data_item:sclass=S)	
	Audio.Medium(*/audio:typ=HEADPHONE,vol=HIGH/data_item:sclass=R)	

Figure A.1: Example policy: threat model view, part 1.

UnAuthorized:Restricted(UAR)	Visual.Max(*/keyboard:typ=FULL/data_item:sclass=R)	UAR
	Visual.High(*/display/gui_window:size=LARGE/data_item:sclass=R)	
	Visual.Medium(*/display/gui_window:size=MEDIUM/data_item:sclass=R)	
	Audio.Max(*/audio:typ=SPEAKER/data_item:sclass=R)	
	Audio.Max(*/keyboard.typ=FULL/data_item:sclass=R)	
	Audio.Medium(*/audio:typ=HEADPHONE,vol=HIGH/data_item:sclass=R)	
OSP + OA	*.Max(*/data_item:sclass=S)	OSP
	.Max(/data_item:sclass=R)	
	Phy.Null(*/storage/crypto:alg=AES,key=256/data_item:sclass=R)	
OSP + UnAuthorized:Unknown (UAU)	Visual.High(*/display/gui_window:size=LARGE/data_item:sclass=R)	OSP
	Visual.Medium(*/display/gui_window:size=MEDIUM/data_item:sclass=R)	
	Audio.Max(*/audio:typ=HEADPHONE,vol=HIGH/data_item:sclass=R)	
	Audio.Medium(*/audio:typ=HEADPHONE,vol=MEDIUM/data_item:sclass=R)	
OSP + OA + UAU	*.Max(*/data_item:sclass=S)	NET
	.Max(/data_item:sclass=R)	

Figure A.2: Example policy: threat model view, part 2.

- *Owner Away* (OA): the physical proximity, or binary presence state, of the device's owner, or custodian, is below a threshold.
- *UnAuthorized:Secure* (UAS): an identifiable third party with a security clearance below *Secure* is in the vicinity of the device.
- *UnAuthorized:Restricted* (UAR): in analogy to the UAS, for the *Restricted* security class.
- *OSP + OA*: the device is outside the physically secure perimeter and the proximity of the device's owner is below the threshold.
- *OSP + UnAutorized:Unknown* (OSP + UAU): the device is outside the physically secure perimeter and a third party is present.
- *OSP + OA + UAU*: the device is outside the physically secure perimeter, the owner is away and presence of a third party is sensed.

Note that we assume that every person present within the secure perimeter can be identified. If a person is not identifiable the threat associated with UAR contextual state applies — assuming *Restricted* is the lowest information sensitivity class.

The entries in the second column are of the general form given by `LoE(containment)` and denote the LoE to be activated for a particular data item containment. The LoE identifier is of the form `ThreatType.Quantifier`. The quantifier ranges between *Null*, *Medium* and *Max*, where the *Null* value represents no significant exposure and *Max* denotes maximum exposure. The $\star$ character replacing the threat type in the LoE identifier replaces all supported threat types — `Phy` (Physical), `Audio`, `Ema` (Emanation) and `Visual`. Data item containments for which a LoE should be activated given a threat are defined using the containment path expressions, as defined in Section 3.4. We use container attribute `sclass` to denote security class of a target data item (`S` for *Secret* and `R` for *Restricted*). For example, the containment expression `mdev/storage:removable=FALSE/data_item:sclass=S` matches data items of security class *Secret* (`sclass=S`), stored on non-removable storage devices (`removable=FALSE`) of any mobile computing unit (`mdev`).

Note that the policy specified for the three contextual states in Figure A.2 is different from concatenation of individual policies as specified for their contextual sub-states individually. This is as, in general, more complex states imply more extensive threats — i.e. the sum is more than its parts.

A.2 Example Policy — Containment View

While the figures A.1 and A.2 show threat model view of the complete example policy, Figure A.3 illustrates a subset of the policy from the point of view of a selection of data item containments. For each of the data item containments the table specifies the LoEs that should be activated (the third row) for each of the known threats (the second row) as well as a set of applicable alternative threat mitigation operations (the fourth row). The format of the first three columns is as specified above. The mitigation operations applicable to each of the threats, at the corresponding containments, are specified in terms of identifiers which were chosen solely for illustration and are self-describing.

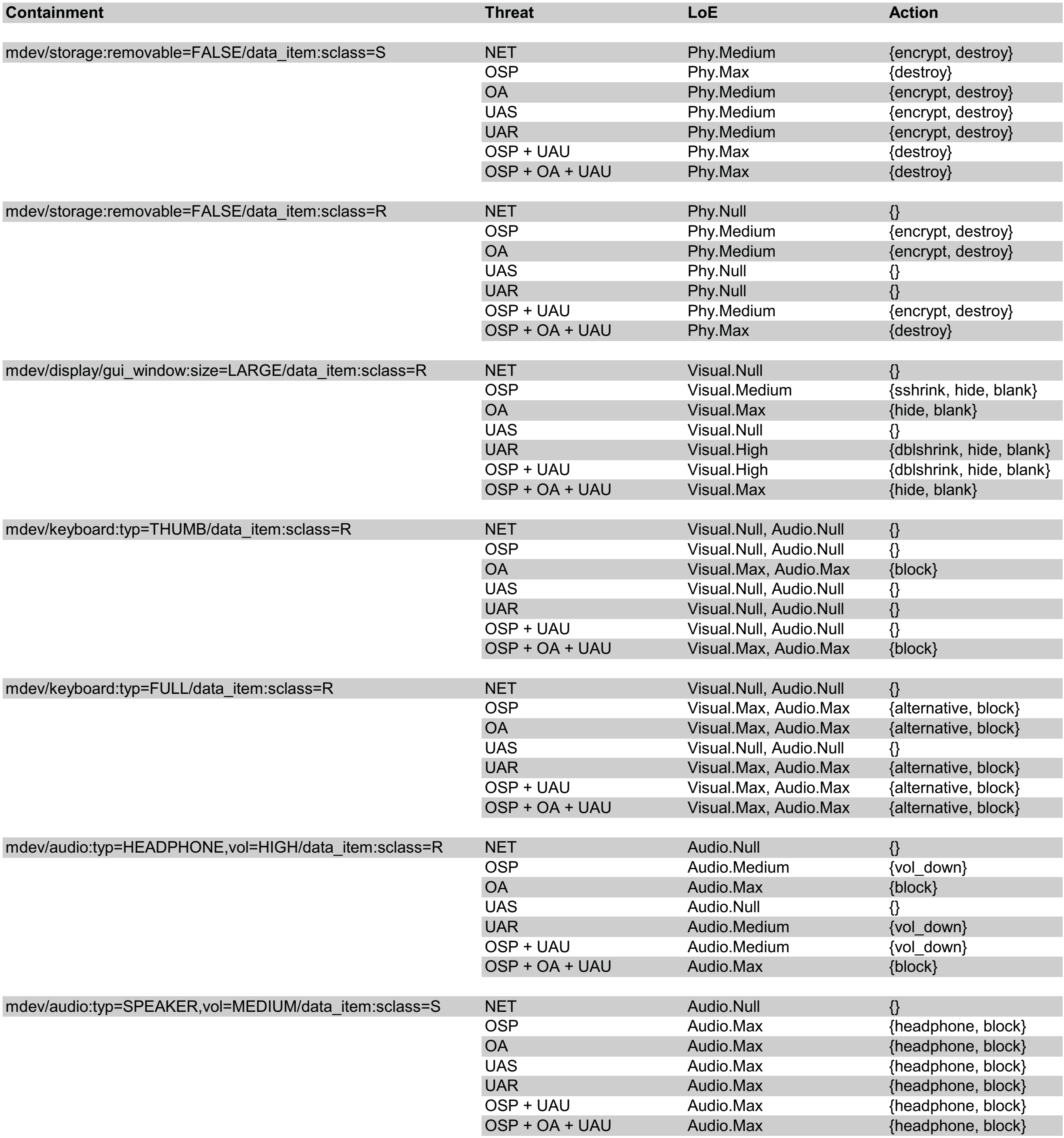
A.3 Example Policy — TFFST View

Having shown the high-level policy specification from two points of view, in this section we present a selection of corresponding TFFSTs as obtained from the process of policy translation (Section 5.5).

Figures A.4, A.5, A.6 and A.7 correspond to the first data item containment from the Figure A.3 (`mdev/storage:removable=FALSE/data_item:sclass=S`). Figure A.4 shows the intermediate FST representation of contextual state to LoE mappings for the data item containment. Figure A.5 shows the corresponding LoE to mitigation operation mapping. The labels on the transitions are specified in terms of TF algebra 5.4: `ata` stands for "as taut as" operator while `tt` stands for "touter than" operator. The composition of the two intermediate TFFSTs is depicted in Figure A.6. In a real world application we expect that a policy would include a `TimeOver` delay, as specified in Rule 4 of Section 5.4, and would further be composed with the `PingPoingConstraint` policy rule specified as a constraint Rule 5 in Section 5.4. For this case, the fully composed TFFST for the containment is given in the Figure A.7.

Moving on to the sixth data item containment of the Figure A.3 (`mdev/audio:typ=HEADPHONE,vol=HIGH/data_item:scalss=R`), Figures A.8, A.9 and A.10 represent the intermediate and the complete TFFSTs for the containment in an analogous fashion as presented for the first data item containment above. The composition of the final TFFST (Figure A.10) with the `PingPongConstraint` is not presented for simplicity purposes.

Finally, Figure A.11 shows a complete TFFST, including the composition with the `TimeOver` and `PingPongConstraint` policy rules, for the third data item containment from Figure A.3. The intermediate stages to obtaining the complete TFFST are analogous to the ones presented above. With respect to the example policy, the TFFST shown in the Figure A.11 is the most complex one. As such we use it for the policy model overheads evaluation in Section 6.3, Chapter 6.



Containment	Threat	LoE	Action
mdev/storage:removable=FALSE/data_item:sclass=S	NET	Phy.Medium	{encrypt, destroy}
	OSP	Phy.Max	{destroy}
	OA	Phy.Medium	{encrypt, destroy}
	UAS	Phy.Medium	{encrypt, destroy}
	UAR	Phy.Medium	{encrypt, destroy}
	OSP + UAU	Phy.Max	{destroy}
	OSP + OA + UAU	Phy.Max	{destroy}
mdev/storage:removable=FALSE/data_item:sclass=R	NET	Phy.Null	{}
	OSP	Phy.Medium	{encrypt, destroy}
	OA	Phy.Medium	{encrypt, destroy}
	UAS	Phy.Null	{}
	UAR	Phy.Null	{}
	OSP + UAU	Phy.Medium	{encrypt, destroy}
	OSP + OA + UAU	Phy.Max	{destroy}
mdev/display/gui_window:size=LARGE/data_item:sclass=R	NET	Visual.Null	{}
	OSP	Visual.Medium	{sshrink, hide, blank}
	OA	Visual.Max	{hide, blank}
	UAS	Visual.Null	{}
	UAR	Visual.High	{dblshrink, hide, blank}
	OSP + UAU	Visual.High	{dblshrink, hide, blank}
	OSP + OA + UAU	Visual.Max	{hide, blank}
mdev/keyboard:typ=THUMB/data_item:sclass=R	NET	Visual.Null, Audio.Null	{}
	OSP	Visual.Null, Audio.Null	{}
	OA	Visual.Max, Audio.Max	{block}
	UAS	Visual.Null, Audio.Null	{}
	UAR	Visual.Null, Audio.Null	{}
	OSP + UAU	Visual.Null, Audio.Null	{}
	OSP + OA + UAU	Visual.Max, Audio.Max	{block}
mdev/keyboard:typ=FULL/data_item:sclass=R	NET	Visual.Null, Audio.Null	{}
	OSP	Visual.Max, Audio.Max	{alternative, block}
	OA	Visual.Max, Audio.Max	{alternative, block}
	UAS	Visual.Null, Audio.Null	{}
	UAR	Visual.Max, Audio.Max	{alternative, block}
	OSP + UAU	Visual.Max, Audio.Max	{alternative, block}
	OSP + OA + UAU	Visual.Max, Audio.Max	{alternative, block}
mdev/audio:typ=HEADPHONE,vol=HIGH/data_item:sclass=R	NET	Audio.Null	{}
	OSP	Audio.Medium	{vol_down}
	OA	Audio.Max	{block}
	UAS	Audio.Null	{}
	UAR	Audio.Medium	{vol_down}
	OSP + UAU	Audio.Medium	{vol_down}
	OSP + OA + UAU	Audio.Max	{block}
mdev/audio:typ=SPEAKER,vol=MEDIUM/data_item:sclass=S	NET	Audio.Null	{}
	OSP	Audio.Max	{headphone, block}
	OA	Audio.Max	{headphone, block}
	UAS	Audio.Max	{headphone, block}
	UAR	Audio.Max	{headphone, block}
	OSP + UAU	Audio.Max	{headphone, block}
	OSP + OA + UAU	Audio.Max	{headphone, block}

Figure A.3: Example policy: partial data item containment view.

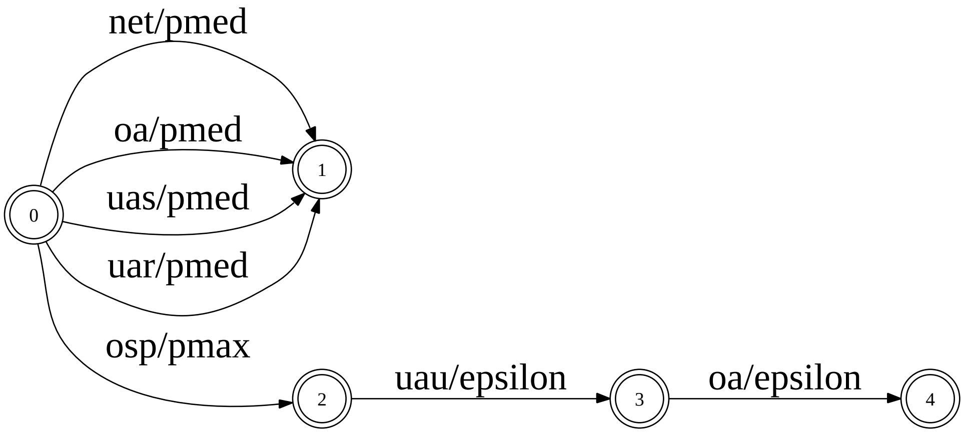


Figure A.4: Containment 1: contextual state - LoE mapping TFFST.

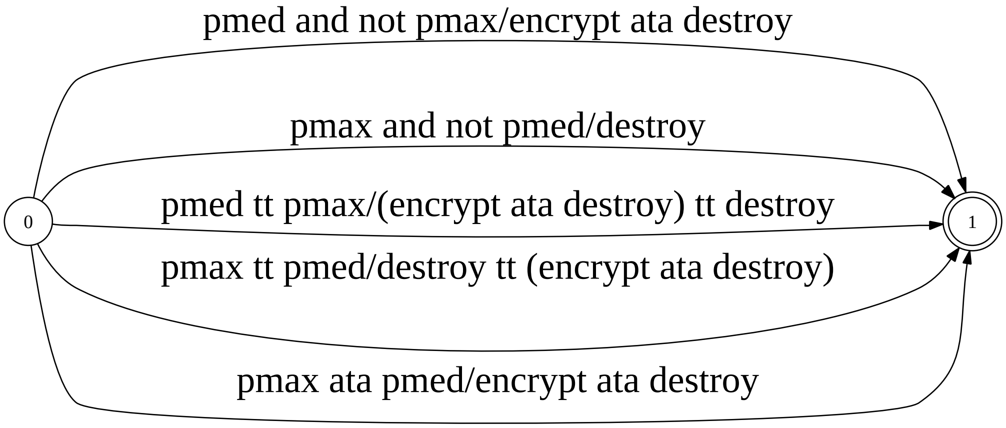

pmed and not pmax/encrypt ata destroy
pmax and not pmed/destroy
pmed tt pmax/(encrypt ata destroy) tt destroy
pmax tt pmed/destroy tt (encrypt ata destroy)
pmax ata pmed/encrypt ata destroy
0
1

Figure A.5: Containment 1: LoE - mitigation operation mapping TFFST.

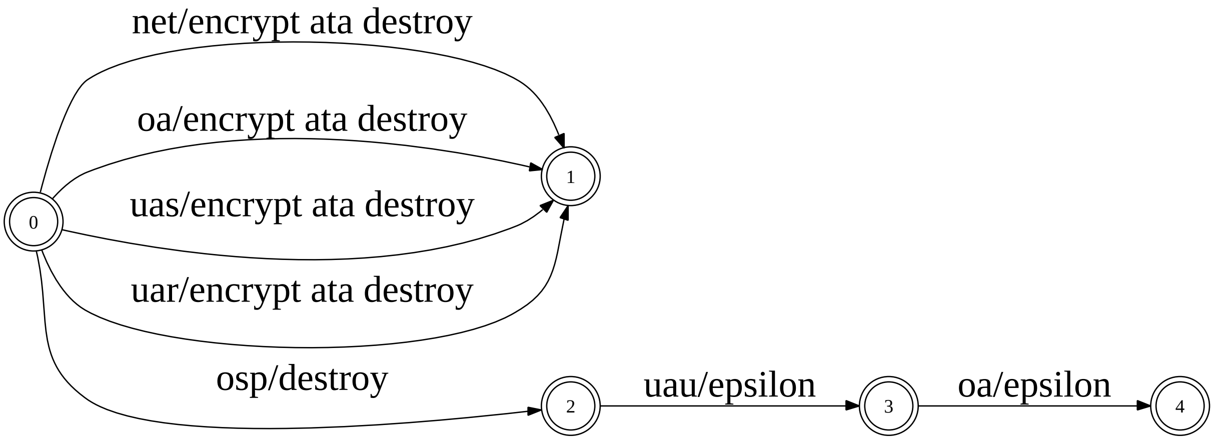


Figure A.6: Containment 1: contextual state - mitigation operation TFFST.

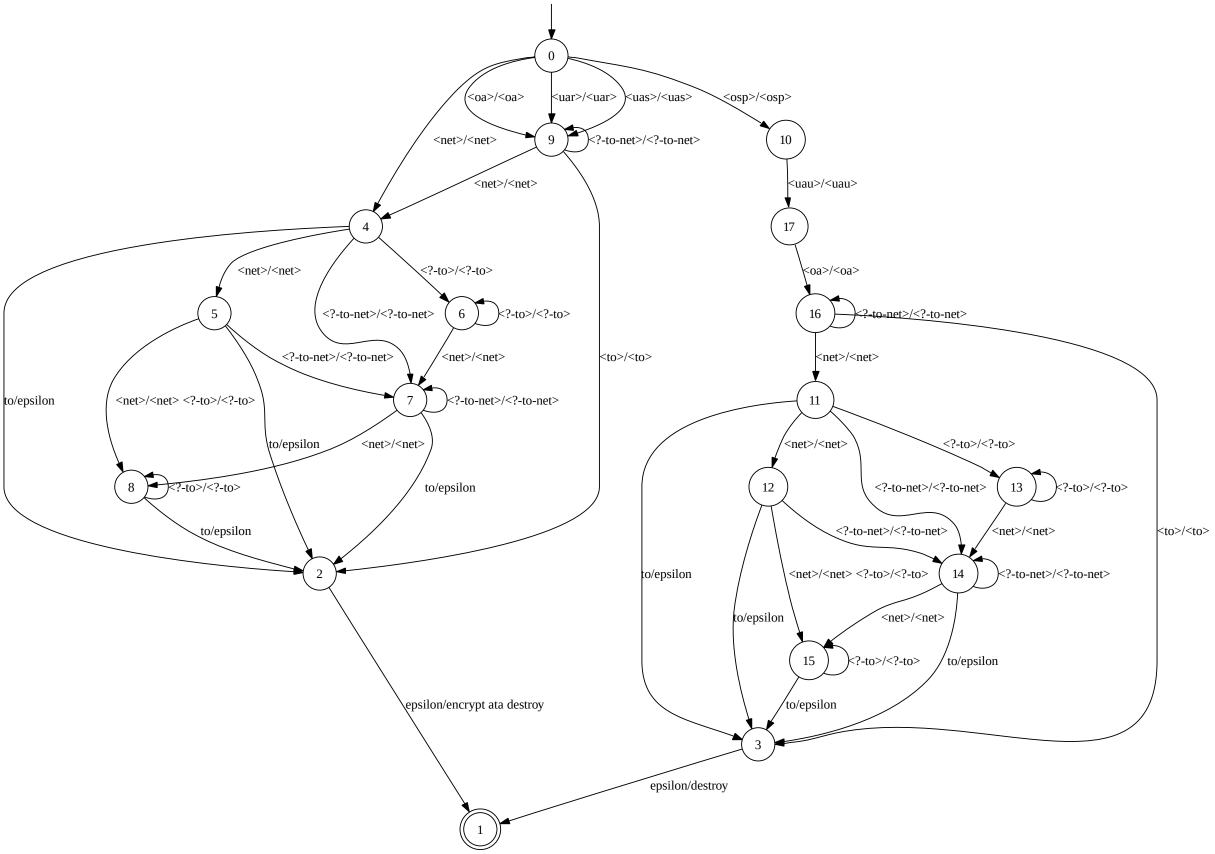


Figure A.7: Containment 1: full policy TFFST.

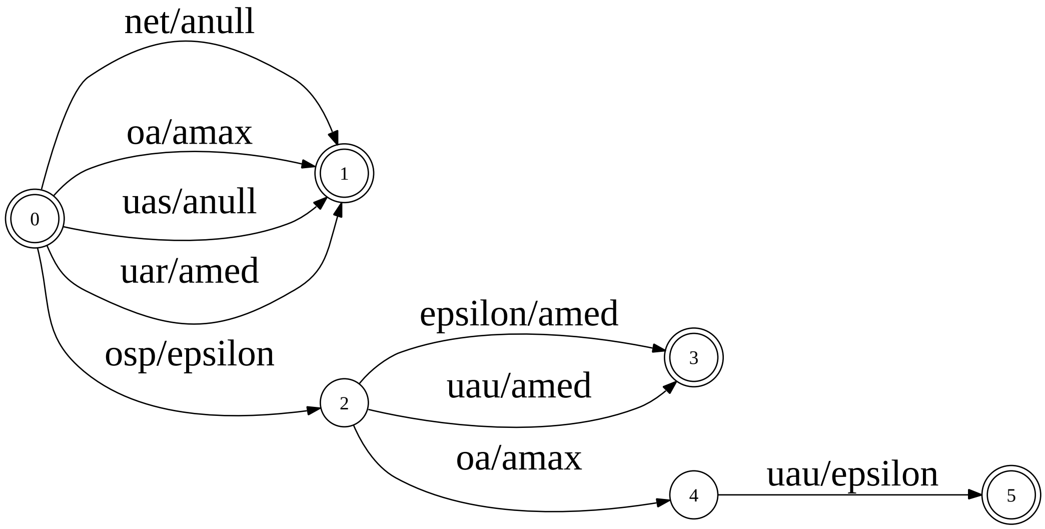


Figure A.8: Containment 6: contextual state - LoE mapping TFFST.

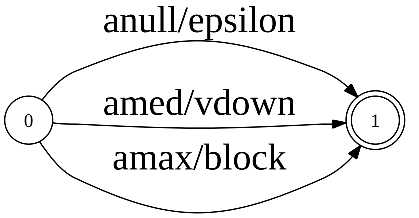


Figure A.9: Containment 6: LoE - mitigation operation mapping TFFST.

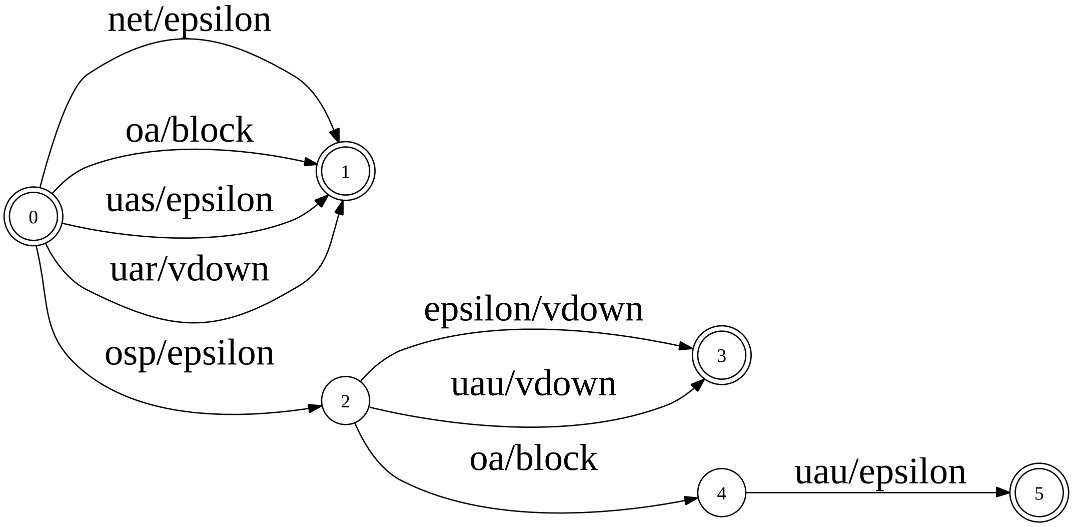


Figure A.10: Containment 6: contextual state - mitigation operation TFFST.

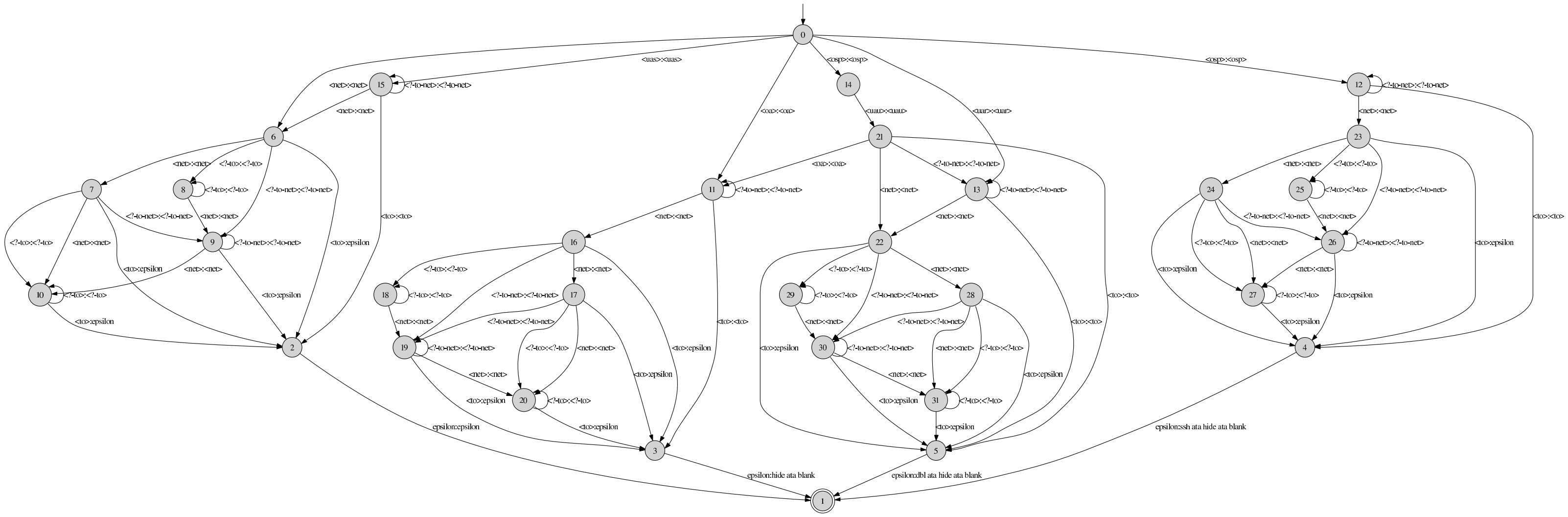


Figure A.11: Containment 3: full policy TFFST.

References

[AB00] J. F. Anderson and R. L. Brown. Risk and insurance. *Study Notes, Society of Actuaries*, (1-21-00), 2000. (Ref: p. 86.)

[ABC$^+$98] R. Anderson, F. Bergadano, B. Crispo, J.-H. Lee, C. Manifavas, and R. Needham. A new family of authentication protocols. *SIGOPS Operating Systems Review*, 32(4):9–20, 1998. (Ref: p. 22.)

[ADB$^+$99] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles. Towards a better understanding of context and context-awareness. In *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computig (HUC)*, volume 1707 of *Lecture Notes in Computer Science*, pp. 304–307. September 1999. (Ref: p. 15, 16.)

[AF03] M. Abadi and C. Fournet. Access control based on execution history. In *Proceedings of the Network and Distributed System Security Symposium*, pp. 107–121. 2003. (Ref: p. 33.)

[Age82] N. S. Agency. Tempest fundamentals. Technical Report NACSIM 5000, National Security Agency, 1982. (Ref: p. 74.)

[AHK$^+$91] M. Abrams, J. Heaney, O. King, L. LaPadula, M. Lazear, and I. Olson. Generalized framework for access control: Toward prototyping the ORGCON policy. In *Proceedings of the 14th NIST-NCSC National Computer Security Conference*, pp. 257–266. 1991. (Ref: p. 28.)

[Amo94] E. G. Amoroso. *Fundamentals of Computer Security Technology*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1994. (Ref: p. 26.)

[AMRCM03] J. Al-Muhtadi, A. Ranganathan, R. Campbell, and M. D. Mickunas. Cerberus: A context-aware security scheme for smart spaces. In *Proceedings of the 1st IEEE International Conference on Pervasive Computing and Communications (PERCOM)*, p. 489. 2003. (Ref: p. 39.)

REFERENCES

[And72] J. Anderson. Computer security technology planning study. Technical Report ESD-TR-73-51, Electronic System Division/AFSC, Hanscom AFB, Bedford, MA 01731, October 1972. [NTIS AD-758 206]. (Ref: p. 29, 31, 40.)

[And93] R. Anderson. Why cryptosystems fail. In *Proceedings of the 1st ACM Conference on Computer and Communications Security (CCS)*, pp. 215–227. 1993. (Ref: p. 7.)

[And01] R. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. John Wiley & Sons, Inc., 2001. (Ref: p. 4, 22, 23, 24, 25, 26, 31, 33, 170, 180.)

[AS99] A. Adams and M. A. Sasse. Users are not the enemy. *Communications of the ACM*, 42(12):40–46, 1999. (Ref: p. 4.)

[Ass] D. C. C. Association. DVD-CCS. Website at http://www.dvdcca.org. (Ref: p. 29.)

[AVTO03] P. G. Argyroudis, R. Verma, H. Tewari, and D. O'Mahony. Performance analysis of cryptographic protocols on handheld devices. Technical Report TCD-CS-2003-46, Trinity College Dublin Computer Science Department, November 2003. (Ref: p. 22, 169.)

[Bar97] J. Barkley. Comparing simple role based access control models and access control lists. In *Proceedings of the 2nd ACM Workshop on Role-Based Access Control*, pp. 127–132. ACM Press, New York, NY, USA, 1997. (Ref: p. 37.)

[BB94] T. Benkart and D. Bitzer. Bfe applicabilityto lan environments. In *Proceedings of the 17th National Computer Security Conference*, pp. 227–236. NIST, October 1994. (Ref: p. 26.)

[BBF01] E. Bertino, P. A. Bonatti, and E. Ferrari. Trbac: A temporal role-based access control model. *ACM Transactions on Information and Systems Security*, 4(3):191–233, 2001. (Ref: p. 38.)

[BCDP05] E. Bertino, B. Catania, M. L. Damiani, and P. Perlasca. GEO-RBAC: a spatially aware rbac. In *Proceedings of the 10th ACM Symposium on Access Control Models and Technologies (SACMAT)*, pp. 29–37. ACM Press, New York, NY, USA, 2005. (Ref: p. 39.)

[BDGS04] D. Balfanz, G. Durfee, R. E. Grinter, and D. Smetters. In search of usable security: Five lessons from the field. *IEEE Security & Privacy*, 2(5):19–24, 2004. (Ref: p. 7.)

[BEM03] A. Belokosztolszki, D. M. Eyers, and K. Moody. Policy contexts: Controlling information flow in parameterised rbac. In *Proceedings of the 4th International Workshop on Policies for Distributed Systems and Networks (POLICY)*, pp. 99–110. 2003. (Ref: p. 38, 177.)

[BF85] W. E. Boebert and C. T. Ferguson. A partial solution to the discretionary trojan horse problem. In *Proceedings of the 8th National Computer Security Conference*, pp. 141–144. 1985. (Ref: p. 35.)

[Bib77] K. J. Biba. Integrity considerations for secure computer systems. Technical Report TR-3153, MITRE Corporation, Bedford, Mass., 1977. (Ref: p. 34.)

[BL73] D. E. Bell and L. LaPadula. Secure computer systems: Mathematical foundations. Technical Report Mitre Report ESD-TR-73-278, MITRE Corporation, April 1973. (Ref: p. 24, 34.)

[Bla93] M. Blaze. A cryptographic file system for unix. In *Proceedings of the 1st ACM Conference on Computer and Communications Security (CCS)*, pp. 9–16. ACM Press, New York, NY, USA, 1993. (Ref: p. 41.)

[BMY02] J. Bacon, K. Moody, and W. Yao. A model of OASIS role-based access control and its support for active security. *ACM Transactions on Information and System Security*, 5(4):492–540, November 2002. (Ref: p. 38, 39, 177, 180.)

[BN89] D. F. C. Brewer and M. J. Nash. The chinese wall security policy. In *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 206–214. 1989. (Ref: p. 35.)

[BPB$^+$04] D. E. Bakken, R. Parameswaran, D. M. Blough, A. A. Franz, and T. J. Palmer. Data obfuscation: Anonymity and desensitization of usable data sets. *IEEE Security and Privacy*, 2(6):34–41, November/December 2004. (Ref: p. 72, 83.)

[BPWC90] J. L. Berger, J. Picciotto, J. P. L. Woodward, and P. T. Cummings. Compartmented mode workstation: Prototype highlights. *IEEE Transactions on Software Engineering*, 16(6):608–618, 1990. (Ref: p. 26.)

[BS04] J. Baliosian and J. Serrat. Finite state transducers for policy evaluation and conflict resolution. In *Proceedings of the 5th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY)*, pp. 250–260. 2004. (Ref: p. v, 119, 129, 130, 133, 139, 150, 162, 163, 168.)

[BVFS98] E. Bertino, S. C. Vimercati, E. Ferrari, and P. Samarati. Exception-based information flow control in object-oriented systems. *ACM Transactions on Information and System Secururity*, 1(1):26–65, 1998. (Ref: p. 35.)

[Car99] L. Cardelli. Semistructured computation. In *Proceedings of the 7th International Workshop on Database Programming Languages: Research Issues in Structured and Semistructured Database Programming*, pp. 1–16. September 1999. (Ref: p. 63.)

[cc:a] Common criteria: An introduction. Available at http://www.commoncriteriaportal.org/public/files/ccintroduction.pdf. (Ref: p. 175.)

[cc:b] Common criteria portal. Website at http://www.commoncriteriaportal.org. (Ref: p. 41, 175, 176.)

[cc:05a] The common criteria for information technology security evaluation: Security assurance requirements, August 2005. Available online at http://www.commoncriteriaportal.org/public/files/ccpart3v2.3.pdf. (Ref: p. 175.)

[cc:05b] The common criteria for information technology security evaluation: Security functional requirements, August 2005. Available online at http://www.commoncriteriaportal.org/public/files/ccpart2v2.3.pdf. (Ref: p. 175, 176.)

[CFG$^+$87] P. Cummings, D. Fullan, M. Goldstien, M. Gosse, J. Woodward, and J. Wynn. Compartmented model workstation: Results through prototyping. In *Proceedings of the 1987 IEEE Symposium of Security and Privacy*, pp. 27–29. April 1987. (Ref: p. 26.)

[CFMS94] S. Castano, M. G. Fugini, G. Martella, and P. Samarati. *Database Security*. ACM Press and Addison-Wesley Publishing Co., New York, NY, USA, 1994. (Ref: p. 32.)

[CFZA02] M. J. Covington, P. Fogla, Z. Zhan, and M. Ahamad. A context-aware security architecture for emerging applications. In *Proceedings of the 18th Annual Computer Security Applications Conference (ACSAC)*, pp. 249–260. 2002. (Ref: p. 39.)

[CG98] L. Cardelli and A. D. Gordon. Mobile ambients. In *Proceedings of the Proceedings of the First International Conference on Foundations of Software Science and Computation Structure*, pp. 140–155. 1998. (Ref: p. 42, 60, 63.)

[CG00] L. Cardelli and A. D. Gordon. Anytime, anywhere modal logics for mobile ambients. In *Proceedings of the 27th ACM Symposium on Principles of Programming Languages (POPL)*, pp. 365–377. 2000. (Ref: p. 42, 43, 60.)

[Cha02] D. Chalmers. *Contextual Mediation to Support Ubiquitous Computing*. Ph.D. thesis, Department of Computing, Imperial College London, 2002. (Ref: p. 17, 72.)

[Cha03] M. Chalmers. Seamful design and ubicomp infrastructure. In *Proceedings of the UbiComp 2003 At the Crossroads Workshop: The Interaction of HCI and Systems Issues in UbiComp*. 2003. (Ref: p. 17, 21.)

[CK00] G. Chen and D. Kotz. A survey of context-aware mobile computing research. Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, November 2000. (Ref: p. 15, 16, 17.)

[CLN03] J. Chomicki, J. Lobo, and S. Naqvi. Conflict resolution using logic programming. *IEEE Transactions on Knowledge and Data Engineering*, 15(1):245 – 250, Jan/Feb 2003. (Ref: p. 132.)

[CLS$^+$01] M. J. Covington, W. Long, S. Srinivasan, A. K. Dev, M. Ahmad, and G. Abowd. Securing context-aware applications using environmental roles. In *Proceedings of the 6th ACM Symposium on Access Controls Models and Technologies (SACMAT)*, pp. 10–20. 2001. (Ref: p. 39.)

[CM03] F. Cuppens and A. Miege. Modelling contexts in the or-bac model. In *Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC)*, p. 416. IEEE Computer Society, Washington, DC, USA, 2003. (Ref: p. 39, 177.)

[CMA00] M. J. Covington, M. J. Moyer, and M. Ahamad. Generalized role-based access control for securing future applications. In *Proceedings of the National Information Systems Security Conference (NISSC)*. October 2000. (Ref: p. 39.)

[CN02] M. Corner and B. D. Noble. Zero-interaction authentication. In *Proceedings of the 8th ACM Conference on Mobile Computing and Networking (MobiCom)*, pp. 1–11. 2002. (Ref: p. 41.)

[CN03] M. Corner and B. D. Noble. Protecting applications with transient authentication. In *Proceedings of the 2st Inernational Conference on Mobile Systems, Applications, and Services (MobiSys)*, pp. 57–70. 2003. (Ref: p. 41, 189.)

[CSD01] D. Chalmers, M. Sloman, and N. Dulay. Map adaptation for users of mobile systems. In *Proceedings of the 10th International World Wide Web Conference (WWW)*. 2001. (Ref: p. 72.)

[CSG^{+}03] V. Cahill, B. Shand, E. Gray, C. Bryce, N. Dimmock, A. Twigg, J. Bacon, C. English, W. Wagealla, S. Terzis, P. Nicon, G. di Marzo Serugendo, J.-M. Seigneur, M. Carbone, K. Krukow, C. Jensen, Y. Chen, and M. Nielsen. Using trust for secure collaboration in uncertain environments. *IEEE Pervasive Computing*, 2(3):52–61, August 2003. (Ref: p. 86.)

[CSRL01] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2001. (Ref: p. 105, 158.)

[CVN05] G. Castagna, J. Vitek, and F. Z. Nardelli. The seal calculus. *Information and Compuation*, 201(1):1 – 54, August 2005. (Ref: p. 42.)

[DA00] A. K. Dey and G. D. Abowd. Cybreminder: A context-aware system for supporting reminders. In *Proceedings of the 2nd international symposium on Handheld and Ubiquitous Computing (HUC)*, pp. 172–186. Springer-Verlag, London, UK, 2000. (Ref: p. 47.)

[Dat03] C. J. Date. *An Introduction to Database Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003. 8th Edition. (Ref: p. 33.)

[DBE^{+}04] N. Dimmock, A. Belokosztolszki, D. Eyers, J. Bacon, and K. Moody. Using trust and risk in role-based access control policies. In *The ninth ACM symposium on Access control models and technologies*, pp. 156–162. ACM Press, New York, NY, USA, 2004. (Ref: p. 177.)

[DBVC06] B. Dragovic, J. Baliosian, P. Vidales, and J. Crowcroft. Autonomic system for context-adaptive information security in ubiquitous computing environments, April 2006. Submitted for publication in Elsevier Journal on Pervasive and Mobile Computing. (Ref: p. v, 124, 189.)

[DC04] B. Dragovic and J. Crowcroft. Information exposure control through data manipulation for ubiquitous computing. In *Proceedings of the New Security Paradigms Workshop (NSPW)*, pp. 57–67. September 2004. (Ref: p. 2.)

[DD77] D. E. Denning and P. J. Denning. Certification of programs for secure information flow. *Communications of ACM*, 20(7):504–513, 1977. (Ref: p. 5, 23, 26.)

[DDLS01] N. Damianou, N. Dulay, E. Lupu, and M. Sloman. The Ponder policy specification language. In *Proceedings of the 2nd IEEE Workshop on Policies for Distributed Systems and Networks*, pp. 18–39. January 2001. (Ref: p. 124, 126, 127, 131.)

[Den76] D. E. Denning. A lattice model of secure information flow. In *Communications of the ACM*, pp. 236–243. May 1976. (Ref: p. 26, 31, 35, 98.)

[Dey01] A. K. Dey. Understanding and using context. *Personal and Ubiquitous Computing*, 5(1):4–7, 2001. (Ref: p. 15.)

[Dim03] N. Dimmock. How much is 'enough'? Risk in trust-based access control. In *IEEE Intl Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises: Enterprise Security*. 2003. (Ref: p. 86.)

[EAC98] G. Edjlali, A. Acharya, and V. Chaudhary. History-based access control for mobile code. In *Proceedings of the 5th ACM Conference on Computer and Communications Security (CCS)*, pp. 38–48. ACM Press, New York, NY, USA, 1998. (Ref: p. 33.)

[EMO$^+$94] J. Epstein, J. McHugh, H. Orman, R. Pascale, A. Marmor-Squires, B. Danner, C. Martin, M. Branstad, G. Denson, and D. Rothnie. A high assurance window system prototype. *Journal of Computer Security*, 2(2):159–190, 1994. (Ref: p. 26.)

[ER99] M. Egenhofer and A. Rodriguez. Relation algebras over containers and surfaces: An ontological study of a room space. *Spatial Cognition and Computation*, 1(2):155–180, 1999. (Ref: p. 42.)

[FG96] C. Fournet and G. Gonthier. The reflexive chemical abstract machine and the join-calculus. In *23rd ACM Symposium on Principles of Programming Languages*. 1996. (Ref: p. 42.)

[FK92] D. Ferraiolo and R. Kuhn. Role-based access control. In *Proceedings of the 15th NIST-NCSC National Computer Security Conference*, pp. 13–16. 1992. (Ref: p. 35, 36, 37, 177.)

[FKG$^+$02] R. Fredriksen, M. Kristiansen, B. A. Gran, K. Stølen, T. A. Opperud, and T. Dimitrakos. The coras framework for a model-based risk management process. In *Proceedings of the 21st Intlernational Conference on Computer Safety, Reliability and Security*, volume 2434 of *Lecture Notes in Computer Science*, pp. 94–105. 2002. (Ref: p. 88, 188.)

[FMSS03] G. Ferrari, C. Montagnero, L. Semini, and S. Semprini. The MOBadtl model and method to design network aware applications. Technical Report TR0308, Computer Science Dept., University of Pisa, 2003. (Ref: p. 42.)

[Fra83] L. J. Fraim. SCOMP: A solution to the multilevel security problem. *IEEE Computer*, 16(7):26–34, July 1983. (Ref: p. 25.)

[FSG$^+$01] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli. Proposed nist standard for role-based access control. *ACM Transactions on Information and Systems Security*, 4(3):224–274, 2001. (Ref: p. 36.)

[GG01] V. Gupta and S. Gupta. Securing the wireless internet. *IEEE Communications Magazine*, 39(12):68–74, 2001. (Ref: p. 169.)

[Gli83] V. D. Gligor. A note on the denial-of-service problem. In *Proceedings of IEEE Symposium on Security and Privacy*, pp. 5101–5111. 1983. (Ref: p. 23.)

[GMPT01] C. K. Georgiadis, I. Mavridis, G. Pangalos, and R. K. Thomas. Flexible team-based access control using contexts. In *Proceedings of the 6th ACM Symposium on Access Control Models and Technologies (SACMAT)*, pp. 21–27. ACM Press, New York, NY, USA, 2001. (Ref: p. 38, 39.)

[Gro] T. C. Group. Trusted computing group. Website at https://www.trustedcomputinggroup.org/home. (Ref: p. 25, 28, 29.)

[Gro05] T. C. Group. Whitepaper: Embedded systems and trusted computing security, September 2005. Available from https://www.trustedcomputinggroup.org/. (Ref: p. 29.)

[HHS$^+$99] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster. The anatomy of a context-aware application. In *Proceedings of the 5th annual ACM/IEEE Conference on Mobile Computing and Networking*, pp. 59–68. 1999. (Ref: p. 15, 51.)

[HI04] K. Henricksen and J. Indulska. A software engineering framework for context-aware pervasive computing. In *Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 77–86. IEEE Computer Society, Washington, DC, USA, 2004. (Ref: p. 16, 47.)

[HIR02] K. Henricksen, J. Indulska, and A. Rakotonirainy. Modeling context information in pervasive computing systems. In *Proceedings or the 1st International Conference on Pervasive Computing (PerCom)*, pp. 167–180. Springer-Verlag, 2002. (Ref: p. 16.)

[HL98] A. Heuer and A. Lubinski. Data reduction - an adaptation technique for mobile environments. In *Interactive Apllications of Mobile Computing (IMC'98)*. 1998. (Ref: p. 72.)

[HO03a] F. Hansen and V. Oleshchuk. Spatial role-based access control model for wireless networks. In *Proceedings of the IEEE Vehicular Technology Conference*. Orlando, USA, October 2003. (Ref: p. 39.)

[HO03b] F. Hansen and V. Oleshchuk. SRBAC: A spatial role-based access control model for mobile systems. In *Proceedings of the Nordic Workshop on Secure IT Systems (NORDSEC)*, pp. 129–141. Gjvik, Norway, October 2003. (Ref: p. 39.)

[HRU75] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman. On protection in operating systems. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, pp. 14–24. 1975. (Ref: p. 25, 32.)

[HRU76] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman. Protection in operating systems. *Communications of the ACM*, 19(8):461–471, 1976. (Ref: p. 25, 32.)

[HSK04] M. Hazas, J. Scott, and J. Krumm. Location-aware computing comes of age. *IEEE Computer*, 37(2):95–97, February 2004. (Ref: p. 15, 154.)

[Hub94] G. Huber. CMW introduction. *ACM Special Interest Group on Security, Audit and Control (SIGSAC) Review*, 12(4):6–10, October 1994. (Ref: p. 26.)

[HZ04] A. Hohl and A. Zugenmaier. Safeguarding personal data with DRM in pervasive computing. In *Proceedings of the 1st Workshop on Security and Privacy at the Conference on Pervasive Computing*, volume 780 of *The International Series in Engineering and Computer Science*. Springer, April 2004. (Ref: p. 28.)

[IBM01] IBM Research. Autonomic Computing: IBM's Perspective on the State of Information Technology, October 2001. (Ref: p. 14, 47, 82.)

[JBLG05] J. Joshi, E. Bertino, U. Latif, and A. Ghafoor. A generalized temporal role-based access control model. *IEEE Transactions on Knowledge and Data Engineering*, 17(1):4–23, 2005. (Ref: p. 38.)

[KA98] M. G. Kuhn and R. J. Anderson. Soft tempest: Hidden data transmission using electromagnetic emanations. In *Proceedings of the 2nd International Workshop on Information Hiding*, pp. 124–142. 1998. (Ref: p. 5, 74.)

[Kar87] P. A. Karger. Limiting the damage potential of discretionary trojan horses. In *Proceedings of the 1987 IEEE Symposium on Security and Privacy*, pp. 32–37. 1987. (Ref: p. 35.)

[KC03] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003. (Ref: p. 14.)

[KFJ03] L. Kagal, T. Finin, and A. Joshi. A policy language for a pervasive computing environment. In *Proceedings of the 4th International IEEE Workshop on Policies for Distributed Systems and Networks (POLICY)*, pp. 63–76. 2003. (Ref: p. 41, 126.)

[KK02] L. Korba and S. Kenny. Towards meeting the privacy challenge: Adapting DRM. In *Proceedings of the Digital Rights Management Workshop*, pp. 118–136. 2002. (Ref: p. 28.)

[KM93] M. H. Kang and I. S. Moskowitz. A pump for rapid, reliable, secure communication. In *Proceedings of the 1st ACM Conference on Computer and Communications Security (CCS)*, pp. 119–129. ACM Press, New York, NY, USA, 1993. (Ref: p. 26.)

[Kni04] F. H. Knight. *Risk, Uncertainty and Profit.* Houghton Mifflin Company, September 2004. (Ref: p. 86.)

[Kuh05] M. G. Kuhn. Security limits for compromising emanations. In *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, pp. 265–279. 2005. (Ref: p. 5, 74.)

[Lam74] B. W. Lampson. Protection. *ACM SIGOPS Operating Systems Review*, 8(1):18–24, 1974. (Ref: p. 32.)

[Lan02] M. Langheinrich. A privacy awareness system for ubiquitous computing environments. In *In Proceedings of the 4th International Conference on Ubiquitous Computing (UbiComp)*, pp. 237–245. 2002. (Ref: p. 28.)

[LCC$^+$75] R. Levin, E. Cohen, W. Corwin, F. Pollack, and W. Wulf. Policy/mechanism separation in hydra. In *Proceedings of the 5th ACM Symposium on Operating Systems Principles (SOSP)*, pp. 132–140. ACM Press, New York, NY, USA, 1975. (Ref: p. 47.)

[Lev95] N. G. Leveson. *Software: System Safety and Computers*, chapter 9. Addison-Wesley, 1995. (Ref: p. 86.)

[LNSV02] S. Lange, A. Nonnengart, C. Stuble, and R. Vogt. Discretionary information flow control (mu) protection profile, September 2002. Available online at http://www.commoncriteriaportal.org/public/files/ppfiles/pp0008be.pdf. (Ref: p. 176.)

[LY02] K. Lyytinen and Y. Yoo. Issues and challenges in ubiquitous computing. *Communications of the ACM*, 45(12):62–65, December 2002. (Ref: p. 14.)

[MCWG04] A. Mas-Colell, M. D. Whinston, and J. R. Green. *Microeconomic Theory*. Oxford University Press Inc, USA, 2004. ISBN: 0195102681. (Ref: p. 80, 85.)

[MESW01] B. Moore, E. Ellesson, J. Strassner, and A. Westerinen. Policy core information model. RFC(3060), IETF, February 2001. (Ref: p. 125.)

[MG04] M.Chalmers and A. Galani. Seamful interweaving: heterogeneity in the theory and design of interactive systems. In *Proceedings of the 2004 conference on Designing interactive systems (DIS)*, pp. 243–252. ACM Press, New York, NY, USA, 2004. (Ref: p. 17, 21.)

[Mica] S. Microsystems. J2se 1.5.0 security. Website at http://java.sun.com/j2se/1.5.0/docs/guide/security/index.html. (Ref: p. 33.)

[Micb] S. Microsystems. Java. Website at http://java.sun.com. (Ref: p. 124.)

[Mil63] J. S. Mill. *Utilitarianism.* History of Economic Thought Books. McMaster University Archive for the History of Economic Thought, 1863. Available at http://ideas.repec.org/b/hay/hetboo/mill1863.html. (Ref: p. 80.)

[Mil99] R. Milner. *Communicating and Mobile Systems: The Pi Calculus.*. Cambridge University Press, 1999. (Ref: p. 42.)

[ML97] A. C. Myers and B. Liskov. A decentralized model for information flow control. In *Proceedings of the 16th ACM Symposium on Operating Systems Principles (SOSP)*, pp. 129–142. ACM Press, New York, NY, USA, 1997. (Ref: p. 26.)

[MMN90] C. D. McCollum, J. R. Messing, and L. Notargiacomo. Beyond the pale of MAC and DAC-defining new forms of access control. In *Proceedings of the 1990 IEEE Symposium on Security and Privacy*, pp. 190–200. 1990. (Ref: p. 28.)

[Mos04] G. K. Mostefaoui. *Towards a Conceptual and Software Framework for Integrating Context-Based Security in Pervasive Environments*. Ph.D. thesis, Department of Informatics University of Fribourg and Lip 6 University Pierre et Marie Curie (Paris VI), 2004. (Ref: p. 15.)

[MPB03] M. Mont, S. Pearson, and P. Bramhall. Towards accountable management of identity and privacy: Sticky policies and enforceable tracing services. Technical Report HPL-2003-49, HP Laboratories, 2003. (Ref: p. 28.)

[MPR04] G. K. Mostefaoui and J. Pasquier-Rocha. Context-aware computing: A guide for the pervasive compting community. In *Proceedings of the IEEE/ACS International Conference on Pervasive Services (ICPS)*, pp. 39–48. July 2004. (Ref: p. 16.)

[Mye99] A. C. Myers. Jflow: practical mostly-static information flow control. In *Proceedings of the 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*, pp. 228–241. ACM Press, New York, NY, USA, 1999. (Ref: p. 26.)

[NWET04] P. Nixon, W. Wagealla, C. English, and S. Terzis. *Privacy, Security, and Trust Issues in Smart Environments*, chapter Smart Environments: Technology, Protocols and Applications, pp. 220–240. Wiley, October 2004. (Ref: p. 19, 22.)

[oct] The operationally critical threat, asset and vulnerability evaluation (octave), carnegie mellon software engineering institute. Web site at http://www.cert.org/octave. (Ref: p. 88, 188.)

[oD85] U. S. D. of Defense. Trusted computer system evaluation criteria. Technical Report 5200.28, US Department of Defense, 1985. (Ref: p. 26.)

[oma03] OMA user agent profile (UAProf), May 2003. Website at http://www.openmobilealliance.org/. (Ref: p. 142.)

[OSM00] S. Osborn, R. Sandhu, and Q. Munawer. Configuring role-based access control to enforce mandatory and discretionary access control policies. *ACM Transactions on Information and Systems Security*, 3(2):85–106, 2000. (Ref: p. 37.)

[Pas98] J. Pascoe. Adding generic contextual capabilities to wearable computers. In *ISWC '98: Proceedings of the 2nd IEEE International Symposium on Wearable Computers*, p. 92. IEEE Computer Society, Washington, DC, USA, 1998. (Ref: p. 15, 16, 17.)

[PKKJ04] A. Patwardhan, V. Korolev, L. Kagal, and A. Joshi. Enforcing policies in pervasive environments. In *Proceedings of the International Conference on Mobile and Ubiquitous Systems: Networking and Services (Mobiquitous)*, pp. 299–308. 2004. (Ref: p. 41.)

[Poi] Pointsec. Company web site. Available at http://www.pointsec.com. (Ref: p. 41.)

[Pon] Ponder. Policy research group. Website at http://www-dse.doc.ic.ac.uk/Research/policies/ponder.shtml. (Ref: p. 124, 138.)

[Pro00] N. Provos. Encrypting virtual memory. In *Proceedings of the USENIX Security Symposium*, pp. 35–44. 2000. (Ref: p. 41.)

[PS02a] J. Park and R. Sandhu. Originator control in usage control. In *Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY)*, p. 60. IEEE Computer Society, Washington, DC, USA, 2002. (Ref: p. 28, 40.)

[PS02b] J. Park and R. Sandhu. Towards usage control models: beyond traditional access control. In *Proceedings of the 7th ACM Symposium on Access Control Models and Technologies (SACMAT)*, pp. 57–64. ACM Press, New York, NY, USA, 2002. (Ref: p. 39, 40.)

[PSS00] J. Park, R. S. Sandhu, and J. Schifalacqua. Security architectures for controlled digital information dissemination. In *Proceedings of the 16th Annual Computer Security Applications Conference (ACSAC)*, p. 224. 2000. (Ref: p. 23, 27, 28, 30.)

[PW98] B. Pomeroy and S. Wiseman. Private desktops and shared store. In *Proceedings of the 14th Annual Computer Security Applications Conference (ACSAC)*, p. 190. December 1998. (Ref: p. 26.)

[RBKW91] F. Rabitti, E. Bertino, W. Kim, and D. Woelk. A model of authorization for next-generation database systems. *ACM Transactions on Database Systems*, 16(1):88–131, 1991. (Ref: p. 33.)

[RE00] A. Rodriguez and M. Egenhofer. A comparison of inferences about containers and surfaces in small-scale and large-scale spaces. *Journal of Visual Languages and Computing*, 11(6):639–662, 2000. (Ref: p. 42.)

[RHR$^+$01] M. Roman, C. K. Hess, A. Ranganathan, P. Madhavarapu, B. Borthakur, P. Viswanathan, R. Cerqueira, R. H. Campbell, and M. D. Mickunas. GaiaOS: An infrastructure for active spaces. Technical Report UIUCDCS-R-2001-2224, University of Illinois at Urbana-Champaign, Champaign, IL, USA, 2001. (Ref: p. 39.)

[RS97] E. Roche and Y. Schabes. Finite-state language processing. Technical report, MIT Press, Cambridge, Massachusetts., 1997. (Ref: p. 129.)

[RSC92] J. Richardson, P. Schwarz, and L.-F. Cabrera. CACL: efficient fine-grained protection for objects. In *Proceedings of the Conference on Object-Oriented*

Programming Systems, Languages, and Applications (OOPSLA), pp. 263–275. ACM Press, New York, NY, USA, 1992. (Ref: p. 33.)

[San92] R. S. Sandhu. The typed access matrix model. In *Proceedings of the 1992 IEEE Symposium on Security and Privacy*, pp. 122–136. IEEE Computer Society, Washington, DC, USA, 1992. (Ref: p. 32.)

[San93] R. S. Sandhu. Lattice-based access control models. *IEEE Computer*, 26(11):9–19, November 1993. (Ref: p. 31, 34, 35.)

[San03] R. Sandhu. Good-enough security: Toward a pragmatic business-driven discipline. *IEEE Internet Computing*, 7(1):66–68, 2003. (Ref: p. 7, 184.)

[Sas03] M. A. Sasse. Computer security: Anatomy of a usability disaster, and a plan for recovery. In *Proceedings of the 2003 Workshop on Human-Computer Interaction and Security Systems (CHI)*, pp. 40–46. 2003. (Ref: p. 7.)

[Sat02] M. Satynarayanan. A catalyst for mobile and ubiquitous computing. *IEEE Pervasive Computing Magazine*, 1(1):2–5, January 2002. (Ref: p. 14.)

[SAW94] B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, pp. 85–90. IEEE Computer Society Press, December 1994. (Ref: p. 15, 16.)

[SBG99] A. Schmidt, M. Beigl, and H.-W. Gellersen. There is more to context than location. *Computers and Graphics*, 23(6):893–901, 1999. (Ref: p. 15.)

[SBW95] O. Sibert, D. Bernstein, and D. V. Wie. The digibox: A self-protecting container for information commerce. In *Proceedings of the 1st USENIX Workshop on Electronic Commerce*, pp. 171–183. 1995. (Ref: p. 30, 40.)

[SC03] F. Stajano and J. Crowcroft. The butt of the iceberg: hidden security problems of ubiquitous systems. pp. 91–101, 2003. (Ref: p. 2, 19, 21.)

[SCFY96] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *IEEE Computer Magazine*, 29(2):38–47, 1996. (Ref: p. 35, 36, 37, 177, 182.)

[Sch95a] W. N. Schilit. *A System Architecture for Context Aware Mobile Computing*. Ph.D. thesis, Columbia University, 1995. (Ref: p. 15.)

[Sch95b] B. Schneier. *Applied cryptography (2nd ed.): protocols, algorithms, and source code in C*. John Wiley & Sons, Inc., New York, NY, USA, 1995. (Ref: p. 22.)

[SCR$^+$99] S. Shekhar, S. Chawla, S. Ravada, S. Fetterer, and C. Liu. Spatial databases: Accomplishments and research needs. *IEEE Trans on Knowledge and Data Engineering*, 11(1):45–55, 1999. (Ref: p. 42.)

[SD92] H. Shen and P. Dewan. Access control for collaborative environments. In *Proceedings of the 1992 ACM Conference on Computer-Supported Cooperative Work*, pp. 51–58. ACM Press, New York, NY, USA, 1992. (Ref: p. 33.)

[SLP04] T. Strang and C. Linnhoff-Popien. A context modeling survey. In *Proceedings of the UbiComp Workshop on Advanced Context Modelling, Reasoning and Management*. 2004. (Ref: p. 16, 17.)

[SNC02] G. Sampemane, P. Naldurg, and R. H. Campbell. Access control for active spaces. In *Proceedings of the 18th Annual Computer Security Applications Conference (ACSAC)*, p. 343. IEEE Computer Society, Washington, DC, USA, 2002. (Ref: p. 39.)

[SP03] R. Sandhu and J. Park. Usage control: A vision for next generation access control. In *Proceedings of the 2nd International Workshop on Mathematical Methods, Models and Architectures for Computer Networks Security*, pp. 17–31. 2003. (Ref: p. 39, 40.)

[SSF99] J. S. Shapiro, J. M. Smith, and D. J. Farber. EROS: a fast capability system. In *Proceedings of the 17th ACM Symposium on Operating Systems Principles (SOSP)*, pp. 170–185. ACM Press, New York, NY, USA, 1999. (Ref: p. 33.)

[ST94] R. S. Sandhu and R. K. Thomas. Conceptual foundations for a model of task-based authorizations. In *Proceedings of the 7th IEEE Computer Security Foundations Workshop*, pp. 66–79. 1994. (Ref: p. 38.)

[Sta02] F. Stajano. *Security for Ubiquitous Computing*. John Wiley and Sons, February 2002. (Ref: p. 19, 22, 23.)

[Sto81] A. Stoughton. Access flow: A protection model which integrates access control and information flow. In *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 9–18. 1981. (Ref: p. 31, 35.)

[TC03] D. S. Tan and M. Czerwinski. Information voyeurism: Social impact of physically large displays on information privacy. In *Proceedings of the 2003 Conference on Human Factors in Computing Systems (CHI)*, pp. 748–749. April 2003. (Ref: p. 5, 74, 154, 161.)

[Ten00] D. Tennenhouse. Proactive computing. *Communications of ACM*, 43(5):43–50, 2000. (Ref: p. 14.)

[Tho97] R. K. Thomas. Team-based access control (TMAC): a primitive for applying role-based access controls in collaborative environments. In *Proceedings of the 2nd ACM Workshop on Role-Based Access Control*, pp. 13–19. ACM Press, New York, NY, USA, 1997. (Ref: p. 38.)

[TS93] R. K. Thomas and R. S. Sandhu. Towards a task-based paradigm for flexible and adaptable access control in distributed applications. In *Proceedings on the 1992-1993 Workshop on New Security Paradigms (NSPW)*, pp. 138–142. ACM Press, New York, NY, USA, 1993. (Ref: p. 38.)

[TS97] R. K. Thomas and R. S. Sandhu. Task-based authorization controls (TBAC): A family of models for active and enterprise-oriented autorization management. In *DBSec*, pp. 166–181. 1997. (Ref: p. 38, 177, 178, 180.)

[TS04a] R. K. Thomas and R. Sandhu. Models, protocols and architectures for secure pervasive computing: Challenges and research directions. In *2nd IEEE Annual Conference on Pervasive Computing and Communications Workshops (PerCom)*, pp. 164–170. 2004. (Ref: p. 19, 20.)

[TS04b] R. K. Thomas and R. S. Sandhu. Towards a multi-dimensional characterization of dissemination control. pp. 197–200. 2004. (Ref: p. 27.)

[VBS$^+$05] P. Vidales, J. Baliosian, J. Serrat, G. Mapp, F. Stajano, and A. Hopper. Autonomic system for mobility support in 4g networks. *IEEE Journal on Special Areas in Communications*, 23(12):2288–2304, December 2005. (Ref: p. v, 124, 189.)

[VGC01] P. Viswanathan, B. Gill, and R. Campbell. Security architecture in Gaia. Technical Report UIUCDCS-R-2001-2215, University of Illinois at Urbana-Champaign, Champaign, IL, USA, 2001. (Ref: p. 39.)

[VMK04] J. Von Neumann, O. Morgenstern, and H. W. Kuhn. *Theory of Games and Economic Behavior*. Princeton University Press, 2004. ISBN: 0691119937. (Ref: p. 80.)

[vNG01] G. van Noord and D. Gerdemann. Finite state transducers with predicates and identities. *Grammars*, 4(3):263 – 286, December 2001. (Ref: p. 129.)

[w3c04] Composite capability/preference profiles (cc/pp): Structure and vocabularies 1.0, January 2004. Website at http://www.w3.org/TR/CCPP-struct-vocab/. (Ref: p. 142.)

[Wei91] M. Weiser. The computer for the 21st century. *Scientific American*, 265(3):94–104, September 1991. (Ref: p. 1, 14, 79, 82.)

[WOR$^+$74] K. G. Walter, W. Ogden, W. C. Rounds, F. T. Bradshaw, S. R. Ames, and D. G. Sumaway. Primitive models for computer security. Technical Report ESD-TR-4-117, Case Western Reserve University, 1974. (Ref: p. 35.)

[XML04] XML. eXtensible Markup Language, 2004. Website at http://www.w3.org/XML. (Ref: p. 148.)

[YBAG00] J. Yan, A. Blackwell, R. Anderson, and A. Grant. The memorability and security of passwords – some empirical results. Technical Report UCAM-CL-TR-500, The Computer Laboratory, University of Cambridge, September 2000. (Ref: p. 4.)

[YBAG04] J. Yan, A. Blackwell, R. Anderson, and A. Grant. Password memorability and security: Empirical results. *IEEE Security and Privacy*, 2(5):25–31, 2004. (Ref: p. 4.)

[Yee04] K.-P. Yee. Aligning security and usability. *IEEE Security and Privacy*, 2(5):48–55, 2004. (Ref: p. 7.)

[Zal05] M. Zalewski. Cracking safes with thermal imaging. Online, 2005. Available online at http://lcamtuf.coredump.cx/tsafe. (Ref: p. 6, 161.)

[Zim96] T. G. Zimmerman. Personal area networks: near-field intrabody communication. *IBM Systems Journal*, 35(3-4):609–617, 1996. (Ref: p. 66, 83.)

[ZP04] G. Zhang and M. Parashar. Context-aware dynamic access control for pervasive applications. In *Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS)*, pp. 219–225. 2004. (Ref: p. 39.)

[ZZT05] L. Zhuang, F. Zhou, and J. Tygar. Keyboard acoustic emanations revisited. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS)*. ACM, 2005. (Ref: p. 6, 161.)