The Trust-Domains Guide

A Guide to Identifying, Modelling, and Establishing Trust Domains

Compiled and edited by
Stephen Crane and Philipp Reinecke

With contributions by Gabrielle Anderson, Myrto Arapinis, Adrian Baldwin, Matthew Collinson, Stephen Crane, Nalin Asanka Gamagedara Arachchilage, Martin Gill, Gina Kounga, Dirk Kuhlmann, John Mace, Andrew Martin, Cornelius Namiluko, Mihai Ordean, David Pym, Philipp Reinecke, Eike Ritter, Mark Ryan, and Adrian Shaw

The Trust-Domains Guide

The Trust-Domains Guide

A Guide to Identifying, Modelling, and Establishing Trust Domains in Collaborations

Compiled and edited by

Stephen Crane and Philipp Reinecke

with contributions by

Gabrielle Anderson, Myrto Arapinis, Adrian Baldwin, Matthew Collinson, Stephen Crane, Nalin Asanka Gamagedara Arachchilage, Martin Gill, Gina Kounga, Dirk Kuhlmann, John Mace, Andrew Martin, Cornelius Namiluko, Mihai Ordean, David Pym, Philipp Reinecke, Eike Ritter, Mark Ryan and Adrian Shaw.

March 2015

Gabrielle Anderson
University of Aberdeen/University College London

Myrto Arapinis
University of Birmingham

Adrian Baldwin
HP Labs Bristol

Matthew Collinson
University of Aberdeen

Stephen Crane
HP Labs Bristol

Nalin Asanka Gamagedara Arachchilage
University of Oxford

Martin Gill
Perpetuity Research

Gina Kounga
University of Oxford

Dirk Kuhlmann
HP Labs Bristol

John Mace
HP Labs Bristol/Newcastle University

Andrew Martin
University of Oxford

Cornelius Namiluko
University of Oxford

Mihai Ordean
University of Oxford

David Pym
University of Aberdeen/University College London

Philipp Reinecke
HP Labs Bristol

Eike Ritter
University of Birmingham

Mark Ryan
University of Birmingham

Adrian Shaw
HP Labs Bristol

Foreword

We live in a world where the physical and digital have become entwined. From the sea of sensors that monitor almost everything we can imagine monitoring, to the computers that collect, store and process all the information, to the clever analytic software that draws conclusions, every aspect of our lives is mediated in some way by the digital. And whether it is food being delivered to the local store, planes landing safely, the scanner in the hospital imaging correctly, or the financial markets not collapsing, increasingly we are completely dependent on all this technology behaving in some reliable way.

The pace of technology change shows no signs of slowing, and the next ten years are as likely to be as transformative as the last ten. Yet with each new wave of technology we do not completely get rid of what is already there, building an ever increasingly complex world where our dependence on technology increases, but worryingly where we understand less and less about how it all works.

And whether it is criminal, activist or government activity, hardly a day goes by without stories of cyberattack in some form, increasing our concerns as to whether we really can depend on all these online systems.

This mix of increasing complexity, lack of understanding, and adversaries who we believe might be smarter than us, gives rise to a great deal of anxiety. Anxiety that is easily compounded by movie scripts that paint hackers and government agencies as capable of completely controlling our digital world.

We desperately need the concepts that will help us build and analyse systems in ways that reduce complexity, increase understanding and reassure us that attackers can only disrupt those systems in limited ways. We would like to be able to trust all this technology in some way.

Yet trust is one of those slippery terms in information security that we all recognise and agree is important, but we have real difficulty nailing down exactly what trust means, how we might measure it and how we can use our

understanding of trust to greatest effect to better protect the assets that we value most highly.

It does seem clear that today the need for trust is present in almost every aspect of life, and that this is reflected in the way we work and how we organise our business practices. As the world begins to experience an unprecedented level of highly targeted, sophisticated attacks on our corporate networks and systems, knowing who, what and when to trust is critically important. Yet 'trust' is used to cover a wide range of concepts. When we consider computing systems, some argue that the concept of trust is irrelevant, perhaps even dangerous, that systems should be designed around guaranteed functionality, and that we should avoid security that 'cannot be measured'. Others claim that it is impossible to ignore trust, and that trust is a fundamental component of any system that involves humans in a decision making or critical role. It is fairly obvious they are not talking about the same thing.

Trust, when used in the context of trusted or trustworthy computing platforms, is very much about technological guarantees. And in this sense these technologies perform well, since their performance can be measured, metrics established, and a quantitative statement about trust made with reliability. But every system has its Achilles heel, and for many it is the 'human in the loop' that raises questions and introduces doubt. Unfortunately, we are a long way from adequately connecting the dots from low level system measurements, up through the technology stack, to how users think about the tasks they are carrying out and either follow or circumvent the rules and norms others are expecting them to follow.

Throughout, a question that remains to a greater extent unanswered, and one that research on trust can address, is whether a dependency on trust is good – or bad. Can trust help us achieve a higher level of security or should we make every attempt to rid trust from our vocabulary?

One reason why trust is so troublesome is that humans are unpredictable. Nevertheless, humans do respond well to their environment. They identify and react to changing priorities, and offer invaluable flexibility. Being able to recognise situations where human intervention is valuable and positive, or even facilitate the formation of such situations, is a powerful capability.

It is a common expectation that people will behave in predictable ways thus providing organisations with the reassurance that policies and procedures will work as anticipated, an important consideration at a time when cyber security is at the top of every CIO's priorities list. Predicting behaviour, though, is not an exact science. And this means that within an organisation knowing when to

let go and when to exert tight control is seen as much of an art as a science. But by understanding the range of possible outcomes that may arise when relaxing or reinforcing practices, we move towards 'informed choice'. And with this new knowledge we have a better understanding of whether existing security techniques can help us or if new concepts are needed.

The Trust Domains project sheds light on all of this and develops our understanding of the role that trust plays in a corporate setting, where rules and roles dominate the management of risk, but where rule-breaking is not uncommon.

As the world becomes more complex and experiences new and highly targeted forms of cyber-attack, research that advances our understanding of where and how we should direct our security efforts is an imperative, enabling organisations to be better positioned to survive what many experts are saying is the inevitability of either system failures of, or major attacks on, our online infrastructure with severe consequences for all of us.

Martin Sadler, OBE
Vice President and Director,
Security and Manageability
HP Labs

Contents

Introduction

Trust is a prerequisite for collaborative human action and efficient co-operation. This fact is of particular importance when collaboration is mediated by information technology, where new domains of interaction are constantly created within, between, across, and orthogonal to predefined organisational processes, boundaries, and solutions, leading to challenges for the analysis, design, deployment and assessment of collaborative IT systems.

Trust Domains are an emergent concept for addressing these challenges. Traditionally, IT systems have been described in terms of classical attributes such as functional composition, reliability, and security. This, however, is too narrow a view for determining the level of trust that all participating entities can have in each other and in the underlying technical infrastructure. Therefore, the trust domains concept also looks at co-factors such as incentive structures, mutual expectations, reliance, and assurance. The goal is to develop novel approaches that explicitly address the issue of reasonable user expectations when describing and designing collaborative information systems.

This book presents the results of the Trust Domains project, a three-year collaboration between HP Labs, Perpetuity Research Limited, Oxford University, Birmingham University, Aberdeen University, and University College London, funded by the UK Technology Strategy Board (TSB).[1] The book is aimed at the professional working in the security field, both from academia and from industry, who wants to understand the basics of trust domains, the challenges they pose, and the ways in which the Trust Domains project addressed them.

[1] *Trust Domains: A Framework for Modelling and Designing E-Service Infrastructures for Controlled Sharing of Information,* TSB Project Number TP/400206

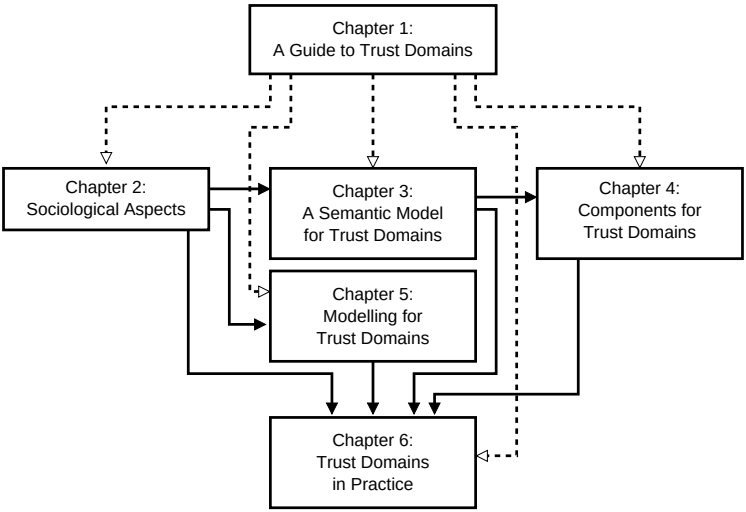


Figure 1: General Structure of the Book.

The general structure of the book is illustrated in Figure 1:

In Chapter 1 we set out with a high-level guide to the insights gained from the Trust Domains project. We discuss requirements for and aspects of trust domains from a criminological and sociological perspective, identifying requirements for establishing trust and factors that undermine trust. We then consider how modelling can help with establishing and evaluating trust domains. Finally, we describe the application of these principles in practice. Each of these areas is then discussed in more detail in the following chapters.

In Chapter 2 we approach the question of trust domains from a sociological and criminological perspective. We discuss the definition of trust and study frameworks for criminological studies. We then present insights on the behaviour of offenders and of the defenders of trust domains and identify main requirements and problems with trust domains. These insights guide our further studies throughout the book.

In Chapter 3 we describe a semantic model for trust domains. This model is built upon standard semantic modelling techniques and simplifies communication about the requirements for and the actual implementation of trust domains at the social, services, and infrastructure layers. In particular, the model

enables us to identify the technical components required to support trust domains.

Chapter 4 then provides a survey of components for trust domains. The survey is followed by the presentation of a technology referred to as a hypervisor that supports monitoring of the system state of client devices.

Chapter 5 addresses the issue of modelling for trust domains. In order to understand the operational properties of a trust domain, and to support the designer and the users we apply several modelling approaches. These are complementary to each other, in that they provide verification of security properties and evaluation of security and performance, both at design-time and at run-time.

The application of the insights presented in Chapters 1–5 is discussed in Chapter 6. Here, we first present an approach for communicating trust domains principles at the sociological level. We then discuss the application of design-time modelling in a particular scenario where there is no specialised trust-domains support available. Finally, we present two systems that specifically support trust domains in very different application scenarios: ConfiChair is a tool for managing the review process of conferences. Operating under strict assumptions, it is possible to prove that the protocol used in ConfiChair is secure. The Trustworthy Collaboration System, on the other hand, supports simple and trustworthy document sharing under loose constraints. While the system is not provably secure, it supports the user in making good decisions for establishing and maintaining trust.

The work presented in this book is largely based on the deliverables produced by the project and the publications of the academic partners. As the presentation here is given in summarised form, throughout the book we refer the reader to these sources for the details.

Acknowledgments

We would like to thank Simon Arnell, Sven Böhmert, Ed Cullen, Matt Keeble, Charlotte Howell, Brian Monahan, Neil Passingham, Joshua Phillips, Anbang Ruan, and all of our interviewees as well as the participants of the International Workshop on Cloud Technologies and Trust Domains. Furthermore, we would like to express our special gratitude to our TSB Project Monitoring Officer Alan Bennett.

Chapter One

A Guide to Trust Domains

This chapter provides a broadly accessible discussion of the key findings presented in this book. The chapter, while also serving as an introduction to the more detailed chapters that follow later, is intended primarily to be an easy-to-follow stand-alone guide to the many important insights gained in the course of the project. Thus, we aim to give a broad overview, and refer the reader to later chapters for the fine details.

This chapter is structured as follows: We start by first introducing the concept of Trust Domains (Section 1.1) and then describe issues of trust arising from empirical studies (Section 1.2). In Section 1.3 we discuss how modelling helps establish trust. Finally, in Section 1.4 we describe two applications of trust-domains principles in practice.

Example: Mergers and Acquisitions Process

Throughout the chapter we employ a typical workflow to illustrate the concepts. This workflow, referred to as the Mergers and Acquisitions (M&A) workflow, is follwed when companies merge with one-another or are acquired. In this example, company A intends to acquire and merge with a company B. Company A draws on the services of a bank to support this process, and also needs the services of lawyers and external analysts. The bank has the task of contacting all parties, analysing data and preparing reports, and finalising the contract. The workflow is as follows: First, the bank contacts both companies and requests necessary data. This data is then analysed, both in-house and by external analysts. Based on the reports, lawyers are contacted to draft a

contract, and the contract is then forwarded to the companies, in order to be signed.

While this workflow is intentionally simplified for the example, it captures the fundamental problems that can be addressed by concepts from Trust Domains: Individuals from distinct organisations that do not necessarily trust each other on all accounts need to cooperate in order to complete a process, and in all steps of the process sensitive data needs to be protected.

1.1 Definitions: What is a Trust Domain?

One of the insights from the Trust Domains project is that the term *Trust Domain* may be defined in various ways. Each definition focusses on different aspects of the concept, and hence leads to different ways of thinking about and different approaches to solving problems involving Trust Domains. As any attempt to merge the definitions would necessarily lose some of the clarity of each individual one, we first present them side by side, and then discuss their relation to each other in detail.

Definition 1: *A Trust Domain is where information that is designated to be in some way privileged is subjected to access controls or similar type of protection.* [GC13]

Definition 2: *We consider a Trust Domain to be the state and processes that allow resources to be shared between entities that are members of the domain, and where these entities have an expectation of, and exhibit, shared and predictable behaviour to protect the resources.* [d3.13, KM12]

Definition 3: *A grouping of two or more entities that share the same level of expectation regarding security of information that they wish to exchange with one another.* [CG12, MKBN12]

Definition 4: *In systems of interacting agents, an individual or group of agents may establish a part of the system, or a collection of agents within the system, that it trusts. Similarly, a system's designer or manager might establish a collection of parts of the system such that, within any given part, the agents trust one another. We shall refer to such a part of the system, or such a collection of agents, as a 'trust domain'.* [ACP13b]

These definitions have in common that in all of them a group of entities or actors comprising the trust domain is distinguished from other entities. This distinction is usually enabled or supported by technical means and enables the

members of the Trust Domain to collaborate. Furthermore, according to these definitions a Trust Domain is characterised by the assumption of homogeneous behaviour within the domain. This assumption is made explicit by Definitions 2 and 3, and is implicitly contained in Definition 4 by the trust that members of the Trust Domain place in each other, and in the fact that the members of the domain have access to privileged information in Definition 1.

On the other hand, the definitions differ with respect to which aspects of the concept they focus on. In particular, we may identify differences in what each definition considers to be the *defining feature of the concept*, and in how the definition enables the *distinction between the inside and the outside* of a Trust Domain.

According to Definition 1, a Trust Domain protects privileged information by some sort of controls. The *defining feature* is thus the existence of privileged information and its protection by such controls. Definition 2 also considers the existence of mechanisms for controlling access in order to define the concept, but puts stronger emphasis on the fact that these controls enable sharing. That is, Definition 1 focusses on the *protection* of information from access by non-members, whereas Definition 2 focusses on the *sharing* of information between members of the domain, which becomes possible by the fact that protection is guaranteed. Both definitions thus focus on the controls that establish the Trust Domain. Additionally, Definition 2 emphasises that the members of the domain share expectations concerning their behaviour with respect to protecting the shared assets, and exhibit the behaviour expected by others. In contrast, Definitions 3 and 4 employ a perspective that emphasises the importance of the entities of the domain and their expectations, rather than of the controls. Definition 3 sees Trust Domain as a group of entities sharing a level of expectation on the security of data they share with each other. Definition 4 is similar, in that here the fact that the entities trust each other is the defining feature. Both definitions emphasise that members of the domain make assumptions about other members that potentially influence their behaviour. In particular – and joining up Definitions 3 and 4 to Definitions 1 and 2 –, shared expectations and trust in other members make enabling access to privileged information feasible, and allow members to share information that they would otherwise not be willing to share. On the other hand, the controls postulated by Definitions 1 and 2 support, and often enable, the shared expectations and the trust that Definitions 3 and 4 consider to be the defining features of Trust Domains.

We now turn our attention to the membership criterion, that, according to the definitions, determines the *inside and outside* of a Trust Domain, i.e. the distinction between entities that are members of the domain and entities that are not. In Definition 1, access controls define this boundary: Anyone that is granted access to a Trust Domain by the access controls is a member of the domain. This definition thus emphasises the importance of controls in practice. Definition 2 adds the notion of expected and exhibited behaviour: Entities may become members of the domain by gaining access, but they should also share the expectations of other members regarding the resources that are available in the domain, and fulfil these expectations. In contrast to the first two definitions, with Definitions 3 and 4 membership of the domain is only determined by properties of the entities, and not by controls. With Definition 3, an entity can be considered to be inside a Trust Domain if it expects the same level of security (with respect to shared information) as other members of the domain. With Definition 4, all entities that trust each other form a Trust Domain. Note that only the membership criterion of Definition 2 combines expectations, fulfilment of expectations, and access controls; all other definitions focus on either expectations or controls, and none of them explicitly requires fulfilment of expectations.

Example: Mergers and Acquisitions Process

In the Mergers and Acquisitions (M&A) example, large amounts of privileged information exist. For instance, intellectual property of the companies may need to be exchanged as well as personally identifiable information (PII) and internal business performance data. Furthermore, even the fact that an M&A process takes place is usually privileged, in that it may affect e.g the stock price of the companies involved. All of this information is usually protected by access control mechanisms, which involve physical, organisational, and technical means. Therefore, considering the location of access control mechanisms, we may easily identify many Trust Domains according to Definition 1 in this scenario: Each organisation (e.g. a company or the bank) constitutes a Trust Domain, and the whole process is also a Trust Domain. Furthermore, as data is shared between organisations, and typically protected from access, the individuals working on shared data are also members of a separate Trust Domain. Following Definition 1, anyone who is granted access by the control mechanisms is de facto a member of the Trust Domain; therefore, if access control mechanisms are too loose, organisations or individuals may easily become un-

intentional members of a domain. For instance, consider PII shared by a company with the bank's team of employees assigned to a specific M&A. Both the company and the bank's M&A team are then intentional members of the Trust Domain for this data. However, if the bank stores this data in a database open to all its employees, de facto the whole bank becomes a member of this Trust Domain.

The same Trust Domains can be identified using Definition 2. However, since Definition 2 additionally requires shared expectations regarding behaviour and fulfilment of these expectations by all members of the Trust Domain, in the example of PII sharing there would only be a Trust Domain if either the company expected the bank team to share information within the bank, or if such sharing was effectively prevented by access control mechanisms.

Applying Definition 3, Trust Domains in the M&A example could be identified wherever organisations or individuals intend to share data and have the same expectations regarding the security of this data. Using the example of PII sharing between a company and the bank's M&A team, if the company expects the data not to be leaked to outsiders, we might identify a Trust Domain here, even if the bank's M&A team shares the data internally throughout the bank.

Finally, Definition 4 focusses on trust between the potential members. Therefore, in each interaction between organisations or individuals in the M&A example where all parties trust each other a Trust Domain is formed. Thus, the situation where PII is shared within the bank may still form a valid Trust Domain even if this is unintended behaviour from the point of view of the company. As long as the company *trusts* the bank's M&A team not to share that data internally, the requirements on a Trust Domain are satisfied.

1.2 Defenders and Offenders

In addressing Trust Domains, the perspectives of the defender or legitimate owner or user of the Trust Domain and that of the attacker must be considered. To date, much of the work on privacy protection and information security takes very little account of criminological thinking. The threat posed by insiders, especially when working in collusion, presents one of the most dangerous threats to a Trust Domain. The discipline that is closely involved in understanding the types of illegal threats to entities is criminology. Criminology helps to explain what causes or triggers crime, the ways in which crime is conducted, and how crime might be prevented. In this section we summarise insights gained

from interview studies in organisations that rely on trust and with offenders convicted for fraud.

1.2.1 Organisations' perspectives on Trust Domains

In interview-based studies with a wide range of organisations we gained a number of insights into how organisations perceive issues of trust and Trust Domains:

Trust is an enabler, and a positive sign or gesture. Trust allows people and processes to evolve to meet the challenges of the changing environment in which most organisations operate, and it builds social capital, including motivation and loyalty among staff. Thus, high-trust systems are desirable from an economic point of view.

A key element of security measures is that they depend on trust. This takes a variety of forms, including trust that problems will be properly assessed; trust that appropriate measures will be recommended; trust that these will be specified, constructed and implemented effectively; trust that they will be managed to good standards; trust that those who benefit and use measures will not set out to undermine them; trust that all people will help to identify weaknesses and report them; and trust that those responding will do so appropriately.

Trust Domains are under threat. There are a number of threats and vulnerabilities to a trusting environment or a Trust Domain, including weaknesses in the design of the domain or of the technology to support it, errors by users, and the actions of those who deliberately seek to undermine it, either from inside, from outside or by collusion between the two. Fraudsters abuse trust in Trust Domains and pose a major threat to Trust Domains. The threats that are of primary concern are insider fraud, theft (including collusion with outsiders) of information that may impact on the share prices of the organisation, and exploitation of residual or unanticipated risks or vulnerabilities.

1.2.2 Behaviour of Offenders

Understanding how fraudsters operate, their modus-operandi, the opportunities they look for, and their personal perception have a significant bearing on how trust domains should be designed.

Offenders need certain skill sets to complete an offence. Understanding these skill sets presents preventive opportunities. According to our interviews with convicted fraudsters, fraudsters' behaviour can be described in five steps: First, the offender chooses the target. Second, the offender sets up the offence. The next step is the comitting of the actual fraud, followed by getting away. The final step is the disposing of the goods. We describe each of these steps and what can be learned from them with respect to Trust Domains in the following.

1. Choosing the target From an offenders perspective there are typically two key features of a target that make it attractive: First, that the target has a value to the offender, and second, that the target is penetrable, i.e. that the goods can be obtained with a low risk of being caught, prosecuted, and/or convicted. Trust Domains thus become a worthwhile target when they contain something that an offender wants and feels he or she can get, usually accompanied by a perceived low risk of getting caught and/or prosecuted/convicted.

Insights for Trust Domains: A key feature of a trust domain is restricted access, and there are two important issues here. The first is the criteria for choosing which individuals will be permitted access. Second the robustness of the access controls: that is the degree to which they are properly specified, are fit for purpose, effectively managed and from a human factors perspective are properly integrated into the working lives of people. Once access is permitted, a number of additional issues become important. This includes: the effective monitoring of the integrity of those with access, monitoring of any changes in the types of threats that are being faced, and ongoing monitoring of the effectiveness of the protection measures.

2. Setting up the offence Most often when a fraud is committed, some type of planning or preparation is required. Those planning offences benefit from gathering information about the target. Technical information is readily available on the Internet to enable motivated fraudsters to learn how to commit offences, the question is, how effective is this information in targeting different types of trust domains?

Insights for Trust Domains: Understanding what is valuable to offenders within trust domains, and the sorts of groups this would be of interest to, may help to identify where the risks are the greatest. Regular risk assessments of vulnerabilities, including the close monitoring of those who are best placed to undermine trust, will often be crucial. Likewise, ongoing assessment of the effectiveness of security measures is important.

3. Committing the fraud In committing the fraud, offenders favour anonymity. It is somewhat ironic that those in whom extensive trust is placed, because it is essential for doing business, represent one of the greatest risks when they abuse that trust for illicit gain. This is not just about seniority – working in isolation and undertaking segregated duties with minimal oversight play into the hands of those who wish to commit offences at work. Those who undertake duties every day become experts both in the tasks they undertake and the gaps that can be exploited. Losing sight of this point represents one of the big dangers for undermining trust domains from the inside (by facilitating outsiders via collusion).

Insights for Trust Domains: As in Step 2, understanding who is in the most vulnerable positions, and applying auditing to these positions as part of a risk-management strategy may help with detecting and preventing fraud from happening, and thus with protecting trust domains.

4. Getting away The fraudster needs to avoid capture, which includes avoiding detection of the offence for as long as possible. Some offenders are able to continually offend, once they have secured access to a domain, because they have created the illusion they are trusted partners. So the robustness of who has access to a domain, and who is allowed continued access becomes an important one.

Insights for Trust Domains: Detection mechanisms may help to protect assets in trust domains; in particular, close monitoring of vulnerable positions and of the assets in question is required.

5. Disposing of the goods The fraudster needs to turn the goods into an advantage for themselves. Whether this is relevant will depend on the type of fraud, but the commission of the fraud is not necessarily the only point at which fraud management might focus. For instance, money may have to be laundered, and this is a point where the fraud may be detected.

Insights for Trust Domains: By being aware of valuable assets and, where possible, making them traceable, fraud may be detected in this step. In trust domains, this refers to being aware of the valuable information, and of applying mechanisms that help to identify the flow of information, in order to be able to detect data flows that leave the domain. For instance, electronic watermarks may be used.

From the above, it is apparent that a trust domain needs to provide access control and monitoring, all within the framework of ongoing review and reassessment of the grounds for trust. With regard to the process of fraud, detecting a fraudster's preparation at an early stage is often a realistic option for organisations given the time fraudsters take to prepare. In the context of audit and monitoring, looking for early signs of fraud or preparations to commit a fraud may sometimes be a useful weapon of defence.

It should be noted that fraud is not just about financial gain. Fraudsters commit fraud for many different reasons, including revenge by disgruntled (ex) employees. Systems that provide anonymity, sole responsibility and that provide an ability to easily override controls are seen by fraudsters as an easy target. Interesting too is the observation that fraudsters fully expect such situations to exist, and that they just need to find and then exploit them to their advantage.

1.2.3 Why security fails

Security measures are required to establish trust domains. In general, security measures work because they focus on reducing opportunities, a greatly influential cause of crime. Measures may increase actual or perceived risk to the offender and/or increase their risks, and/or they reduce actual or perceived reward for the offender, thereby rendering an offence less attractive.

Establishing clear objectives and functional requirements will go a long way towards ensuring that the performance of security systems meets expectations and that the return on the investment is measurable. To make the most of any security related system it is essential to understand and define what is needed from the technology, both in terms of the function that the system is to perform and in terms of the security that needs to be provided. There are three main reasons why security measures fail:

1. The wrong type of security measure was proposed in a specific context. This is the main reason why security specialists advocate the use of se-

curity risk assessments that take account of the threat and the likelihood of it occurring, and a security gap analysis to understand the differences between what is needed and what exists [Spe12].

2. Security may not be cost-effective. In reality, stopping crime in any one location is often possible if enough measures are used, but this does not mean that it is an economically rational way to behave. Furthermore, security measures must take into account civil liberty issues. Both aspects have a bearing on the effectiveness and the applicability of security measures.
3. Measures may be poorly installed and initiatives badly implemented, or not used in a way that was intended. Implementation failure is common, and there is often a lot that is needed to make measures work.

That measures may not be used as were intended, has become a major research area in what is broadly known as human factors or ergonomics research. There are two key points here that have been outlined by Sasse [Sas06]. The first is that security is more effective if organisations base their approach on shared norms that promote co-operation, and that it is of fundamental importance that security fits with the demands and tasks of workers' jobs, and that '(s)ecurity mechanisms that make employees lives difficult may even provide a means of [...] the breaking of security services'. So where security is not engrained into working practices it becomes a prime cause of security failure. The second factor relates to the motivation of employees. Specifically, where they see security as important and are geared to engaging with, it is likely to be more effective. As [Sas06] notes, 'committed, well trained staff who care for the organisation and their fellow employees may be the best countermeasures against these types of attack'.

1.2.4 A Trust Domains framework for building trust

Based on this high-level description and our own insights, we define a non-exhaustive set of trust-domain primitives that we believe are present in some combination in all trust domains. These primitives are shown in Table 1.1: Trust Primitives are the main features that need to be present for a trust domain to be effective. The first required primitive is the reliable identification of entities, both within and outside of the trust domain. In particular, potential members of the Trust Domain must be identifiable, as otherwise access control could not be exerted. Furthermore, the fact that entities can be identified needs

Trust Primitives	Reliable identification of entities Reliable belief in mututal values Reliable expectation of behaviour
Trust-Building Primitives	Accountability Audit Delegated Authority Trust Management Assurance
Flow-Control Primitives	Isolation Separation Policy

Table 1.1: Trust-Domain Primitives

to be communicated, and the identification needs to be communicated reliably as well. Second, there must be a reliable belief among the entities that the entities that form the domain subscribe to mutual values with respect to the purpose of the domain. Third, entities must be able to have confidence that other entities behave according to the shared values.

Trust-Building Primitives are the properties of a trust domain that enable trust to be established in the entities. The following five trust-building primitives appear important: Accountability of entities for their actions at an organisational or individual level, Audit, that is, the capability to provide evidence of correct behaviour to third parties, Delegated Authority, i.e. the sharing of information and processing capabilities with other groups, Trust Management to revise policies in order to maintain or strengthen trust, and Assurance, i.e. the confirmation of trustworthiness before shraing

Whereas trust-building primitives relate to the constituents of the trust domain and the processes within the domain, Flow-Control Primitives govern information flow. Of particular importance are isolation, that is, the ability to place restrictions on information-sharing outside of a tightly controlled group, separation, i.e. the assurance that information is only shared between trusting entities, and the existence of a policy, i.e. the definition of operating characteristics that meet security needs.

In practice, these primitives need to be implemented using a combination of technology and social means. For instance, the reliable expectation of acceptable behaviour can be supported by technical or legal enforcements.

The above features are required for the implementation of a Trust Domain. In general, these features will require technical support in order to be implemented. At the same time, security measures that support Trust Domains must be carefully balanced against the work that is performed inside the Trust Domain, that is, they should not affect entities involved in the Trust Domain to such an extent that they offset the additional value obtained from the Trust Domain, or such that they are perceived as impacting on the work by the employees. Either might result in individuals trying to circumvent the security measures, thus making breaches of trust more likely.

Example: Mergers and Acquisitions Process

Illustrating the insights from criminology, let us assume that a bank employee is involved in insider fraught, for example, in selling data on an on-going M&A process to outside parties. Following the above five-step decision-making process, the employee first chooses the target. We may argue that in this example the choice has already been made, as the person is already employed by the bank. The employee may have specifically started work with this bank because they perceived that the bank would be an attractive target. Once employed, the would-be fraudster has to choose an appropriate M&A target, which they may choose based on the value that can be obtained from this target and on the ease of getting access to the valuable data. In the second step, the fraudster then sets up the fraud. In particular, this involves ensuring access to the data and procuring means for exfiltrating a copy from the bank. Both of these steps will involve some sort of exploratory behaviour, which may be detected. In the third step, the fraudster commits the fraud, i.e. they copy the data to a device that is outside of the bank's security controls and can thus be used to hand-over the copy to the fraudster's customer. Step 4 is closely related to Step 3 in this example, as the employee has to move the copy out of the bank. Furthermore, the fraudster will attempt to hide their traces by, for instance, modifying or disabling audit logs. Finally, in Step 5 the fraudster disposes of the goods by, in this case, selling the copy to their customer.

1.3 Modelling

In practice, a Trust Domain comprises a complex network of interactions between organisations, individuals, which are governed and supported by social norms, expectations, and technical infrastructure. Modelling is used to explore

	Verification	*Evaluation*
Design-time	PoliVer, StatVerif	Gnosis++ Automated Model-Generation Tool (AMG)
Run-time	—	Online Model-Generation Tool (OMG)

Table 1.2: Modelling tools for Trust Domains.

this situation, helping to understand the implications of particular decisions, and thus ensuring that important properties hold. Modelling can support trust, as it can give guarantees on some properties, and it can support designers and users in making optimal decisions, both when the system is designed and when the system is in use. As illustrated in Table 1.2, we broadly distinguish modelling approaches based on the nature of their outcomes and on the point at which they are applied in the system lifecycle:

Modelling for verification: The goal of modelling for verification is to verify that something can or cannot happen, i.e. if the rules that govern the system's behaviour allow certain outcomes to occur. For instance, modelling for verification could prove that an attacker cannot gain access to some information. A system, or components of a system, for which such a proof has been obtained can then be trusted.

Modelling for evaluation: The goal of modelling for evaluation is to obtain insights into quantitative properties of the system under the assumption of stochastic behaviour. Quantitative properties of interest could be leakage rates of information or the time it takes to perform certain operations. Modelling for evaluation thus helps us to understand the impact of decisions on operational properties. This insight is helpful in optimising parameters when there are tradeoffs between e.g. performance and security.

We also distinguish according to the point in time at which modelling is applied:

Design-time modelling: Modelling at design-time is performed in order to optimise the design of the system before the system goes on-line. Modelling decisions to be made here include the type and number of tech-

nical components, policy settings, and operational parameters of components. Typically, design-time modelling is not time-critical, and thus approaches that have a high overhead in terms of constructing the model or in obtaining results from the model can be applied.

Run-time modelling: Modelling at run-time helps to support decisions during the operational phase of the system by assessing and illustrating the impact of user decisions on properties such as security or performance. Run-time modelling is particularly important in the complex settings typical for most Trust Domains, where decisions may have far-reaching unintended results. As run-time modelling is applied to support the user in operational decisions, it must be possible to return results fast.

The Trust Domains project has developed and applied various methods for modelling in these scenarios. We discuss the modelling approaches in the following sections.

1.3.1 Modelling for Verification

In modelling for verification the goal is to establish the possibility or impossibility of performing certain actions or achieving certain goals in a given setting. This allows us to verify if a security protocol fulfills its requirements, i.e. is robust against specific attacks, or if the policies defined for a system prevent users of that system from performing undesirable actions. Modelling for verification thus helps with the task of determining if a complex system of rules, such as a system policy or a secure communication protocol, accurately reflects the intentions of its designer.

In modelling for verification the system is described in terms of states and actions of agents that change the states. The modeller then queries the model if, starting from a given state (or set of states), another specified state or set of states can be reached. Typically the tool will then report not only whether the target state can be reached, but also return a sequence of actions that lead to the target state. This sequence helps to identify potential paths of attack, and how to prevent them from succeeding.

In the Trust Domains project two modelling tools support specific needs of the project by verification methods:

Policy Verification Tool (PoliVer): Policies support Trust Domains by formalising permitted and forbidden actions of agents in a system and enabling

the automated enforcement of these permissions. In this way, trust can be placed in a system, if it is known that the system does not permit actions that would break the trust. The design of policies for any system of realistic size is tremendously complex, as the designer has to ensure that the policies capture all cases and forbid all unintended actions. The PoliVer tool helps the designer in this task by automated checking if a system of policies fulfils its requirements.

With PoliVer, the designer first specifies the states of the system and the actions that can be performed by agents in that system to change the state. Then, the designer specifies the policies that restrict the set of actions that can be performed in each state. Finally, the designer specifies an initial state and an undesired target state, i.e. a state that should not be reachable. The PoliVer tool then applies a search proceeding backwards from the target state, establishing sets of predecessor states until either no further states can be discovered or the initial states are found. If the latter is the case, then the policy allows a sequence of actions that leads to the undesirable state; i.e. the policy is not effective. The tool also returns the sequence of actions, which provides an example of the path an attacker would take. This example can guide the designer in deciding how to amend the policy in order to make it more effective.

Protocol Verification Tool (StatVerif): Security protocols describe and control the interactions of the components that form a Trust Domain. When followed, a secure protocol ensures that trust can be placed in a system, and thereby helps to establish a Trust Domain. The StatVerif tool [ARR11, Sta] is used to prove the security of protocols for systems with a global persistent state.

With StatVerif, the protocol designer first describes the protocol in terms of processes exchanging messages. Processes model agents, while their message exchanges represent their interactions. The behaviour of processes may depend on a global state, which may be changed by actions performed by the processes. The designer then formulates a query that describes an unwanted state, for instance, an attacker having access to data that should be protected. StatVerif evaluates the given protocol model and determines whether this state can be reached. If the state can be reached, the protocol needs to be modified such that the successful attack becomes impossible.

1.3.2 Modelling for Evaluation

In modelling for evaluation we assume stochastic behaviour of the entities involved in a system. This allows us to capture situations where entities can make choices; in particular, it enables us to model the case where human agents may choose between different ways of performing an action, each of which may have different impact on security, and thus on the trust that can be placed in that agent. Furthermore, based on the assumption of stochastic behaviour, we can represent mistakes made by human agents, and faults, errors and failures of technical components. We then obtain values for quantitative properties of the system, such as the time until a security incident occurs, a measure of the trustworthiness level, or the time that an action will require when security measures are applied. These values help to make informed decisions with respect to parameters that affect tradeoffs in system design and system operation.

In modelling for evaluation we can use different levels of detail. Models with a high degree of abstraction represent the system and its properties using mathematical equations and stochastics. They produce general results and can often be evaluated with little computational effort. On the other hand, such models omit many details, capturing them in stochastic behaviour instead, and may therefore misrepresent crucial aspects of the system. Furthermore, they may require considerable effort in abstraction and translation on the part of the user in order to choose appropriate parameters and in order to apply these parameters to the practice. Models with a low degree of abstraction represent the system and its properties using abstractions of the processes and resources in the system. With these concepts, a much higher level of detail is achieved, as the behaviour of entities can be modelled using a notation that is similar to a programming language. This allows the modeller to capture important properties, and also makes it easier to parameterise the model, as parameters can be reflected directly, without the need for abstraction. On the other hand, the results from the models are specific to the modelled situation, and evaluation of these models typically requires simulation, which is computationally expensive.

High-level and low-level modelling complement each other, and the application of approaches from both abstraction levels can help in different aspects of the evaluation of Trust Domains. In the Trust Domains project we have developed three modelling approaches to support decision-makers both at design-time and at run-time:

Gnosis++ Gnosis++ is an extension of the Gnosis modelling tool [Cor]. Gnosis++ enables low-level, high-detail modelling of system behaviour and thus helps the system designer in assessing the impact of a wide range of changes on the security and trustworthiness of a Trust Domain. The tool employs the concepts of process, resource, information, location, and link. *Processes* implement the behaviour of system components and human agents. Processes operate on and modify resources and information. *Resources* model physical resources, such as hardware tokens, while *information* models data. The difference between resources and information is that the latter can be created, copied, destroyed, and accessed by more than one agent at a time, whereas physical resources cannot be created nor destroyed, and can only be moved. That is, the global number of physical resources always stays constant in a model. Resources and information reside in *locations*, which can be connected by *links*. Locations and links thus can be used to represent the permitted movement of resources and data. In particular, this can be used to model containment and isolation mechanisms. The Gnosis++ tool allows the evalulation for a wide range of metrics, reflecting security, trustworthiness, performance, and dependability, amongst others. These metrics are typically defined on the amount of resources and information at specific location, but can also utilise various events, such as successful break-ins. Gnosis++ is implemented in Ruby, and thus supports co-routines and all object-oriented features of the Ruby language, which makes it particularly easy to use.

Gnosis++ is typically employed as follows: First, the modeller decides on which physical and logical assets of the system need to be included, and which metrics need to be obtained. Physical assets are then modelled as resources, while logical assets are modelled as information. Then, the behaviour of of technical components, such as servers or firewalls, and human agents, such as users, administrators, or attackers, is modelled. These are implemented as processes operating on resources and information. Processes can be implemented in an object-oriented fashion, where each instance of a process has its own local state, and where process behaviour can be inherited and modified, in order to model classes of agents with similar characteristics and to support patterns of agent behaviour. The model is then simulated, and the metrics are obtained, e.g. by counting the number of information items on a particular location.

As the model has to be simulated, this approach is computationally expensive and can only be applied at design time.

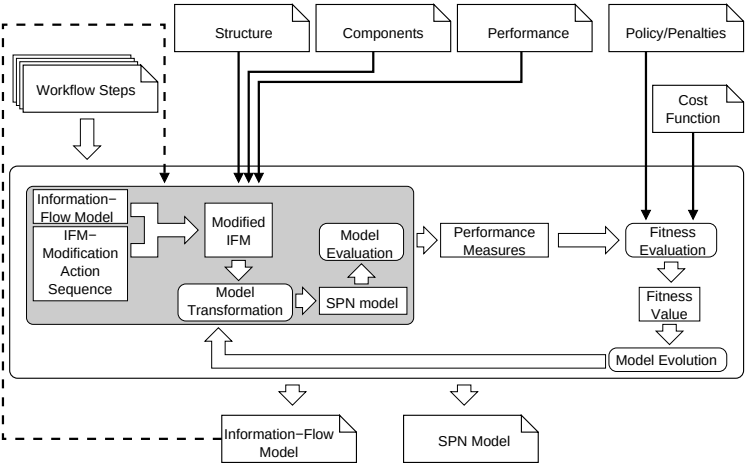


Figure 1.1: Operation of the AMG Tool.

Automated Model-Generation Tool (AMG Tool) The Automated Model-Generation Tool (AMG Tool) helps system designers in evaluating the impact of policy settings and performance characteristics on the behaviour of human agents affected by the policies, and on the security on the system. The tool uses an information-flow abstraction, that is, it models the system in terms of information flows between source and destination nodes in a graph. The behaviour of human agents is modelled as sequences of actions establishing or removing connections, which then enable or disable flows. Performance characteristics are evaluated as the time it takes for information to flow from the source node to the destination node. The security is evaluated by the time it takes for the information to flow to an attacker node.

Figure 1.1 illustrates the operation of the tool: The user of the tool describes a workflow by splitting it into information-exchange steps. In each of these steps, data needs to be transmitted from a source to a destination. The data transfer can be achieved using different technical means (e.g., the data might be shared using a Cloud service or printed and posted using a courier service). The source, the destination, and the means of transport are modelled as nodes in a directed graph (referred to as the Information-Flow Model (IFM)). In order to form a graph that connects the source and the destination, the nodes must be connected by links. For each of the links the tool user can

specify the speed and a penalty associated with using that link. The penalty models the policy; i.e., modes of transport that are discouraged can be associated with a high penalty. Initially, only the properties of these links are given, but no links are established. The tool then applies a genetic algorithm to find a graph that connects the nodes, such that the graph is optimal with respect to a user-specified cost function. The cost function can capture aspects such as risk-affinity, by trading off penalties against performance. Furthermore, the cost function can also be used to model the impact of a timeout. The graph produced by the tool serves two purposes. First, it illustrates the flow of information between the source and the destination. Second, it can be used as an input to the modelling of attacker behaviour. In this step, the genetic algorithm modifies the graph in order to direct information to the attacker's node. This model then illustrates attack points.

In the background, the genetic algorithm operates on sequences of actions that add or remove connections in the graph; that is, the genomes of the algorithm are sequences of such actions. A sequence's fitness is evaluated based on the model that results from applying the sequence to the graph model, as follows: Penalty costs are accumulated from the actions in the sequence. Performance costs are computed by transforming the graph model to a Stochastic Petri-Net (SPN), from which the distribution of times for moving data from the source to the destination can be derived as a phase-type distribution.

The output of this approach is thus two-fold. On the one hand, it produces results that are immediately useful for assessing the impact of choices at design-time. On the other hand, the tool automatically generates Stochastic Petri-Net (SPN) models of the system (optionally including attacker behaviour) from high-level specifications of performance characteristics and penalties. These models can be used in further evaluation of various properties. The advantage of automatic generation of models over manual modelling is that the former may include details that could be overlooked by a human modeller. For instance, a combination of policy settings and performance characteristics might favour the use of an insecure way of transport, but this might not be obvious from the specifications, and thus would be omitted in a manually created model.

It should be noted that this approach is computationally expensive and thus is typically applied at design-time, rather than at run-time.

Online Model-Generation Tool (OMG Tool) The Online Model-Generation Tool (OMG Tool) applies principles of stochastic modelling to help users make

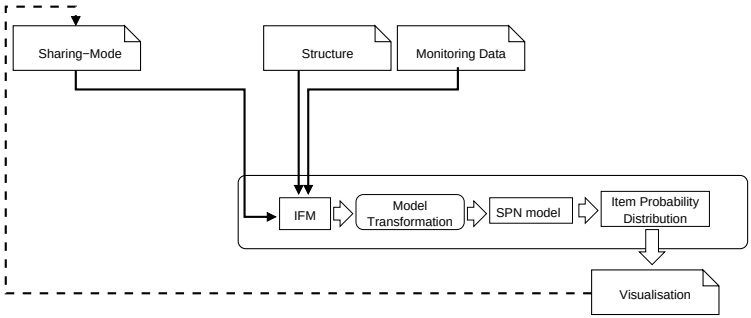


Figure 1.2: Operation of the OMG Tool.

informed decisions at run-time. In this approach, we consider the following situation: A user wants to share data with one set of users, but does not want another set of users to gain access to the data. Recipients of the data may vary in their trustworthiness with respect to not sharing the data, and the users' devices may also be likely to leak the data. The data may be shared in various ways with different security properties, e.g. on some sharing methods it may be possible to prevent leakage of the information from the recipient's device. We assume that there is some monitoring data available on the trustworthiness of users and their devices, and we want to support the sharing user in making a decision of recipients and sharing mode, such that it is unlikely that unintended recipients receive the data.

The operation of the tool is illustrated in Figure 1.2: Potential recipients of the data are abstracted as nodes in a directed graph. The edges of the graph model information flows. Each edge has a parameter that specifies the rate at which data flows along this edge. The rates for data flow from the sharing user to the intended recipients is set to reflect the speed at which data can be shared. The outgoing information-flow rates on recipient nodes are parameterised based on a combination of trustworthiness monitoring data and the security of the selected sharing-method. The graph model is transformed to a Stochastic Petri-Net, from which the probabilities of each node having seen the data before a given time t are computed. These probabilities can then be displayed graphically, e.g. as a terrain map or as a colour-coded visualisation of the graph. Based on this information, the user can revise and adjust decisions accordingly, in order to reduce the probability of unintended parties receiving the data.

The underlying equations for this approach can be solved quickly, and therefore the method is applicable at run-time.

Example: Mergers and Acquisitions Process

We will now illustrate applications of modelling in the Mergers and Acquisitions (M&A) example. In this example, complex interactions take place between the organisations involved in the M&A, e.g. the bank and the companies, between the individuals tasked with the work, and between the human agents and the technical components that they use to transmit, process, and protect the data required throughout the process.

Each of the organisations will typically have policies in place that aim to protect the data its employees operate on. The existence of such policies will also typically be a prerequisite that other parties demand before entrusting data to the organisation. The PoliVer tool can be used to ensure that these policies are effective in protecting the data, and in assuring other parties that the data is secure. For instance, the bank needs to process data from the companies. In order to convince the companies that the data is secure and can only be accessed by the individuals tasked with the respective M&A, the bank would verify its policies using the PoliVer tool. The result could then be communicated to the data owners. Alternatively, the bank could also communicate the policies, and the data owners could then verify their effectiveness against breaches. Verification thus helps to enable trust.

Throughout the M&A process, data has to be exchanged between organisations. There are various ways in which data may be exchanged, such as by e-mail, using special infrastructure, using a Cloud sharing service, or by physical tokens, e.g. USB media. Which method is appropriate depends on the amount of data to be exchanged, on how time-critical the exchange is, and on the sensitivity of the material. Using the Gnosis++ tool, the system designer can explore the impact of different sharing methods on the performance, dependability, and security of the data exchange. For instance, in order to exchange large amounts of data, USB media might be used; however, as such media might easily be lost, the data needs to be encrypted. Encryption, on the other hand, may slow down the process, and therefore agents might opt not to encrypt the data. With an appropriate Gnosis++ model, the designer can explore how much data may be lost, and how much data may be successfully exchanged. This will give insight into how to set policies and what impact these will have; for instance, encryption may be made mandatory, but this would in turn require a relaxation of time constraints.

The Automated Model-Generation (AMG) tool can be used for exploring the effect of policy settings at a higher level of abstraction. For instance, the bank may install a special secure service for sharing data with outside organisations and require all its employees to use this service. Typically, employees would follow this policy, as they want to protect the data. However, employees also need to perform their work, and if the security mechanisms slow them down, they may try to find ways to circumvent them. The AMG tool will help to identify such problems. Furthermore, it will allow the policy designer to experiment with assigning penalties for behaviour that is not compliant with the policy. Such penalties will encourage employees to follow the policy. However, it may also be the case that policy-compliant behaviour is not compatible with completing the task on time. This will also be highlighted by the AMG tool.

The Online Model-Generation (OMG) tool supports the user at run-time. For instance, an employee at one of the companies involved in the M&A may need to send internal data to the M&A team at the bank. However, as there are many teams within the bank, the company employee needs to be sure that the data is not shared with bank employees that work in teams responsible for other M&A processes. The company employee may use different modes of sharing the data, e.g. he may send the data by e-mail, or he may require the use of a specially secured virtual machine by the bank. He then has to decide which mode would be appropriate. In this situation, on-line modelling helps the employee by illustrating the probability of the data leaking to other employees or other teams within the bank. These probabilities can depend on trustworthiness values obtained from special monitoring infrastructure and also on the sharing mode; e.g., if a locked-down virtual machine is used, the probability of data leaking is lowered.

1.4 Trust Domains in Practice

In this section we discuss two software packages developed in the Trust Domains project to support Trust Domains in practical scenarios. These packages address two different situations: ConfiChair is a trustworthy management system for the organisation of the review process of scientific conferences, while the Trustworthy Collaboration System (TCS) is a general collaboration system for document sharing.

1.4.1 The ConfiChair Conference-Management System

ConfiChair addresses the paper-review process for scientific conferences. In this scenario, authors submit papers to the conference. A chair assigns the papers to reviewers and, based on their recommendations, decides which papers will be accepted. This is a scenario where the workflow is well-specified. The scenario also has a clearly defined attacker model: It is assumed that the system is hosted on a Cloud Computing service. Two properties need to be maintained: First, the content of papers and reviews must be kept secret. Second, unlinkability between author and reviewer has to be ensured, that is, it should not be possible to determine which reviewer has reviewed a paper by a particular author, and vice versa.

ConfiChair achieves the first of these goals by encrypting all data that is stored in the Cloud. The second goal is achieved by encrypting the data and mixing the identifiers that correlate reviewers and papers between steps of the workflow, such that all links that may be determined in one step are invalidated in the next.

The strict assumptions on the behaviour of the participants make it possible to obtain a proof on the security of the ConfiChair system. Using the ProVerif tool [BAF08, Pro], it has been proven that the Cloud cannot gain access to any information, provided that all participants adhere to the security protocol.

1.4.2 The Trustworthy Collaboration System (TCS)

The Trustworthy Collaboration System (TCS) focusses on the scenario of loose collaborations where documents need to be shared between individuals. In this scenario there is no clearly-defined workflow and no clearly-defined attacker: Users of the system share data based on a workflow that is not prescribed in the TCS, and users decide who to share with at run-time, making this decision based on the data and contextual information that is outside of the system. For illustration purposes we employ the following example for such a collaboration: A presenter wants to share slides with an audience. Some of the slides may only be shared with a sub-set of the audience, as they contain sensitive information. The presenter makes sharing-decisions based on whether they perceive the recipient of the data as legitimate, and on how trustworthy they consider the recipient and their device to be. The goal of the TCS is to support the user by providing facilities for helping them make optimal decisions with respect to whom to share with, and by enforcing these decisions by technical means.

The TCS achieves this goal by employing modelling and monitoring. Monitoring enables the system to provide the user with run-time feedback on the trustworthiness of entities and feeds into run-time modelling. Modelling is applied at design-time and at run-time. Design-time modelling supplies feedback on how particular parameter choices will affect a Trust Domain before the Trust Domain is created. Design-time modelling takes as input parameter choices and obtains Gnosis++ to obtain output values that represent to the user the impact of these choices on the Trust Domain. The user can thus select parameters that result in a Trust Domain with acceptable properties. Run-time modelling provides feedback on the likelihood of entities receiving a document that is being shared. The presenter can then decide whether to share the document with these settings, based on whether they trust the potential recipients.

The TCS does not provide guarantees on the security of the Trust Domains implemented with it. Instead, it enables the user to make decisions that result in intended outputs, and to avoid mistakes resulting from insufficient knowledge of the effect of decisions.

* * *

This chapter gave a high-level overview of the major results of the Trust Domains project. This overview serves as a guide for establishing trust domains, and as a guide to the more detailed chapters of this book.

Chapter Two

Sociological Aspects

In this chapter we approach trust domains from a sociological and criminological point of view. Our aim is to identify basic characteristics of trust and of breaches in trust that will guide our understanding in later chapters.

Figure 2.1 gives a high-level overview of this chapter: We first discuss the concept of trust from a sociological perspective. A survey of existing work shows that trust is generally considered a beneficial attribute, both in society at large and in business, particulary since it simplifies and even enables interaction. We then address the question of defining trust and give a definition of a Trust Domain that is based on the sociological discussion.

The next step in understanding trust domains is in understanding the behaviour of those who attack them and the difficulties in defending them. We draw upon insights from criminology to describe the behavioural patterns of offenders. In particular, we discuss the behaviour of fraudsters and identify the fraudster's decision-making process as well as the resources required to commit a fraud. Understanding the behaviour of the offender and their resources helps to determine how to prevent a fraud from taking place. In order to understand the difficulties with trust domains in practice we interviewed a broad range of professionals from different organisations. Their responses allow us to explore the reasons why trust domains often fail in practice. Understanding these enables us to identify which questions need to be addressed by technical implementations and by modelling.

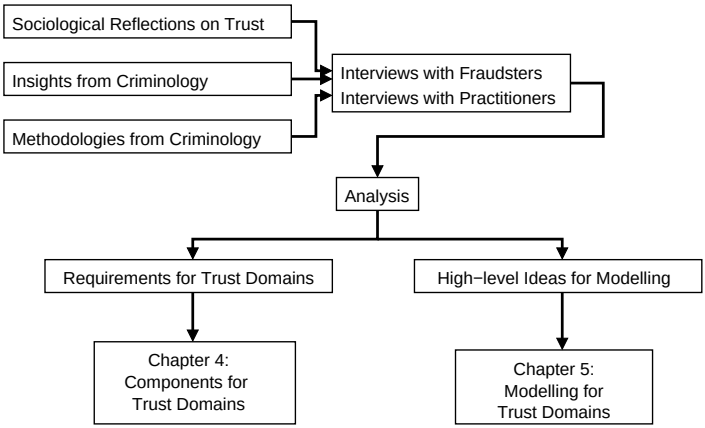


Figure 2.1: General Structure of the Chapter.

2.1 Trust from a sociological perspective

The issue of trust, and what it means in different contexts and different disciplines, has been widely discussed. In this section we highlight some of the key characteristics of trust, and what they may mean for the discussion of trust domains, conscious that research findings on trust are mixed and more developed in some areas than others.

The early Greeks studied trust in order to better understand humans and human behaviour [Bai02]. More recently, trust has been studied in a wide range of fields, helping to better understand interpersonal relationships in psychology [Rot67], adherence to rules in management (where trust can be seen as a form of informal governance) [Sak98], and the relationship between buyers and sellers [MZD92]. In political science, the concept of trust is seen as a means of studying levels of social trust between citizens [DHP07]. In these, and in many other disciplines trust is seen as a core component (although not always the most important one [DHP07]) of understanding relationships.

There are a host of studies that have evaluated how generating trust can be good for business, not least in generating sales via meaningful trusting relationship with clients [JG05]. One writer has stated that trust ‘is emerging as one of the key concepts in the search for the secrets of competitive success’([New98],

p.35). The work on trusted platforms also has suggested a number of benefits: 'The higher levels of trust that are enabled through technology by Trusted Platforms are valuable to business because companies gain by being trustworthy, [b]rand image suffers if there is a breach of trust or privacy, [b]etter trust enables more powerful management services, [c]onsumers' trust is a major business enabler, [and] [i]mproved trust and security is necessary to the delivery of business-critical e-services' ([PCPP03], p.29). There is much research that focusses on online businesses, where the need to build trust from customers is a major challenge [BP07], requiring a range of imaginative approaches down to the design of the user interface [WE05]. The public sector faces a similar challenge, and one study at least has shown that the perceived ease of use of the technology, and its trustworthiness, are significant predictors of citizens' intention to use an e-government on-line service [CB05].

On the other hand, the findings from research on trust are somewhat mixed. In particular, for some researchers trust is but one element and not necessarily the most important in mitigating risk [LCP06]. Similarly, one study has noted that consumer confidence is provided by other factors such as 'caring policies, evidence of a popular outlet, attractive packaging, and clean and bright spacious shops' ([Nat02], p.1). Clearly this will vary with different types of business transactions and with the types of parties involved, but trust is typically an important factor in instilling confidence between parties.

Trust is generally viewed as a positive attribute, as it is a key characteristic of a virtuous society [Fuk95]. In particular, trust is viewed as central to good business, as all sides engaged in transactions need to trust others involved, even though the amount of trust given or placed will vary with circumstance [Har06] and cultures [Bie09], and will be tested by different circumstances such as rapid company development [Rob96], as well as by the presence of a known or potential threat, which is where the study of crime and security comes in.

This interpretation of trust as a positive attribute becomes clear when we consider failures of trust. There is a range of studies which have shown the high degree to which a company's market value is based on intangibles such as reputation [ENS07]. The reputational damage from being a victim of e.g. fraud can be considerable when there is a danger that the organisation may be seen as culpable. The implications are that the organisation did not manage its affairs sufficiently well and as a consequence cannot be trusted [Lev08]. Repairing the damage can be time-consuming and costly [Far05].

In a different way the presence of trust has been linked to levels of staff satisfaction at work [Dri78], facilitating good relationships with clients [JG05],

and to generating competitive advantage [New98]. The level of trust in technology and the outputs from technology have been linked to the quality of security decisions that are made [KMV+12]. Even in research, building up trust with an interviewee is deemed important and key to generating valid data, and this includes work with offenders [Zha10]. Not surprisingly then, managing trust is seen as something to be good at [ENS07]. Furthermore, it has been argued in ergonomics research that building trust into security systems is desirable from an economic point of view [SAL+07], since without trust high levels of reassurance have to be built into systems which are costly and less flexible.

In summary, there is broad agreement that whilst trusting people involves some degree of risk, trust has many benefits: Trust allows people and processes to evolve to meet the challenges of the changing environment in which most organisations operate, and it builds social capital, including motivation and loyalty among staff. Consequently, high-trust systems, i.e. systems where individuals place high levels of trust in each other, are desirable from an economic point of view.

2.1.1 Defining Trust

Although there is agreement on the benefits of trust, trust is difficult to define. Indeed, defining trust has long been held as conceptually problematic [Bai02]. The lack of a single unified definition of trust makes it difficult to operationalise in the business environment, as expressed by [MC10]:

> *'Contrasting, the requirement for trust leaves the team in a vacuum. What they face is an endless series of questions. What definition of trust should be applied – behavioural, cognitive, statistical? – and how does a definition fit with their own perception of trust? Is this definition culturally acceptable? Is the requirement really about a system that has to be trusted, that is desired to be trusted or that is supposed to be trustworthy? Does this requirement relate to the design, to operation, to the system or to its social environment? Who should be consulted – psychologist, social scientist, cryptographer or philosopher? – How can requirements be captured, progress measured, designs developed and tested? What design procedures should be applied and what methodologies should be used? What metric can be used to judge the success?'*

Nonetheless, [WE05] identify four key aspects of trust that can be used in analysing trust relationships:

Trustor and trustee: There must exist two specific parties in any trusting relationship: a trusting party (trustor) and a party to be trusted (trustee). Trust depends on the ability of the trustee to act in the best interest of the trustor, and the extent to which this is reciprocated.

Vulnerability: Trust is only a requirement when there is uncertainty and there are risks. Trustors take the risk of losing something important to them; trustees must be effective and must not take advantage of the vulnerability.

Produced actions: Trust is related to risk-taking behaviors. These will vary considerably with context.

Subjective matter: Trust is a subjective matter and so is the perception of what are acceptable risk levels and trust relationships.

2.1.2 A Definition of Trust Domains

As a consequence of the difficulties in defining trust, trust domains are equally difficult to define. Nonetheless, it is important to attempt a definition of a trust domain, accepting that in so doing we enter largely unexplored (yet somewhat contested) territory. From a sociological point of view, we define a Trust Domain as follows:

> A trust domain is where information that is designated to be in some way privileged is subjected to access controls or similar type of protection.

Privilege relates to the fact that access to information constrained by a trust domain is not freely available to everyone. But a trust domain is not just about information, although we recognise that information and information flows dominate the discussions on this topic. A trust domain can also consist of resources and/or assets, which are typically used to process information or support a business process. For example, a critical infrastructure provider would have systems that they manage in order to satisfy a critical national need. It is where information that is designated to be in some way privileged is subjected to access controls or similar type of protection.

There is an expectation that those with access to the information will be required to act in a compliant way, and face sanctions if they do not. Organisations appear typically to implement several trust domains. Some might

'overlap' because an individual could have more than one role, which grants them access to multiple trust domains.

2.2 Criminology: Crime, Crime Prevention and Crime Frameworks

The discipline that is closely involved in understanding the types of illegal threats to entities is criminology. Criminology is the study of crime and the social responses to it, and includes theories and frameworks to guide thinking and practice on its prevention. It also includes the process of treating and punishing offenders, and the processes by which people become victims. As a subject it draws upon a wide range of disciplines both in the natural and social sciences.

Criminology has been improving crime control by developing theories and frameworks of crime, which can be used for crime prevention. These frameworks and approaches have been developed to explain the deeper causes and immediate triggers for crime, the ways in which crime is conducted, and, consequently, how crime might be prevented. The relevance of these frameworks will vary with the type of crime or threat being considered, the types of mitigation already in place, the suitability and effectiveness of mitigation measures in place, the context in which offences take place, the mindset of the offender, the skill sets and experiences of the offender, the culture of the organisations in question, and the resourcefulness of victims.

On a very general level, there are *theories of causes*, which focus on injustices or inadequate structures in the way society is organised or on the mal-adjustment of the individual. These causes have their place in explaining crime, but are difficult to address. On the other hand, theories that explain crime in terms of how offenders make *economic* judgements on whether to commit crime based on their perception of the likely success and/or reward, which is the focus on the rational offender, have the advantage that they present opportunities for more immediate action.

Opportunity-of-crime theories, sometimes referred to as the criminologies of everyday life, have been developed from several perspectives: Felson's Routine Activity Theory [CF79, Fel02], the crime pattern theory of the Brantinghams and environmental criminology [BB08], cf. [PG11], and the limited rational-choice theory, which has fed the interest in Clarke's situational crime prevention. These have been very powerful in explaining how crime occurs

and what makes specific targets attractive. This helps explain what needs to be done to reduce the chances of a crime occurring. However, there has been a lack of attempts to integrate them, an exception being Ekblom, who has attempted to do so in his Conjunction of Crime and Opportunity (CCO) [Ekb11].

Marcus Felson's *Routine Activity Theory* is a relatively straightforward explanation of crime, but nevertheless a very powerful one. The theory purports to explain the conditions necessary for a crime to take place. It has undergone fairly extensive revision since it was first introduced, and the originaly three key conditions have been extended to six. They are:

1. There needs to be a law or rule to break
2. There must be sufficiently likely offenders, i.e. people able to commit the offence
3. There must be sufficiently capable offenders, i.e. people with the skills and resources needed to commit the offence
4. There must be suitable targets, i.e. targets that can be attacked
5. There must not be a sufficiently credible guardian (that could prevent the offence from taking place)
6. There must not be a sufficiently significant and censorious handler (a specific type of guardian), who the offender would not want to offend in sight of.

Each of these factors provides a focus for crime prevention efforts, and most can to a greater or lesser extent be controlled by measures. The most significant piece of work undertaken on reducing opportunities is that by Ron Clarke in his *25 Techniques of Situational Crime Prevention* [Cla95]. The approach is based on the premise that offenders (e.g. fraudsters) commit an offence because an opportunity presents itself. Thus, in this approach, a key reason why a crime takes place is because there is the opportunity, so by taking away the opportunity we take away the chances that a crime will be successfully completed. Clarke's *25 opportunity reduction techniques* derive from five key principles, namely that 1) crime prevention should aim to increase the effort it takes for a fraudster to commit an offence; 2) increase the risk of him or her being caught; 3) reduce the reward if successful; 4) reduce the number of provocations that lead people to commit offences; and 5) remove the excuses that are sometimes offered as a reason for committing fraud (such as 'the company can afford it'). Each principle is then divided into a set of techniques, conveniently five for

each one, making 25 techniques in all. The situational crime prevention or opportunity reduction approach is very flexible and can and has been applied to a wide range of offences.

There are many criticisms of the situational approach, including that it ignores causes, blames the victim, and creates inequalities among them. For Ekblom [Ekb11] the limits of various approaches, including Clarke's, and the obversvation that they are simplistic and incomplete, led him to launch a framework he calls the *Conjunction of Crime and Opportunity (CCO)*.

If the criticisms of many frameworks are that they are simplistic, the same cannot be said of the CCO. It is an ambitious framework that maps a range of different intervention principles, outlines the requirements for prevention mechanisms to be triggered, defines perpetrator techniques, and guides crime assessment [Ekb10]. It provides a map of 11 causes of crime, that is, events that can result in an offence occurring, and then 11 counterpart principles of intervention which aim to mitigate the effects of those causes or stop them happening. The CCO provides a helpful encompassing framework for thinking about understanding problems and the potential of different solutions to provide an effective and targeted response.

There is one other approach amongst many that merit attention here that is known as the *5Is Framework*, a very helpful framework for thinking about how to evaluate initiatives that permits the use of different methodologies [Ekb11]. It focusses on five areas that constitute essential components of successful programmes, and as such provide the key areas of attention for any evaluative approach or for checking progress as an initiative is being developed and implemented.

2.2.1 Criminological Thinking and Trust Domains

The criminological frameworks discussed above have been applied in discussing e.g. armed robbery, burglary, or shop theft, but are rarely considered in the context of information security. Indeed, they have not often been applied to thinking about how best to respond to crime in organisations,[1] but they clearly have potential in this area. They offer an opportunity to think through four key aspects: The *causes* of crime, the different possible ways of *responding* to causes, ideas for identifying the key mechanisms or drivers for *crime pre-*

[1] See [GC13] for a discussion of existing applications.

vention, and ways of assessing risks and the credibility and appropriateness of different *solutions*.

These frameworks can thus be used to explore the dynamics of different types of threats to trust domains, including some of the potential strengths and weaknesses of different mitigation measures.

On the other hand, it is worth noting that a key element of security measures is that they depend on trust. This takes a variety of forms, including trust that the problems will be properly assessed; trust that appropriate measures will be recommended; trust that these will be specified, constructed and implemented effectively; trust that they will be managed to good standards; trust that those who benefit and use measures will not set out to undermine them; trust that all people will help to identify weaknesses and report them; and trust that those responding will do so appropriately.

2.3 Understanding the Offender

Understanding the views of offenders is important, because from offenders we can learn how they approach their crimes and what they view as the characteristics of an easy or difficult target. Parker [Par98], writing in the context of considering the criminal threat, noted the importance of understanding cyber criminals for their skill, knowledge, resources, access and motives.

It should be noted that the study of offenders' perspectives pose methodological difficulties. In particular, precisely because a crime event is illegal, it is secretive, and so researching it is always difficult. Although some studies have shown that offenders are a rich source of data, there are inevitably methodological problems, including the fact that it is difficult to verify the accuracy of what offenders state. There is always the danger that offenders will mislead interviewers, deliberately or accidentally, and this includes not being clear themselves as to the complex reasons that can often be relevant to the commission of an offence [Ber10]. Furthermore, the offender perspective remains a minor issue in crime studies, and this includes studies of computer crime and insider threats [Wil06, WS09]

A related problem, which applies not just to crimes in business, is especially relevant here: Research into many types of offences, and fraud included, is of limited value in that there is a tendency to focus on reasons why people commit offences, rather than how they do so, the specific skills and tools necessary. For instance, people are often advised to 'think thief,' i.e. to identify and assess the effectiveness of security measures by seeing through the eyes of

a thief. However, there is an implicit assumption that this is easy. In fact, even many thieves are not accustomed to 'thinking thief,' and lack the skills to do so.

Offenders' Scripts are an important way of understanding crime. They were originally used in cognitive science to understand sequences of decisions and the links between them, and have been developed to explain crime commission. As Cornish [Cor94a], pp.157–158 notes, 'Scripts are members of a family of hypothesized knowledge structures, or schemata, considered to organize our knowledge of people and events. Such schemata are held to guide our understanding of others' behaviour, and our own actions. The script is a special type of schema, known as an "event" schema, since it organizes our knowledge about how to understand and enact commonplace behavioural processes or routines' (see also [Cor93] and [CC86]. As argued in [WS09], p. 136, 'One benefit of developing a script is that it encourages practitioners to consider all stages of crime commission. In this way, all the criminal behaviour in the process can feasibly be identified. Once this is achieved the next stage is to implement the appropriate controls.'

Willison ([Wil06], p. 315) argued that, 'each script comprises event sequences extended over time. The events in the sequence are interrelated, given that events at the early stages of a script afford the occurrence of later ones ... the script concept focuses on behavioural processes involved in rational goal-oriented actions'. And, 'Script analysis may thus be viewed as a conceptual and empirical scheme for delineating how crime-relevant performance and learning opportunities interconnect' ([LT03], p. 190).

There are different types of crime scripts. As Cornish notes they 'can operate at different levels of abstraction [. . .] from the most specific instances to the more inclusive and more abstract categories of script' ([Cor94a], p.159). As discussed in [Cor94a, Cor94b], the different types include the Universal Script, the Metascript, the Protoscript, the Script and the Track. In another way, scripts can be used to evaluate why crimes fail just as much as why crimes succeed, and this can offer important crime prevention lessons. The essential elements of the 'universal script' proposed by Cornish in [Cor94a] are:

- Preparation (what needs to be done)
- Entry (how access is gained)
- Pre-condition (What needs to happen for a crime to be possible)
- Instrumental pre-condition (what needs to be done to access the goods)
- Instrumental initiation (the process of getting access to the goods)

- Instrumental Actualisation (how the offence is facilitated)
- Doing (conducting the offence)
- Post condition (how one covers one tracks)
- Exit (getting away)

In a practical application to the study of cheque fraud Cornish et al [Cor94a] identify four components of a script:

- Getting an identity kit (for example by stealing from the letter box)
- Getting cheques (for example by opening a bank account)
- Making cheques (for example by computer scanning)
- Scoring (for example cashing the fraudulent checks).

They found six ways of getting an identity kit, five ways to getting cheques, four ways to making cheques and two ways of scoring. The authors then compared these four components with both the personal characteristics of each offender, and the outcomes of each individual's case (for example, whether charged, number of frauds committed) to examine cheque fraud. Potentially each element of the script offers a point for intervening to stop the crime.

The script approach has its critics, not least in the extent to which crime scripts can actually capture the complexity of criminal acts, and crime scripts can be dismissed as a simplification [Cor94a]. Nonetheless, beyond their importance for crime-prevention purposes they also provide a way of understanding how offenders learn about crime. In doing so, they complement Sutherland's much discussed *differential association theory* [Sut47, Sut49], which describes how offenders learn from those around them, but does not explain the process by which knowledge is acquired. Crime scripts might also help to explain why some offenders are more successful and more innovative than others, and identify and explain changes in the ways that offences are committed over time [LT03]. Indeed, there are two processes involved: 1) that of learning about crime; and 2) that of acquiring performance knowledge (which will be discussed in an evaluation of resources needed to commit offences in Section 2.3.3).

In order to assess more precisely how offenders behave, the decisions they take and the reasoning behind them, we will now focus on a specific type of offender, viz. the fraudster. There are several advantages to focussing on fraudsters. The first is that fraud consists of an array of offences, undertaken

for material gain, that increasingly involve some type of technology (hence the relevance to the subject of trust domains). In the UK the legal definition of fraud was tidied up by the Fraud Act 2006. Of interest here is the three-way classification of fraud given in the Fraud Act, which encompasses *false representation*, *failure to disclose information* and *abuse of position*. A second major advantage of our focus is that we can draw upon original research to understand fraud from offenders' perspectives.

This work in part draws upon (mostly unpublished and new) research with fraudsters in prison carried out by Martin Gill. The first is a study of those who steal from their employer ([Gil05a, Gil07, GGW10], see also [DG07]), and the second a study of different types of fraudster and how they are influenced by economic circumstances [Gil11a, Gil11b].

2.3.1 Why and how fraudsters commit fraud

Probably the most frequently referred-to work on motivations for fraud is by Cressey, who saw trust violation as a key ingredient of a white collar crime. He developed the 'fraud triangle' and asserted three key elements of a fraud by employees are necessary: offenders with a *motivation* (he wrote in particular of offenders having non-shareable problems), an *opportunity* (in his work Cressey was especially interested in the offender abusing a position of trust), and *rationalisations* to neutralise guilt (see [Cre50, Cre53]). The argument he makes is that taking away any one element can prevent a fraud from occurring. The model has been subjected to extensive critique as new research of different types of fraud conducted in different contexts has questioned the universal application of the approach [DG08, SL13].

In [GGW12b], Gill and Goldstraw-White summarise that the reason why people commit fraud can be classified into three general groups. First, frauds are committed because the fraudsters make an economically rational decision that the benefits outweigh the costs. Second, fraudsters are motivated by the need to resolve personal pressures that fraud provides a means of generating release from. Third, fraud is a response to exploiting opportunities that are presented (see [BME09]). The remedies rest, in the first case, on making the costs of committing fraud high (especially increasing the prospect of getting caught), in the second case on identifying and helping to resolve personal pressures on individuals (and especially those who have a propensity to commit financial crimes if they can be identified), and in the third case on reducing opportunities in the way that the organisation is run and does business.

Martin Gill notes that when you ask offenders why they committed a crime on a particular day, all too often they state they did it because it was ‘easy’. So despite all the measures in place at banks and in retail outlets, both bank robbers and shop thieves all too often state that the crime was ‘easy’. Sometimes, measures can even work in the interests of offenders. For example, labels attached to goods in stores stating that they ‘are only sold in high street stores’ (and therefore render them more difficult to be sold in other locales such as market stalls) can be an advantage for thieves. Some stated that when selling stolen goods buyers were rarely concerned about them being stolen, but they did worry about the possibility of them being counterfeit; a label from a high street store added to their authenticity.

Gill’s interviews with fraudsters who stole from their employers revealed seven reasons for the offence, namely [Gil05a, Gil07, GGW12a]:

- Debt (which was generally perceived as out of control)
- Boredom (they were seeking excitement)
- Search for status (and money from fraud helped provide this)
- External coercion by blackmail (they were threatened to be exposed for another offence)
- A temporary lack of emotional balance (they lost control of their senses)
- The influence of organisational cultures (characterised by poor ethics, weak leadership, and tolerance of dishonest practices)
- Opportunism.

More recently [Gil11b] interviewed 16 fraudsters in prison about their approach to fraud and the economic climate. In terms of motivations, five reasons were offered, these were:

- They needed money, because they were bad at managing debt
- They want to win favour, and fraud produced the funding to facilitate that
- They had an addiction, and fraud helped provide the funds required to satisfy the addiction
- The opportunity was there, and some stated that it was easy and that attracted them
- The fraudster was enabled to commit fraud as part of carrying out normal business duties.

There are clearly overlaps in the explanations, and for some individuals several apply at the same or different times. The last of these is included as a separate category, although in reality it is facilitated by the appearance of an opportunity. But its importance should not be underestimated, because where organisations create opportunities in the way they conduct business they must expect (or not be surprised) if at certain points these lead to crimes taking place. Mitigation is therefore imperative. Indeed, while opportunity is not necessarily a condition for fraud, it is typically important and often mediated by 1) the extent to which an 'inner voice' helps offenders overcome a desire/need to commit crime [SL13] and 2) company culture [BW06].

There is one other topic that needs to be discussed, albeit briefly, in any assessment of why people commit crime, and that is the driver of 'need' and 'greed'. There is little doubt that these are relevant to some fraud offences. For example, Tunley [Tun11] (p. 314) has examined the motivations for benefit fraud and concluded, 'empirical evidence has been offered that benefit fraud is motivated by need and greed, with opportunity acting as a catalyst'. Goldstraw-White [GW12] has assessed the issue of need and greed in fraud offences from her interviews in prison. She concluded that white-collar offenders often gave accounts for their offending, which could be classified as 'greedy'. She noted that, 'greed appears to feed upon itself' (p. 109). In other words, when fraudsters realise that they have a way of making money from crime that appears easy, they find it difficult to stop, even if the original financial reason for offending has been satisfied.

2.3.2 The Fraudster Decision-Making Process

It is important to note that there are various ways of intervening to prevent crime, and in an organisational setting, as elsewhere, dealing with motivations is clearly important but not always practical. Often the cause of crime at the workplace relates to problems in personal lives, such as addictions or financial concerns, which organisations may not be aware of. Even if they are, employers may feel there is little they can do about it.

Another mechanism for thinking about intervening in crime is to look at how decisions are made. What is presented below is a framework for how many fraudsters will approach their offending. Clearly not all the elements will be relevant to the same degree, nor will they always be present or made in the order depicted, but they do represent key decision points that most fraudsters will consider. They also represent key points in offending which provide an

opportunity to influence how the offender behaves; ideally to stop an offence taking place.

1. Choosing the target. In reality there are many factors that can influence what makes a target attractive to a fraudster. A key factor will be the amount the offender knows, or can discover, about the size of the reward and the security weaknesses at and around the target. In the case of crime, familiarity breeds opportunity. Clearly staff, perhaps those who are trusted the most (but this is not always the case), are the ones who are key to protecting the organisation. Ensuring that everyone works towards the company's best interests, and is motivated to report any suspicious activity or security weakness they identify is important. It is also very important that the company does not get a reputation for 'being easy'.

It is worth noting that burglars sometimes return to the scenes of their crime because they know they will be successful; and fraudsters sometimes return to victims from previous offences, because they believe they will be easy targets. Some victims even appear on 'sucker lists' which are bought and sold on the illicit market [Gil11a, Gil11b]. Taking a strong line on dealing with crime – and this includes determined but effective response – can often be a very effective method of protecting the business. Offenders favour both finding easy targets and avoiding difficult ones.

2. Setting up the fraud. Most often when a fraud is committed some type of planning or preparation is required. This is not always the case, of course; some frauds are opportunistic. The process of setting up a fraud is more difficult for insiders when, for example, individuals are monitored closely, where work is checked or audited, and where security weaknesses are understood, monitored and acted upon. The key here is to think of fraudsters and ask, 'what is it that will make this offence more difficult?'

3. Committing the fraud. The third stage is perhaps the most crucial of all as it involves committing the fraud itself. Clearly at this point the dangers are considerable, for the offender and, if he or she is successful, for the victim too.

There are at least two major considerations for the offender, and these are nearly always present and therefore should be a consideration when thinking about preventing crime. The first is that they will want to ensure they get away, while the second is obtaining the rewards. The more difficult this is, the more

work that has to be done (and the more clues they may leave afterwards), the less attractive the target is.

4. Getting away. Another issue of crucial importance to the fraudster is avoiding capture. In practice this has a number of aspects, including avoiding the offence being detected for as long as possible (ideally, from an offender's viewpoint, altogether), avoiding any link directly back to the offender, avoiding action being taken once discovered, preferring that if some reaction is necessary it takes place at as low a level as possible, and ultimately that no prosecution takes place and if it does that it is unsuccessful. There is much to commend initiatives that make offences more risky.

5. Disposing of the goods. Whether this is relevant will depend on the type of fraud. But the commission of the fraud is not necessarily the only point at which fraud management might focus. Money may have to be laundered and a process will be needed for that. If goods have been obtained they need to be converted into money (unless for self use). This has many dangers for the offender [Sut10], as the police have become more adept at managing the second-hand goods market.

2.3.3 Fraud Facilitators

Another issue when considering offenders is not just how they make decisions, but the resources they need to undertake the offence successfully. Routine Activities Theory has highlighted that in addition to being motivated, an offender must also have the resources needed to carry out the offence without also being caught. Work has been conducted on the resources needed for offending (see [Gil05b]). The discussion here is applied to fraud (see [Gil05a]) and builds on work conducted in prisons with fraudsters [Gil05a, Gil07, Gil11a, Gil11b]:

Resources for handling emotional state. In order to commit an offence, fraudsters need to be emotionally prepared to do so. In the criminological literature much has been written about 'techniques of neutralisation,' (focusing on the need of fraudsters to overcome any feelings of guilt they may have). But it seems that many fraudsters do not feel guilty, either because they do not sufficiently consider the consequences of their actions or because they feel the victim is deserving or can afford being defrauded. As a consequence, avoiding

staff having grievance, and dealing with them speedily and effectively when they do, is not just good management, but also good crime prevention.

Resources derived from personality/character traits. Different characteristics or features of a personality impact not just on whether a fraud should be committed, but the type of fraud that could be committed. In [Gil11a], Martin Gill found many fraudsters to be risk-takers, good at taking advantage of opportunities. In [Gil11b], he reported that some fraudsters felt that committing fraud was easy but having nerve when handling the proceeds of fraud was crucial; that was when they felt there was a greater danger of getting caught. One fraudster claimed to have committed the offence principally because he had low self esteem and the process of frauds provided the funding to appear generous which he hoped would win favour with people. Another fraudster highlighted the importance of a good memory for avoiding capture: 'I had to convince a fraud inspector once and I did. You have to have all the information in your head, you have to, dates of birth, the lot. You must have it all in your head, you cannot hesitate, they are getting paid a bonus to find you these days'. Clearly, the type of character/personality traits needed will vary with the type of fraud, but profiling fraudsters and understanding characteristics that make offences possible is a much under-researched methodology.

Knowledge-based resources. [Gil05a] found that fraudsters mostly gained the knowledge they needed from their everyday work. They knew the processes and procedures and different types of fraud-prevention measures in place, and, crucially, they knew how to circumvent them. Martin Gill in his study of offenders found one long firm fraudster who learned how to build up credit, by buying flowers, then paying for them promptly so that over time they were able to earn the right to 90 days credit. At this point they would make a large order and disappear with the flowers and without paying. Another interviewee, an accountant, described how his knowledge of 'how to massage figures on profit and loss and balance sheet,' enabled him to defraud his company.

Skills-based resources. Skills differ from knowledge in that the former are the practical techniques needed to apply knowledge (the facts). The important issues about skills concern which ones are needed to commit different types of fraud, and where and how are they learned. In [Gil11b] Gill found that some

fraudsters were caught because of a lack of skills. One solicitor was convicted of fraud partly because he did not properly check the identity of those who were asking him to act, and it transpired they were defrauding an organisation. He knew what should have been done but 'took [his] eye off the boil' in what he later recognised was poor judgement. Another cheque fraudster (he made and then cashed counterfeit cheques) noted: 'There is a lot of preparation work, making cheques, that takes a lot of patience. You need template and then it is easier, putting details in is easy but laborious. You scan in a proper cheque and then you have to work on it, there is a lot to it. Then you have to plan what you do and where you go.'

Resources derived from physical traits. Physical traits, such as strength, may be used to intimidate or physically restrain victims to enable the offence to be executed. This is usually less important for fraud than other offences, although there has been value in some frauds in the offender appearing intimidating.

Tools or 'crime facilitators'. Crime facilitators are the factors that help offenders commit crimes. For committing fraud, a crime facilitator may for example include a computer programme that enables the fraudster to access personal accounts and download personal information. Many identity fraudsters need birth certificates or other personal information in order to create or take over identities that can then be used for fraud.

Associates and contacts. An associate is a specific form of crime facilitator, separated out because of its importance. Often but for a contact a fraud would not take place, like where someone provides information about a target which makes the fraudster believe that a fraud is worthwhile, perhaps relating to weaknesses in fraud prevention. [Gil11b] found one fraudster who claimed *'I would buy details off of friends . . . I had people who would sell me their NI number and I would get their dole.'* Another fraudster admitted that it was the need for friendship that led him to work for a group of people who had asked him to commit fraud. They provided all the details about the offence, his job was merely to withdraw money. Getting rid of the proceeds of fraud (when not for self-use) will normally require contacts. The cheque fraudster noted above claimed he always passed the goods he obtained through fraud via the same fence, 'He was always reliable. He once owed me £20k and paid me.'

In this section we have identified the fraudsters' decision-making process and the facilitators that enable a fraud to take place. These insights help to identify weaknesses in trust domains and ways to address them. In particular, by interrupting the decision-making process, and by preventing the would-be fraudster from gaining necessary resources fraud can be prevented.

2.4 Difficulties with Trust Domains in Practice

We will now turn our attention from the offender to the defender. We present an empirical study designed to identify the importance of trust domains in practice and the practical problems involved with establishing trust domains.

2.4.1 Study methodology

The empirical study was based on interviews with members of staff of a broad range of organisations.. The interviews were with both senior members of staff who have oversight of organisational governance and with staff in operational roles. Every organisation and every individual was carefully selected to ensure a broad mix of disciplines and business verticals. The organisations include public and private sector, profit and not-for-profit, product-led and service-led, national (most UK-based) and international organisations. The selected organisations are known to deal with sensitive issues because they are involved in police and social work or are engaged in highly competitive industries. Interviewees were selected by a process known as 'snowballing': Starting with contacts known to the research team and its partners, we built up further contacts via those we interviewed.

The key areas of focus for the study were *information-flows*, i.e. information at rest, in transit, and leaks and losses of information, and *operating boundaries*. Within these two areas closer attention was paid to identifying:

- The main risks to business in engaging in information exchange
- What constitutes trust in the context of information exchange
- How trust is valued, and how it reinforces/undermines security
- How organisations use information about each other to enhance confidence and improve interactions
- How organisations react when forced to share information in uncomfortable situations, in situations where priorities are at odds with each other

- To what extent, if at all, technology helps.

Our schedule of questions was designed around four roles that we identified as key to defining trust domains with an organisation. Each role supports, to a greater or lesser degree, the role of information protector. The four roles are the *Risk manager*, the *Business manager*, the *Security architect*, and the *Technical administrator*. We assume for clarity that the four roles are non-overlapping, but we appreciate from our study that this may not be the case in practice, since roles are rarely this clearly defined and distinct from one another. Our rationale is that these roles cover operational management (risk ownership) and support/strategic planning.

The focus, and a feature at the heart of our interviewing schedule, is on what trust means to the interviewee in the context of his or her organisation. We envisage individuals having a duty to manage information, which leads to a consideration as to how that information is protected when shared. All of our findings were anonymised, an obligation we made to everyone we interviewed as is normal practice in research of this kind.

2.4.2 Insights

Organisations on the whole rely on trust domains, although evidence so far suggests they never call them that. In our interviews interviewees report how organisations and employees organise information in terms of domains of trust. Yet, there is a range of issues that appear to make the process of generating effective trust domains problematic. Although intuitive, as noted above, trust domains can be hard to describe and hard to explain (particularly beyond anything more than a superficial interpretation), and therefore it can be hard to see just what constitutes trust at a theoretical level. Specifically, the criteria defining the practice governing who can be trusted and why is undefined. A further issue requiring clarity surrounds the mechanisms used for engendering trust, and similarly to respond to situations where trust is threatened or broken. Below we summarise these issues and highlight the key reasons why, according to our interviewees, they have emerged as causes of concern:

- Many of the polices that organisations rely on do not work because they are flawed or not followed
- Technology does not take sufficient account of human behaviour
- In many organisations the status of information protection is low, compared to other business tasks, such as selling and finance

- International differences complicate the development of consistent regimes
- There are real limits to the technology being deployed
- Organisations do not know who they are dealing with
- The nature of business means relying on trust.

Each of these headline observations – and in practice they can often overlap – is discussed in turn. A more detailed discussion may be found in [GC14].

The rules for securing trust domains do not work. Part of the difficulty for those working in data protection and information security is that often the outcome is dependent on people within the organisation following rules, but that data protection and information security is often not the priority of these people. This becomes an acute problem when the rules that are there to help protect assets start to get in the way of doing business.

There are at least three explanations here. First the rules may be ineffective in that they do not, or are not perceived to, adequately protect information. Second, the rules are bad ones, bad in the sense that although they will protect information they are not conducive to the working practices of those on the front line, so consequently they are not adhered to. Third, the rules are good but there is not sufficient awareness of them. A linked theme around awareness is that the criticality of the rules, in particular the role that the rules play in protecting an organisation, are not widely appreciated. Indeed training and awareness about rules is reported to often be less than is required. Of course, all three of these explanations may on occasion be evident in any situation to a greater or lesser extent.

Whether it is the principles behind the rules at fault, or the interpretation and embodiment of the principles that is lacking, it is clear that policies can create difficulty. Where policies do not work or are circumvented, it means that the organisation has less control over what happens, and that increases risks (assuming of course there was a sound reasoning for the process in the first place). But does less control imply greater flexibility? Balancing risk and flexibility is key to understanding when and where trust domains make sense, and where they breakdown and become unworkable.

Technology does not take sufficient account of human behaviour. The focus here is very much on human behaviour rather than the rules themselves.

It is not difficult to protect information where security is the only aim, or agreed principle aim, but this is rarely the case. In reality, security is rarely user-friendly and often an impediment, and the realities of making things work in practice under potentially high pressure are not thought through.

One reason that security measures are circumvented is that people have good knowledge of the other people they are dealing with, either within their own organisation or another one. This facilitates trust in circumventing policies that may otherwise complicate business.

Another reason are cultural issues. It was noted that in universities it was necessary to take account of a culture that emphasises individuality and academic freedom, and is sometimes stretched to mean that as academics the employees have a free reign on activities involving sensitive and potentially damaging information.

These issues are sometimes known to organisations, and taken into account. One interviewee noted that their organisation set aspirational policies on purpose, designed to exploit the psychology of people behaviour. He gave the example of passwords, where the aim was to make people aware of what was needed in the hope some would follow what was being prescribed. They knew not everyone would follow good practice, but at least some would have adopted better security practices as a result.

The status of information protection is low. In many businesses, tasks like selling and finance are perceived to be of more importance and higher status than the protection of information. This is potentially a controversial claim, but there are a number of findings that appear to support this assertion. The first is the extent to which policies are overridden by senior managers in pursuit of their own ends, which can lead to some fractious discussions. In other examples we have seen interviewees pointing to policies as being 'aspirational', rather than requirements that arise to satisfy the fulfilling of organisational priorities. An interviewee noted that complex environments can lead to policies being developed that are contradictory, and as a consequence become a low priority or result in the policy being seen as not important.

This resulted in confusion and a lower take-up than was hoped for. Another interviewee described other departments pursing policies in flagrant disregard for good information security, and noted that this was in part a response to them not viewing information security in the right way. We were also informed that these departments also believed that they could do whatever they want because someone else was responsible for information security.

This interviewee noted that the resources devoted to information security were slight, and this may reflect its status. Yet another interviewee thought that ignorance was the main reason why security attracted a low priority.

International differences and the development of consistent regimes. A major impediment to good systems for securing trust is the different regulatory requirements that apply internationally, meaning that organisations need to apply different approaches in different parts of the world. This can mean, and often does mean, that different processes and technologies are needed, and there are different requirements on staff too. There may also be geographical policy differences, in part because requirements are different, but also perhaps because there is less trust in regimes in some parts of the world. Hence the amount that one might want to disclose may be very different. Using a mixture of localised (i.e. different) processes can increase risks.

There are real limits to the technology being deployed. When technology operates in silos, having overarching domains where people share information can be problematic and at times impossible to achieve. Another problem reported relating to technology is that security people often ask questions that are difficult for non-security people to answer, for example, 'How do you want to protect your information?', or perhaps more relevantly, 'What level of protection do you need?' Being able to understand technical jargon and make informed choices requires a specific set of skills, and these questions assume that people know the 'how and what' of security, which too often they do not. Similarly, we heard that technology is not always easy to use for a non-technical person, and security people are sometimes seen as 'specialist in making things difficult'. The particular interviewee pointing this out noted that protection measures are hard to use.

Another issue reported with technology is that it quickly becomes dated. Worse still, more devices are coming on board that individuals can afford and have access to but companies just cannot, which creates the need for policy exceptions to accomodate private devices. This forces companies to develop strategies for Bring-Your-Own-Device (BYOD) scenarios (cf. [Inf]).

Technology certainly enables users to do more with information, or to do their job in multiple ways. And this complicates the situation for the security architect. New technology, as we heard form one interviewee, needs to be thoroughly understood and tested before it can be safely deployed, and that takes time and resources to complete.

Security controls are also described as sometimes not being ‘up to scratch’. One interviewee admitted that organisations cannot mitigate all possible threats. Some choose to focus on just the major risks. Another problem mentioned relates to managing the controls. This interviewee also noted that there are technology issues in the way systems are structured, meaning that it can be difficult to connect to third parties.

Organisations do not know who they are dealing with. Many interviewees admitted that they had to share information without knowing who they were sharing with. They would know the organisation, and may have a rough idea if it was a customer/partner, but not know any details. Many of those interviewed said that they had at some time shared data and received data that they should not, the consequences of which could have been severe. But there is also evidence that personal relationships play a major part in building trust. For instance, the ex-work colleague who has moved into a new role is often cited as a strong trust point in the mutual exchange of information.

The nature of business means relying on trust. Several interviewees noted that the more typical criticism of security is that it reduces liberty, in that security restrictions reduce freedoms. In practical terms, the more practical consequences of security are that it reduces reliance on trust. While this can sometimes be a good thing, it needs to be understood that business generally operates on trust, as one interviewee noted: ‘Bear in mind that the way that brokering works is based on trust. That is the perception of insurance. We trust clients to tell the truth and we trust insurance brokers to treat clients’ claims truthfully. The fact is we just don’t see people as untrustworthy, actually the opposite. This is how insurance has always been done; the old ways survive in our world. Some industries do nail down every possibility, but that is not trust.’

2.5 Analysing Trust Domains

Our working definition of a trust domain is a grouping of two or more entities that share the same level of expectation regarding security of information that they wish to exchange with one another. Further, we recognise that there exist one or more entities that must be denied access to the information. The basis for trust between the entities that form the trust domain can be built on both

Trust Primitives	Reliable identification of entities Reliable belief in mututal values Reliable expectation of behaviour
Trust-Building Primitives	Accountability Audit Delegated Authority Trust Management Assurance
Flow-Control Primitives	Isolation Separation Policy

Table 2.1: Trust-Domain Primitives

technical and human factors, legislation and regulation, and policies and procedures. Conceptually, trust is based on mutual human values. Technology is called for in situations where human behaviour and human limitations present a threat to the trust domain, or where technology offers improved functionality. For example, technology can make compliance with the aims of a trust domain easier to achieve.

We employ the concept of *information flows*, defined as an exchange of data between two or more entities that together represent a trust domain, where this trust domain describes the boundary across which access is restricted. Data flows according to a stated process, where the security of the data that flows demands exceptional protection (with respect to data that is allowed to freely flow between all entities). In practice, we are interested in information flows within and between trust domains, i.e. between organisations, departments and individuals. For example, within an organisation we see trust domains consisting of two or more departments, work colleagues and managers (possibly located at different geographical sites). Similarly, we see trust domains being formed between organisations. At a personal level, we see trust domains being formed between individuals, for example a patient and their doctor, an individual and their bank manager, and between friends and family.

Based on this high-level description, we define a non-exhaustive set of trust-domain primitives that we believe are present in some combination in all trust domains. These primitives are shown in Table 2.1: *Trust Primitives*

are the main features that need to be present for a trust domain to be effective. The first required primitive is the reliable identification of entities, both within and outside of the trust domain. In particular, potential members of the trust domain must be identifiable, as otherwise access control could not be exerted. Furthermore, the fact that entities can be identified needs to be communicated, and the identification needs to be communicated reliably as well. Second, there must be a reliable belief among the entities that the constituents of the domain subscribe to mutual values with respect to the purpose of the domain. Third, entities must be able to reliably expect other entities to behave according to the shared values.

Trust-Building Primitives are the properties of a trust domain that enable trust to be established in the participating entities. The following five trust-building primitives emerge as important: Accountability of entities for their actions at an organisational or individual level, Audit, that is, the capability to provide evidence of correct behaviour to third parties, Delegated Authority, i.e. the sharing of information and processing capabilities with other groups, Trust Management to revise policies in order to maintain or strengthen trust, and Assurance, i.e. the confirmation of trustworthiness before sharing

Whereas trust-building primitives relate to the constituents of the trust domain and the processes within the domain, *Flow-Control Primitives* govern information flow. Of particular importance are *isolation*, that is, the ability to place restrictions on information-sharing outside of a tightly controlled group, *separation*, i.e. the assurance that information is only shared between trusting entities, and the *existence of a policy*, i.e. the definition of operating characteristics that meet security needs.

In practice, these primitives need to be implemented using a combination of technology and social means. For instance, the reliable expectation of acceptable behaviour can be supported by technical or legal enforcements.

2.6 Modelling Trust Domains

A key feature of the trust domains project has been recognising the importance of information flows, and to take them as the basis for future modelling activities. The concept of sharing information, of information flowing from one organisation to another, and that information is important and needs to be protected, was generally well understood and openly accepted by our interviewees. Risks vary with organisational context and culture, and this is where modelling can have the greatest impact.

In this section we identify important questions to be answered using modelling, the aspects that need to be modelled, and the available parameters. Furthermore, we point out the existence of tradeoffs.

Based on our interviews, the following priorities emerged as the most important aspects for those attempting to determine whether to trust a relationship build around a trust domain:

- Why should I trust?
- What problems do I need to be aware of?
- Is there a better way that I should be considering?

Our interviewees also pointed out the need to understand a situation better, which often arose from a sense of professional ignorance. They stated that they do not have the necessary information to make a choice, they are not used to dealing with a particular terminology, and that they still needed to provide a convincing argument to their peers, even if they understood the situation.

We dectected in talking to several interviewees that the trust decision-making process tended to be based on subjective values. Modelling may be able to bring a degree of objectivity, for example by supporting in-house review of another party's policy or technology, vetting of individuals, reliable separation of roles and duties, agreement on acceptable levels of risk, assessment of the costs of breaches or failures to comply with legislation, and by helping to determine reputational damage.

In the following we discuss several aspects that give rise to specific questions of modelling.

Attitude to taking risks. Perhaps predictably, all interviewees that we have spoken to recognise the need to comply with legislation and other regulatory requirements. They also believe the organisations they work for are committed to achieving this. However, on several occasions iterviewees reported that policies that had been developed to promote good information security were being ignored. The reason was often that the policy in question prevented individuals from doing their job, sometimes altogether and sometimes by creating inconvenience. As a result, employess found workarounds to the rules. These 'workarounds' invariably led to greater risk for the organisation and for the individual who appear to have chosen to ignore corporate practices, and were only possible if individuals 'abused the trust placed in them' – albeit, they reasoned, with the best of intentions.

According to our interviewees, for the most part employees' actions were well intended and justified on the basis that the policies were unworkable in

the given situation, or at least were not seen as the best guidance for optimising broader benefits to the organisation. This leads to an intriguing balancing equation between a commitment to following policy requirements and a commitment to completing a task. It appears from our questioning that people are willing to risk personal admonishment (or worse) in order to 'get their job done'. However, by and large they were not expecting to get caught, and even if they did they expected to be able to appeal that they were working in the best interests of the company. In many cases the implication was that they believe that should their actions be noticed, 'commonsense will prevail', and the organisation will understand the need for rule infraction in the pursuit of a greater good.

This leads to the *tradeoff between performance or convenience and security*. All interviewees naturally showed concern about risks to their operation, and all keenly demonstrated a rational approach to dealing with risk. However, there emerged a distinct difference in the level of risk that one organisation would accept compared to another. For example, an organisation involved in preserving national critical infrastructure is typically averse to taking any risk, and seeks strong assurance when operating outside of its tightly protected perimeter, and is often reluctant to cross the boundary at all. At the other end of this spectrum are organisations well versed in taking risks, that accept failure as the 'cost and consequence of doing business'. In between we see what can be perhaps best described as 'informed risk taking'. Whilst attempts are made to reduce risk, there is general acceptance that a level of residual risk is acceptable. This residual risk is typically expressed in terms of failure to meet regulatory requirements, penalties (especially resulting in reputational damage), and personal accountability (e.g. penalties imposed on individuals). In effect, there is a tradeoff between the risk of failure to deliver and the penalty for circumventing policy.

Ownership of security In the sample of interviewees, two clear groups are in evidence; those that define security and those that are affected by security. This inevitably leads, on occasions, to different agendas. We heard on several occasions that the policy setters reported (or were reported) as being less concerned with the practicalities of the requirements they set, and more concerned with being seen to have defined a good security solution for a given set of threats. The consequence of this, it was reported, was that some of the solutions would turn out to not be applicable to their particular organisation, seemingly a very poor practise. This led some interviewees in operational

roles to routinely ignore or circumvent policy in order do their job. The first weakness leads to, and then compounds, the second.

This observation points to the existence of a *principal-agent problem* (cf. e.g. [MCWG95]), i.e. tensions between the rules prescribed by a principal and agents performing under these rules.

Internal vs. Outsourced service provision Whilst some organisations already outsource some functionality, particularly around the provision of technology infrastructure, many (perhaps most) are reported to be still evaluating the options. Online services, like those described as a good fit for the Cloud, are certainly high on some organisations' outsourcing agenda, and it may simply be a matter of time before the move to greater outsourcing for service provision occurs. Most interviewees noted that a strong argument for moving to the Cloud was reduced operating costs, although in a couple of cases there was recognition that an external specialist service provider – particularly a technical/security specialist – should be able to provide a better service too and would therefore be more trusted than an internally provided service. In these conversations there was also an element of being able to shift the blame when 'things go wrong'. Some interviewees were very uneasy and resistant to the idea of outsourcing, e.g. the critical infrastructure provider. Most often this centred in being unsure and/or unconvinced by the advantages to risk management and the quality of assurances given by outsourcing providers, and the interviewees appeared reluctant to fully trust them. All agreed that they would need more time and strong evidence – stronger than for an internal partnership – before they would feel comfortable trusting external partners. They envisaged needing to be more diligent when gathering evidence to build trust, and that getting to the point where they would be happy to proceed with a partnership would take longer than for a similar internal partnership.

This observation points to the requirement that modelling should help to explore *different design alternatives*.

Information Flows It has emerged is that information-flows are in fact an important focus of this research enquiry, providing a basis for understanding and describing how trust domains work. Perhaps with the exception of the critical infrastructure provider, all organisations that we spoke to base their business model on a need to efficiently send and receive information with parties both inside and outside of their organisation.

That information has value is widely appreciated, but it appears that organisations have great difficulty articulating that value (as do practitioners and academics). The concept of data classification is not widely adopted, but it is talked about occasionally when describing higher value information, and it is not clear that even documents marked 'private and confidential' are always tightly controlled.

* * *

In this chapter we have studied issues of trust and trust domains from sociological and criminological perspectives. Based on interviews with fraudsters and within organisations we have learned that organisations rely on trust in order to perform their function, but may fail to enable trust and to ensure that trust is well-placed, and that offenders undermine and use trust in order to gain advantages or to cause damage.

We discussed crime scripts as an appropriate way of describing offender behaviour. Based on this concept we identified the five-step decision-making process of fraudsters and the resources that they need in order to perpretrate fault. Our interview studies with employees of organisations that handle sensitive data allowed us to identify key properties that must be supported in a trust domain and key issues to be addressed by modelling.

The insights presented here summarise the work done in the Trust Domains project. More details, including full interviews can be found in the deliverables of the project, particularly [CG12, GC13] and in [GC14].

Chapter Three

A Semantic Model for Trust Domains

The concept of a trust domain exists to provide a foundation for securely sharing information among a group of (possibly distrusting) entities. It enables the parties involved and any observers to appreciate the level of trust present before proceeding to share data.

A definition of a trust domain that is suitable for this purpose is the following: A trust domain is *the state and processes that allow resources to be shared between entities that are members of the domain and where these entities have an expectation of and exhibit shared and predictable behaviour to protect the resources.* The key points in this definition are that a domain should ensure that members exhibit certain behaviour and that this behaviour can be checked by those placing trust in a domain.

A trust domain is established by allowing participants to specify the information that they are willing to share with each another, according to a policy that defines how, when and with whom this information can be shared. This policy is then enforced by control mechanisms, defined as part of the definition of the trust domain, which provide evidence that supports assertions about the properties of the trust domain as well as enforcement of the policies. These controls have to be achieved through some technical (e.g. access control, roles, cryptography) means as well as social means (e.g. contracts, obligations, law enforcement). Based on this definition, a trust domain aims to:

1. Define mechanisms for controlling membership, that is, it defines procedures for joining or leaving the domain

2. Provide interfaces through which data flows. These interfaces enable separation of channels through which security-sensitive and non-security-sensitive information can flow
3. Provide infrastructure that determines and enforces which data can flow through which interfaces, both within organisations and across multiple organisational boundaries
4. Provide the mechanisms for determining the level of security provided.

In this chapter we identify the concepts that can be used to describe a trust domain and how these concepts are related. This characterisation is captured in the form of a model based on semantic web techniques. This *semantic model* enables and simplifies communication about trust domains, which is required both in implementing and in modelling them for evaluation and verification.

The chapter is structured as follows: We first identify key technical and social components. We then present a three-layer approach for discussing these aspects and study the relations between these components we identify. Based on this, we define our model. Finally, we discuss applications and extensions to this model.

3.1 Objectives and Approach

The purpose of the semantic model derived in this chapter is to formalise the notion of a trust domain in such a way that it can serve as a means of communication among parties involved in the establishment, design and maintenance of a trust domain. The objectives can thus be summarised as follows

1. Provide a common conceptualisation that can be used in the construction of trust domains and communication of their properties and characteristics among stakeholders
2. Provide extension-points that enable the notion of a trust domain to be integrated with other supporting or related concepts.
3. Re-use existing (related) models to develop a rich set of concepts that can be used to conceptualise the notion of a trust domain.

A trust domain is constructed using a combination of social and technical systems. Social mechanisms define the norms and responsibilities of entities (i.e. human), while technical systems provide mechanisms through which the activities of participants are controlled and monitored. Our aim is to identify those concepts that can be used to construct a trust domain. Such concepts

could exist in various forms and at different levels of abstraction. Therefore, to capture them we need to adopt an approach that enables us to describe these concepts not only at a particular level of abstraction but also in terms of the mechanisms needed to integrate the concepts across the layers of abstraction. In other words, the approach has to cater for inter-relations among the concepts at the same level of abstraction as well as across abstraction layers.

In order to capture the knowledge about trust domains, we utilise methods and tools developed in the context of the Semantic Web initiative. This gives us means of describing concepts within the application domain and relating them to each other. We employ the Protege tool [GMF+03] for formalising the concepts. In particular, this approach allows for further formalisation, e.g. by means of description logic [KMR04]. The model is constructed as a set of related ontologies (cf. [vR] for a definition of the term 'ontology'), which can be extended or refined as we develop a better understanding of trust domains.

3.2 Trust Domains: A Layered View

Throughout this chapter we adopt a perspective that permits us to discuss a trust domain as a set of layers. From top to bottom, the major layers concern the general tasks and activities that the trust domain addresses, the services deployed within it, and the technology underlying these services, each of which will be discussed in turn.

Processes or Social Layer. The processes or social layer is concerned with general tasks and activities to be addressed when operating a trust domain. This layer describes the purpose of the trust domain, that is, the supported activities. These are linked to corresponding business goals and lead to questions concerning the people-specific processes being supported within the domain. Activities supported by a trust domain can be thought of as being translated (either formally or informally) into a set of processes that are operated by the members of the domain. People and processes are supported by the resources of access devices, services and information. The following aspects are of interest from a top-layer perspective when processes are performed:

Actions by people and services can affect the type of an information item, e.g. the attributes regarding ownership, purpose, or security classification of the item. For example, a report in preparation may contain confidential information, but then a person involved in the preparation

may add personal information, which changes the type of the information.

Data-processing moves information between and through access devices and services. We therefore have to consider the way data spreads across these different elements and resources. One concept for following the spread of data is to consider it in terms of an *information surface*, i.e. the number of different ways to access an information item or parts thereof. Analogously to an attack surface, the information surface changes in the course of the processing of the data, e.g. by changes in ownership, transmission over unprotected channels, or storage in unprotected persistent storage..

Individual goals guide and affect the behaviour of the individuals and organisations that comprise a trust domain, thus affecting how they process information. For instance, an individual who is incentivised to deliver on a particular task before a deadline may take additional risks in handling information in order to achieve this goal, e.g., by sending the data to a person not authorised to receive it, or by using insecure mechanisms of storing or transmitting the data. The importance of this aspect has been highlighted by the empirical studies in Chapter 2, which have pointed out the importance of taking into account the tradeoffs that individuals may be forced to make.

A suitable model for the process layer must therefore take into account the following questions:

1. What are the basic constraints and properties that should be guaranteed by the particular trust domain?
2. What are the processes required for defining domain membership and for maintaining an appropriate set of policies?
3. Which people and roles are involved in setting up and maintaining the trust domain?
4. What are the necessary processes and information items to establish trust between a trust domain and its members?
5. How can trust be communicated between individuals and trust domains, i.e. how can individuals enquire about the trust status of the domain or items within it?

In order to tackle the first question, we need to think about a declarative high-level description of policies that need to be implemented to a sufficient standard by members and supporting services. The second question touches on the management and life-cycle of policies and membership. As trust domains can be dynamic, with people coming and going and the threat environment constantly changing, membership criteria and policies must accommodate for these changes. With respect to the third question, management of a trust domain requires the role of a trust domain owner as the one who is ultimately responsible for all tasks. As the tasks themselves can be complex and may require a wide variety of skills, it is likely that the owner has to delegate many of the corresponding activities to other individuals. Since trust establishment is a key concept, we need to understand what types of information (evidence) must be conveyed for a member to be allowed to join or connect to a trust domain, and to maintain their membership and connection. Question four concerns mechanisms for exchanging information about the identity and state of members and platforms in a trustworthy fashion, e.g. as credentials about attestation, run-time monitoring, and audit or management processes. A collaborative project may span more than one trust domain, or it may exchange information beyond its boundary with other trust domains. Here we need to consider processes for exchanging trust information as well as the provenance information associated with particular pieces of information. This aspect of establishing trust suggests that we need core trust domain services that help collect and store evidence.

Services Layer. This layer describes the types, organisation and orchestration of the services that form the trust domain. This layer concerns the technical realisation of the trust domain, and specifically explains how the policies and processes defined in the top layer can be implemented so that people can use the trust domain in such a way that its purpose is fulfilled and the data-protection requirements are met. Conceptually, this layer could be subdivided into an architectural layer and an implementation layer. It should be pointed out that in addition to services supporting business activities, a trust domain relies on a number of core services that are typically part of the infrastructure, such as utilities for assigning and changing ownership of information items, filtering rules for inter-process communication by means of signalling, resource sharing, or communication across the network. Support for business activities can be implemented through multiple different service providers (e.g. as Cloud services). Each of them would have to conform to some architectural standard

(or constraints). By means of core services the business services then provide appropriate assurance that they are trustable.

Infrastructure Layer. Below the service layer we find the infrastructure layer, that is, the actual technical implementation of the services in the trust domain. The properties of this layer can have a significant effect on the overall trust properties of the system. On the one hand, security mechanisms implemented at the technology layer are necessary to support trust at higher layers. On the other hand, security mechanisms may also get in the way of business (cf. Chapter 2), and may therefore be circumvented by users.

When designing a trust domain we need to think about how we refine policies and trust requirements from the high-level description down into system configurations, and technical and management controls, and how we create the evidence-trail that demonstrates that the participating entities are trustable.

3.3 Building Blocks

Since the notion of a trust domain is centred around information flow, it is important that we consider the channels through which information may flow in any set-up designed to support information sharing. Depending on the type of information intended to be shared, a trust domain may be enacted in one of three ways: i) purely in a social setting; ii) purely in a technical infrastructure; and iii) across the social and technical systems.

A trust domain can be described using some fundamental concepts that exist in both social settings and technical infrastructure. These concepts can be used to create trust domains of different types, enforcement mechanisms and, hence, varying properties. To understand the types, composition and properties of trust domains, we begin by characterising a trust domain using concepts from the two areas. We then identify the main building blocks of our semantic model.

3.3.1 Social Structures

Social structures provide a means of controlling, restricting and monitoring the behaviour of individuals and organisations. For example, organisations, professional bodies and associations define a code of conduct that specifies how its members are expected to behave, and the level of punishment brought

for any breach of the code. For instance, when sending a message by post, the message is transported by entities outside of the intended audience, such as the Post Office. But social structures (e.g. the code of conduct defined by the Post Office) prevent the postman and other entities from learning the contents of the message.

Social structures may also be used to influence the properties of a trust domain. Typically, there will be several kinds of social structures. The exact type used, and how they are used in a particular instance of a trust domain, will vary depending on a number of factors including the purpose of the domain, the properties desired as well as the environment in which such a trust domain occurs. Legislation and regulations play an important role in almost all trust domain instances. Contracts and organisational structures are used to hold participants accountable for certain aspects of a trust domain, while cultural values and risk perception are critical for determining the attitudes towards data sharing or accountability in a trust domain.

3.3.2 Infrastructure Support

The infrastructure serves to provide mechanisms that ensure that technical systems involved in or used as a means of processing, communicating or sharing information observe and enforce the required information-flow constraints. Such an infrastructure will comprise resources that store, transport or use the information to perform computations, as well as components that control how, when and where a particular piece of information can flow. The infrastructure should be viewed as a means of enabling data-sharing among a set of participants rather than as an end in itself. For this reason, different types of infrastructure may be used depending on other requirements such as performance, accessibility, ease of use and reliability, as well as the strength of security mechanisms desired.

The infrastructure consists of several layers of abstraction. At the lowest layer are the physical hardware devices such as desktop computers, servers and mobile phones. These devices interact according to the interactions defined in the system architecture and provide services to other layers. A number of other layers can be built on top of the physical devices. These higher layers serve as a way of abstracting away the finer details of the underlying technology and thus simplify the usage of the infrastructure. For example, in a Cloud computing model, the Infrastructure as a Service (IaaS) is considered to be at a lower level of abstraction on which the Platform as a Service (PaaS) is built. Similarly,

services in the Application as a Service (AaaS) layer serve as an abstraction mechanism built on top of PaaS. One thing that is common to all the layers of abstraction is that they each expose one or more interfaces through which their services can be accessed. In many recent distributed systems, the interface is defined using the Web Services architecture [BHM+04]. This architecture defines a resource as the main entity in the architecture and a service as a means of interacting with the resource, and can be used at any layer of abstraction.

In addition to the interfaces described above, a number of processes are normally set up to provide the necessary services. In the context of trust domains such processes must enable trustworthy operation. In other words, the components involved, and the processes they perform to set-up the services, must ensure that services are only accessible through appropriate interfaces, that the interfaces restrict access to authorised entities and that data-flow to all the interfaces is controlled. We base our model on the Trusted Multi-tenant Infrastructure defined by the Trusted Computing Group [TCG].

3.3.3 Modelling Frameworks

A trust domain can be deployed on different types of infrastructure. Each infrastructure may employ different technologies to serve the needs of a trust domain. However, at the architectural level, several aspects can be identified that are common among the various types of infrastructure. As discussed above, aspects of interest to trust domains include the web services architecture and the trusted multi-tenant infrastructure. In the following we discuss these frameworks in more detail.

Web Services Architecture Model The Web Services architecture [BHM+04] defines functional components, relations among them and constraints under which they operate. The architecture defines four models (message-oriented, policy, service-oriented and resource) that capture concepts and relations among them as viewed from a certain perspective. We collate the concepts defined in these models into an integrated architectural model as illustrated in Figure 3.1. The message-oriented model defines *Message* as a first-class concept that has a *Sender* and one or more *Receivers*. A message is delivered by some *Message Transport mechanism*, which is constrained by a *Delivery Policy*. The delivery policy is a subclass of *Policy*, defined in the policy model, which defines *Message Reliability* properties of the *Message Transport* mechanism.

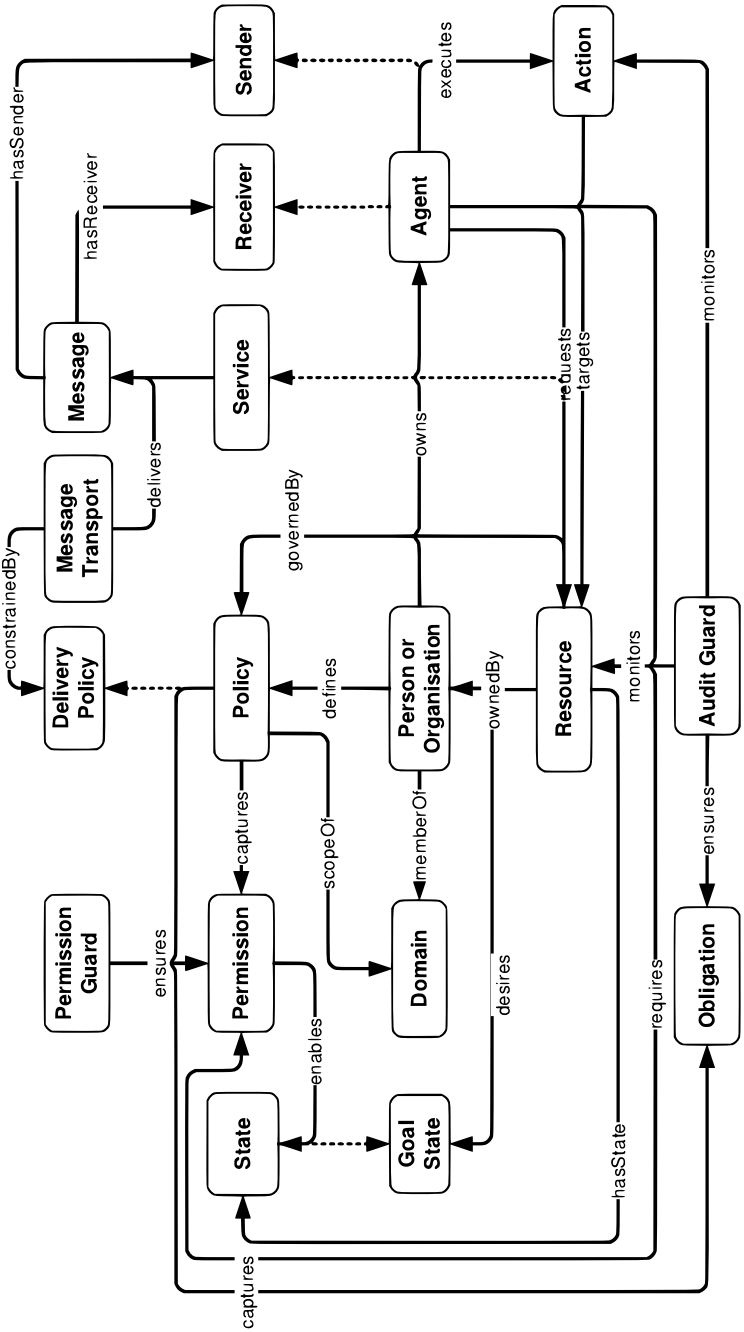


Figure 3.1: Concepts of Web-Services Architectural Models.

Messages are used as the main mechanisms for communicating service requests and request responses, where a service is defined in the service-oriented model as an abstract resource capable of performing one or more tasks. This definition of a service allows us to link a service to a *Resource* defined in the resource model through a subclass relation. A resource has a *State* and is owned by a person or organisation who defines a *Policy* that governs how the resource is used or behaves. This policy also defines the scope of a domain in a way that captures the desired goal state. Policies capture permissions that are required by certain agents in order to enable certain resources to attain some state, and obligations that agents have on the resources. Permissions are enforced by a *Permission Guard* while obligations are enforced by an *Audit Guard*.

Trusted System Domain Model The TCG TMI Working Group [TCG] aims to build a set of specifications for trustworthy operation in multi-tenant infrastructures. As part of this effort, the working group has released the first version of the use cases for a Trustworthy Multi-tenant Infrastructure (TMI). We use these use cases to build a model that includes concepts and relations among the concepts defined in them. The concepts capture the parts of a TMI while the relations capture the dependencies, causalities and interactions among the concepts necessary to capture the functionality of a TMI as described in the use cases.

The model, illustrated in Figure 3.2, is built around the three main entities *Policy*, *Asset*, and *Role*. An *Asset* is an entity that is of value to a person or organisation that participates in a TMI. Its value is defined by the entity that has an interest in it, implying that the same asset could have different values depending on from whose point of view this value is being computed. A *Policy* is a piece of data that defines the constraints on how the assets are used or accessed by entities within a TMI as well as the expected behaviour of such entities. A *Role* is a set of responsibilities assumed by an entity in the system. It also defines the activities that any entity with such a role can perform.

The trustworthy multi-tenant infrastructure distinguishes between two main roles, viz. the *Consumer* and the *Provider*, and defines several concepts, some of which are generic across domains of the two roles, while others are specific to each domain. A *Domain Audit Agent* is defined as a generic agent that forwards *Audit Events* to the *Central Audit Store*, a component that stores audit events collected from various parts of a TMI. The *Consumer Audit Agent* and the *Provider Audit Agent* are subclasses of the *Domain Audit Agent* that

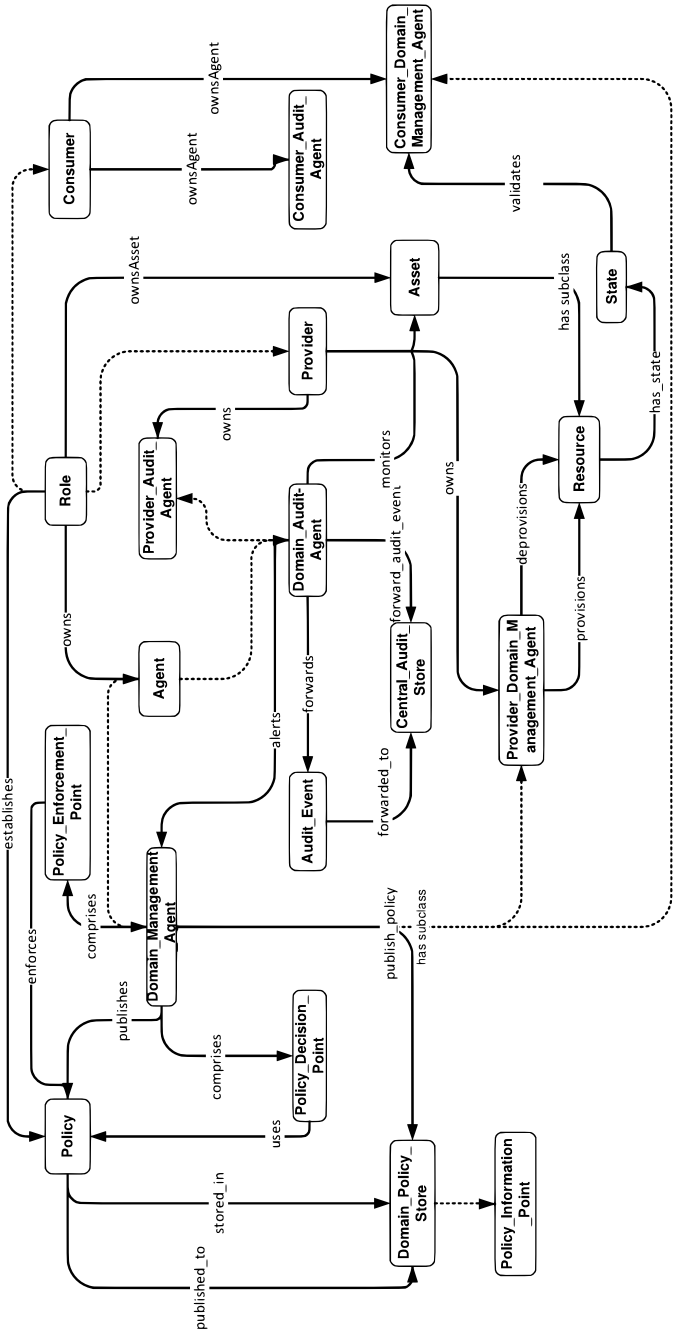


Figure 3.2: Concepts of Trusted System-Domain Models.

are specific to *Consumer* and *Provider* domains, respectively. Each role establishes a policy that is published to a *Domain Policy Store* and also creates a *Domain Management Agent* to manage the domain according to the policy they define. This policy is expected to be enforced by a *Policy Enforcement Point* in the domain in which the policy is set up, and is monitored by the specific type of domain-audit agent for the particular domain. *Consumer Domain Management Agent* and *Provider Domain Management Agent* are subclasses of *Domain Management Agent* that are specific to Consumer and Provider domains, respectively.

A *Resource* is defined as a subclass of *Asset* which has a *State* and can be provisioned or de-provisioned in the provider domain by the *Provider Domain Management Agent*. To ensure that the correct policy is being enforced on a resource, the *Consumer* can validate the state of the resource using the *Consumer Domain Management Agent.*

3.4 Semantic Model

In this section we discuss how the models identified in the previous section are combined to create a trust domain model. We illustrate the concepts that can be used to integrate the models and discuss how the semantic gap in the usage of these concepts can be bridged. We first discuss important concepts and relations, before integrating the concepts into the trust domain model.

3.4.1 Concepts and Relations

The proposed model consists of a number of concepts, such that each concept captures a class of things that may exist in a trust domain, be used to build a trust domain and used within a trust domain. Though all these concepts may be used in different instances of a trust domain, a few of them can be characterised as being fundamental to the existence of a trust domain. We identify the concepts of Actions, Assets, Policies, Controls, Roles, and Evidence as being fundamental:

Actions are series of functionality performed by components and agents in a trust domain. An action typically either consumes or produces some data or results in the change of state of a particular component or system. Depending on the level of abstraction desired, an action can be divided into sub-actions, where each sub-action contributes to the overall outcome of the parent action.

Asset Our conceptualisation of a trust domain is based on the idea of enabling secure information-flow among a set of entities. Such entities may each have a set of devices through which they share the data. Furthermore, these entities may provide access to the information stored on the devices or other media to other members of the domain. For this reason, we define the concept of an Asset as being a fundamental element of a trust domain. An asset is something of value to the owner, but could also be valuable to other entities such as attackers or competitors. One example of an asset is data.

Data may exist in many different forms, and indeed each form is faced with various kinds of challenges with respect to controlling how it flows. For example, printed material could be prevented from being taken out of the building. However, it is still possible that someone could scan the material and send it over a network or copy it to a USB stick. For this reason, different types of protections maybe required to overcome each type of challenge. To reduce the scope of our work, we limit the model to data that exists in digital form. Other examples of assets include resources such as computers, services such as web services, and communication infrastructure such as networks.

Policies are a means of specifying the behaviour of entities within a trust domain, and how data flows within or outside of a trust domain. Policies describe the required or expected behaviour of processing elements with respect to information resources as a set of assignments, constraints, and rules. Each policy specifies the expected relationship/dependency between one or more entities within a domain. For example, a policy can be used to specify the data allowed to flow into or out of a trust domain, the characteristics of the controls that should exist in a trust domain and the kind of evidence that these controls should provide. To satisfy the requirements for a trust domain, it must be made explicit how each policy is enforced and how decisions following from the policies are made.

Controls are a set of mechanisms, processes or procedures that enforce the policies within a trust domain. These controls could be accomplished through social or technical means, e.g. penalties or trusted computing, respectively. Controls monitor activities that occur within a trust domain and produce evidence, described below, that can be used to determine the properties of a trust domain or its constituents.

Roles are used to specify the level of participation in a trust domain. Each role defines the types of activities that an entity assuming that role can

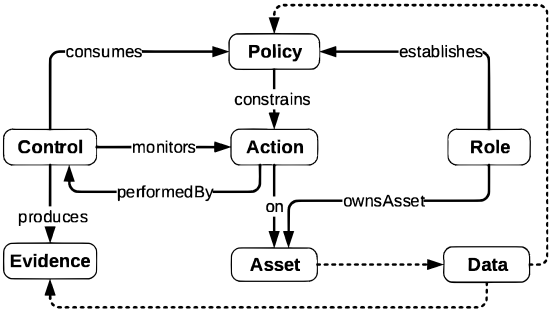


Figure 3.3: Concepts and Relations in the Conceptual Model.

do as well as the types of behaviour that entities would be accountable for. Roles are also a way of separating concerns in a trust domain and isolating the activities that participants can perform.

Evidence is data that is produced by the controls within a trust domain to indicate the kinds of activities that have occurred. These activities are captured by monitoring the actions that are performed by or on behalf of roles that exist within the domain. Examples of such evidence include provenance, i.e. records of how data came to be, audit logs, i.e. logs of events that occur during the lifetime of a trust domain, integrity measurement lists, i.e. a record of the binary hashes of software components and data, and digital certificates, i.e. cryptographic identities of components that perform certain actions.

The relationships between these concepts are illustrated in Figure 3.3: A Role owns Assets that will exist within a trust domain and establishes a Policy that constrains Actions. The act of defining a Role establishes one or more policies within the domain. However, any given policy can only be established by one role. This means that if two roles establish identical policies, then both policies are treated as unique entities, which can be linked through the equivalence property. Actions are performed by a given role or by some agent that represents a particular role. These actions are monitored by Controls to ensure that the policy is being upheld. These controls produce Evidence to indicate that actions have been performed in accordance to the policies. Both Evidence and Policy can be considered to be a form of data which can be manipulated in

the same way as other data and may be subject to the same information-flow restrictions.

3.4.2 Integrated Model

The fundamental concepts discussed above exist in more than one of the infrastructure modelling methodologies discussed in Section 3.3.2. To build a model for trust domains that makes use of these models, we need to understand how the models can be integrated. In order to achieve this, we use the fundamental concepts as integration points, so that each concept that exists in more than one model is linked using equivalence. This allows all the relations that apply to a concept in a particular model to apply to an equivalent concept in another model. As an example, Policy, which is defined in the fundamental model, exists in the Trusted System Domain (TSD) model as well as in the policy model of the WS-architecture. This allows us to relate the policy concept in the TSD model to the Permission concept in the WS policy model through the equivalence property between the two concepts.

Using this approach, however, we are faced with the challenge of a possibility of conflicts and inconsistencies. These must be resolved depending on the requirements for the concrete system that was modelled. For example, in the WS policy model, the policy is defined by a person or organisation while the TSD model specifies that a policy is established by a Role. To address this issue, we first identify a possible relationship between person or organisation and role. Using this relationship, we can then determine the appropriate concept to be linked directly to the common concept, i.e. Policy. In this particular case we decided on Role because it is possible that other entities such as autonomous systems might be able to set-up a policy.

In the following we describe the integrated model of a trust domain that results from applying the above integration approach. The model is illustrated in Figure 3.4: We define a *DomainEntity* as a first-class entity in a trust domain. It is defined as an entity that has a *memberOf* relation to the *Domain* and has exactly one of the following types: *Person, Organisation, System, Process, Resource* or *Agent*. Furthermore, each domain entity has a *Role* within the domain. In this model we do not restrict the membership and the role relationship, so that a domain entity may participate in more than one domain at any given time and may assume more than one role.

We define a *Policy* as data whose scope is limited to a Domain. In other words, a policy is only effective within the domain. It does not directly influence the behaviour or properties of entities outside the domain.

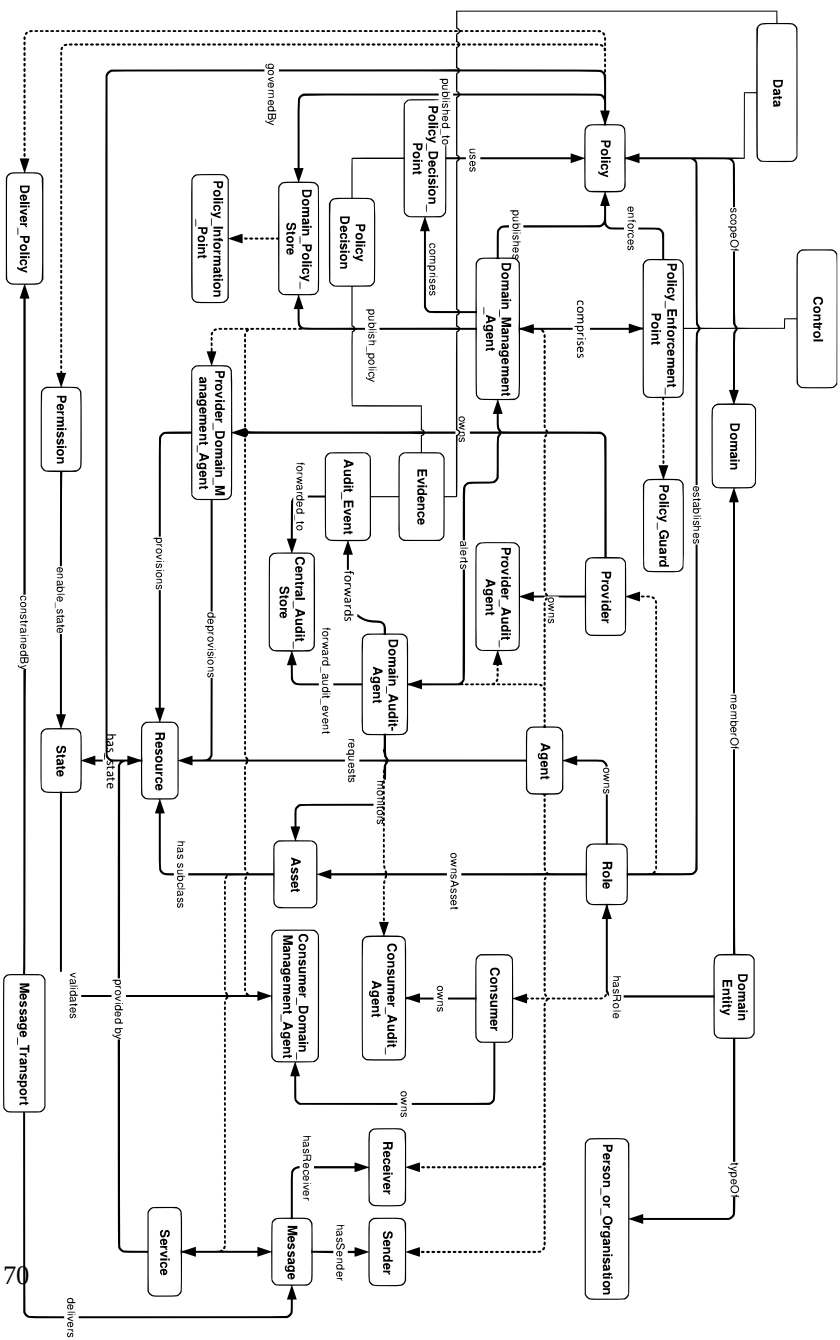


Figure 3.4: Integrated Semantic Model for Trust Domains.

Another important question relates to the definition of identical domains, i.e. domains that enforce exactly the same kinds of policies. To answer this question, we define a Policy to be singleton object but allow it to be cloned, i.e. copies can be made for use in other domains. The cloned policy can be linked to the original policy through the equivalence property. The policy is consumed by the *PolicyDecisionPoint*, which creates *PolicyDecisions*. Policy decisions exist in the form of *Permissions* or *Obligations* and are enforced by a type of control called *PolicyEnforcementPoint*. The policy decisions can be considered as a kind of evidence, which together with some policy meta-data can be used to express and communicate how the policy has been broken down into enforceable constraints.

An important aspect in trust domains is the ability to relate decisions to the policies that triggered these decisions. For example, when a constraint specifies that a certain action is permitted, trust domains must be able to demonstrate how that constraint (i.e. to permit an action) was reached and the policies which influenced this decision. For this reason we define *influenced* as a relation between the set of policies and the set of decisions which were used to arrive at the decisions. For this to work we require a policy to be consumed by a *PolicyDecisionPoint* that creates the *PolicyDecision*.

A *Role* may also own *Agents*, which may act on their behalf, and *Assets*, which are shared with other members of the domain. Assets have a type, i.e. *AssetType* which could either be *Resource*, *Data* or *Service*. Furthermore, the model specifies that individual assets and agents must be owned by exactly one role.

Actions are performed on Assets by domain entities and Controls within the domain. However, before an action can be performed, certain constraints determined by the policy must be satisfied. For this reason we specify that Policies constrain Actions. To enable checking that appropriate constraints are satisfied before an action is performed, controls within the domain monitor actions.

Resources are an asset type which has state. A resource can be provisioned or de-provisioned by controls within the trust domain. When a resource is provisioned, it becomes available for use by agents and domain entities who request for Assets (including Resources) in the domain. The state of the resource also plays an important role in determining the properties of a trust domain.

We therefore allow state to be validated by a *DomainManagementAgent*. This validation determines whether or not the behaviour or state of resources

is in line with domain policies. A special type of control, referred to as *DomainAuditAgent*, is responsible for generating audit events by monitoring the activities in the domain. The *DomainAuditAgent* forwards *AuditEvents* to a *CentralAuditStore* where they can be analysed as part of evidence to determine the properties of a trust domain, and alerts the *DomainManagementAgent* when certain critical events are observed.

We define messages as the main mechanisms for accessing services provided within the domain. Each message has a *Sender* and one or more *Receivers*, both defined as agents and is delivered through some MessageTransport mechanisms such as RPC or web services, which is constrained by a *DeliveryPolicy*. The model defines a service as a Resource which may be provided by another Resource.

3.4.3 Formalisation of the Model

The model discussed above consists of a number of concepts and relationships. We capture the concepts using OWL (Web Ontology Language) Classes, and define relations as object properties. Policy, Asset and Role as Classes are defined as main entities, and a number of properties relate these concepts. The `ownsAsset` relation defines an $1 : M$ relationship between a role and an asset, so that a given role can own multiple assets. We capture this as an OWL inverse functional property so that for every object (i.e.policy) in this property, there exists a single subject (i.e. role). We capture the established property in a similar manner, so that one role can establish multiple policies but each policy can only be established by a single role. Properties that specify a functional relation are captured as OWL functional properties.

3.5 Using the Model

The model described in the previous section can be used for a number of purposes. In this section, we describe some of the essential uses of the model.

Using the Model as a Vocabulary The model includes the essential constituents of a trust domain. This makes it useful as a vocabulary for discussions on the nature of trust domains. For example, in the collaborative system, a number of components are included to ensure that policies specified by infrastructure owners are enforced by relevant job execution mechanisms. Figure 3.5

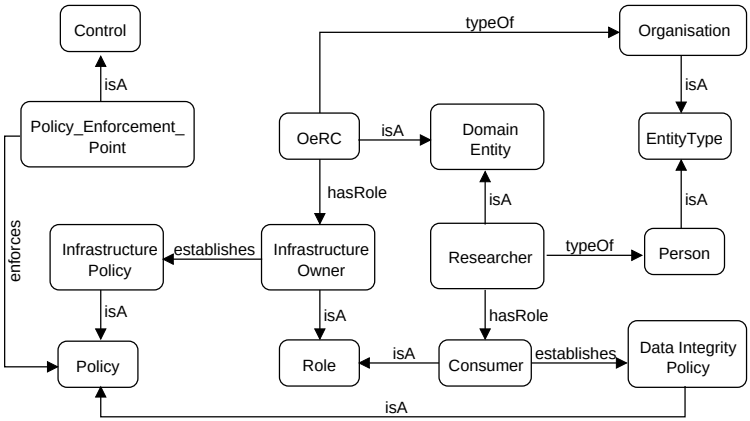


Figure 3.5: Example usage of the model as a vocabulary for the collaborative system in [ANM13, KM12]

illustrates how a vocabulary can be created from the model. Both researchers and OeRC can be defined as being domain entities.

In [KM12, ANM13], the Oxford e-Research Centre (OeRC) can be defined as an organisation that assumes the role of infrastructure owner. The infrastructure owner establishes an infrastructure policy to constrain how computation and storage resources (an asset owned by the infrastructure owner) are used or accessed. A researcher is a person who assumes the role of consumer and sets up a data integrity policy. Using the model as a vocabulary has a lot of advantages including enabling communication, providing a common understanding among stakeholders, and inference of new knowledge.

Instantiating Trust Domains In any given situation there will be multiple trust domains, each utilising different types of entities and enforcement and monitoring mechanisms, but sharing certain properties due to their nature. For example, two trust domains that both use different types of firewalls, each configured in a different way, will still share certain properties, such as enforcement being based on traffic filtering, due to the way firewalls operate.

A trust domain model helps to capture those properties that may be shared due to the nature of entities used to construct it. It provides a characterisation

of the common entities and their properties and thus enables trust domains of varying properties to be instantiated from a common set of entities.

Comparing Trust Domains The use of a common set of entities to instantiate trust domains provides an opportunity to determine the differences between trust domains. For example, if two trust domains are constructed such that the only difference between them is the type of domain entities (e.g. humans vs. processes) allowed, then a comparison can be made by focusing on the properties of the domain entities.

Inferring New Knowledge One distinguishing characteristic of trust domains is their ability to make explicit both the intentions of the domain owner and the consequences of assuming certain configurations. For example, when a target company in an M&A scenario (cf. Section 1) sets up a policy to control how its data (e.g. intellectual property) is used by the bank, a trust domain makes explicit what part of data is affected by the policy as well as consequences of applying this policy, such as how accessible the data becomes or the impact on the performance. The model serves as an important tool for inferring consequences of certain configurations and new knowledge about the attributes of a trust domain resulting from these configurations.

The model developed here is based on a high-level understanding of a trust domain. As this understanding may change in response to new developments or when applying the concept to specific scenarios, it is important that the model be extensible. We achieve extensibility by building the model on existing Semantic Web concepts that are designed to support extensibility and interoperability. The model can be extended in various ways including addition of new concepts, specialising existing concepts, specifying new inference rules and linking existing concepts to related concepts in other application domains. In the remainder of this section we discuss how each of these extensions can be achieved.

Specialising Existing Concepts An existing concept could be found to be too coarse-grained for a particular application domain. For example, the concept of a Resource could be further broken down into specific types that are useful for a given application domain. This is achieved by specifying the specialised concept as a subclass of an existing concept that is being specialised.

The new specialised concept will inherit the properties of the superclass, which may be overridden if necessary. New relations can also be defined to link the new subclass to other concepts in the model.

Specifying Inference Rules Inference rules are a way of specifying how new knowledge can be obtained from knowledge in the model or instances of the model. Rules are specified as relationships (conjunction and disjunction) between one or more properties (antecedents) and one or more statements (conclusions) such that when all the antecedents are true, then the conclusions will also be true. This may help, for instance, to identify inconsistencies between policies.

Specifying New Concepts A concept is used to denote an identifiable class of things. All the concepts in the model are defined as OWL classes. Therefore, to add a new concept to the model, one defines a new OWL class giving it an identity. Such an identity must be unique to avoid confusion. Once the concept has been identified, it has to be linked to existing concepts. For example, a new concept such as Contract could be defined to denote a set of legal contracts. Such a concept could be considered a type of data and therefore must be linked to the Data concept. Furthermore, it might be necessary to indicate that a contract is different from a policy, i.e. a contract should be linked to the Policy concept through the Disjoint property, which captures the idea that any given instance cannot both be a policy and a contract.

* * *

In this chapter we have discussed how a semantic model for trust domains can be defined. The model is based on existing modelling methodologies for describing systems in an ordered way and can be used to structure the thinking about trust domains in particular scenarios as well as for guiding the implementation.

The concepts described here are based on the work presented in Deliverable 3.1 of the Trust Domains project [MKBN12] and in [ANM13, AM14]. For a much more extensive discussion we refer the reader to these publications, particularly Deliverable 3.1.

Chapter Four

Components for Trust Domains

In this section we provide an overview of existing technologies that can be used as components for Trust Domains. Our focus is on Cloud-based technologies. We identify technologies and components that can help implement the main characteristics of trust domains in the cloud. These are:

1. Control of information-flow to help ensure that information flows to the right entities.
2. Comparison of observed behaviour against expected behaviour, in order to help determine whether the entities composing the trust domain behaved and currently behave as required. This necessitates collecting evidence demonstrating the nature of the behaviour that an entity had and currently has.
3. Membership and access control to help ensure that only entities that participate in the trust domain can access the resources shared within it.

In our discussion we distinguish between mechanisms for monitoring and mechanisms for enforcement; additionally, we distinguish according to whether these mechanisms apply to data or to processing components. Table 4.1 provides an overview of the approaches discussed in this chapter according to this structure.

The chapter is organised as follows: We first discuss existing technologies for enforcement and monitoring focussing on processing nodes and on data. We then discuss monitoring of device behaviour using a specialised hypervisor.

	Processing Elements	*Data*
Enforcement	Data-Path Isolation	Data-Storage Isolation Sticky Policies
Monitoring	Assurance based on Trusted Computing, Hypervisor-based Monitoring	Provenance-based approaches

Table 4.1: Overview of Components for Trust Domains, structured by the objects that they apply to (Processing Elements or Data) and the type of mechanism (Monitoring or Enforcement) that is applied.

4.1 Enforcement

Trust domains are designed to provide assurance that data exchanged between collaborating parties flows to the right entities and does not leak. For that, a trust domain's data and information flows should be isolated from other flows. Similarly, when trust domains operate in the Cloud, their resources need to be isolated from the resources created by customers who do not participate in the trust domain, and their information-flows should be isolated from other customers' information-flows.

Multitenant Cloud infrastructures are designed to provide an 'end-to-end logical separation between different tenants' [JOP12], where a tenant is a customer. Therefore, they rely on techniques that could help provide the type of isolation required for a trust domain to operate properly. In this section we study how mechanisms for isolating tenants in multitenant Cloud infrastructures could be leveraged to build Cloud-based trust domains. We focus on virtualisation, as this technology provides the logical separation of resources that are physically collocated. At the application level, virtualisation allows tenants to run their applications on the same physical host but in the same way as if the applications were run on different hosts. To this end, applications run in virtual machines (VMs) which provide them with isolated OS environments. However, as VMs located on the same physical host can have their data sent on the same LAN [HLMS10], the level of isolation provided by VMs is not sufficient for isolating tenants' data traffic. Besides this, applications may need to access data which, if not isolated, might be accessed by a different tenant than the one to which the data is associated. In this section we survey some of the technologies and techniques used to provide path isolation and data isolation.

Data-Path Isolation Path isolation can be provided by creating virtual networks between VMs and devices. Here, we summarise approaches to create virtual networks that were proposed by VMWare [VMW] and Oracle [Ora12, Ora, Ora10].

In a VMWare infrastructure, *VSwitches* (virtual switches) provide a virtual network infrastructure. VSwitches located on the same physical host do not share physical Ethernet adapters and cannot be interconnected. This enables a strong level of isolation between information flows by specifying that only VMs associated with a given trust domain should be connected to the same VSwitch. *Port groups* capture all the settings (Virtual switch name, VLAN IDs and policies for tagging and filtering, teaming policy, Layer-2 security options, traffic shaping parameters) to provide 'persistent and consistent network access for virtual network adapters' [VMW]. If the VSwitches to be used by the VMs associated to a given trust domain are assigned the same port group(s), then it could help ensure that trust domains are always assigned the same network access independently of the physical host on which they run. A virtual port group on a virtual switch ensures that frames do not leak to a different VLAN. This can help ensure that trust domain data does not leak. In *VMware ESX servers*, virtual ports on VSwitches 'know authoritatively what the configured receive filters are for virtual Ethernet adapters attached to them' [VMW]. They also 'authoritatively know the "hard" configuration of the virtual Ethernet adapters attached to them' [VMW]. This may help define policies based on the 'hard' configuration of the virtual Ethernet.

In Oracle servers, virtual machines are referred to as *domains*. Each domain can be assigned a role which constrains the access rights granted to the domain. This can help control and monitor the manner in which VMs access physical resources. *Logical Domain Channels (LDCs)* are full-duplex point-to-point communication links providing a data path between guest domains and virtual devices [Ora10]. The hypervisor creates one LDC for each link. This technology can help build isolated communication links between VMs and therefore help prevent information from leaking.

Data Isolation To prevent data from being accessed by unauthorised tenants, security mechanisms controlling data access are used and, in multitenant Clouds, additional data isolation techniques are implemented. In this section we focus on three data-isolation techniques that can be used to isolate the data of a trust domain. This discussion is based on [CCW06].

Separated databases A first data separation approach consists in storing tenants' data in separated databases. Then, metadata can help associate each tenant to the database where the respective data is stored. This approach simplifies the management of tenants' data as the modification of a tenant's data model or the restoration of a tenant's data following a failure does not have any impact on other tenants' data. However, as the number of databases that can be hosted on a server is limited, this approach has high hardware costs.

Applied to trust domains, this approach makes it possible to isolate data associated with a trust domain from data associated with other customers. Within the same trust domain, it also makes it possible to isolate data associated with different participants by assigning them their own database tables.

Separated schemas Another approach to provide data separation consists in storing tenants' data in the same database and in associating each tenant with its own database schema. Each schema regroups the corresponding tenant's tables where his data are stored.

This approach has lower hardware cost than the one described previously and also makes it possible to extend a tenant's data model without impacting other tenants' data. However, restoration of a single tenant's data after failure can have an impact on other tenants' data, as a roll-back often affects the whole database. Therefore, this approach is only considered to be suitable for cases where tenants need a small number of tables, i.e., less than 100 tables. Finally, compared to the isolated-database approach, this approach may require more complex mechanisms to be used in order to secure tenants' data. Within the same trust domain, this approach may also allow participants' data to be isolated in different tables provided that the number of participants is low. Otherwise, isolation of participant's data may need to be managed at the record level. Each participant could be assigned a unique identifier to be associated to all his records.

Shared schemas A third approach to provide data isolation consists in storing all tenants' data in the same database and in the same tables. Each tenant record can then be identified with a tenant-specific identifier.

This approach has the lowest hardware cost. However, if there is a large number of rows in the tables and a large number of tables, restoration after failure can have a bad effect on performance for all tenants having

their data stored in the database. Besides this, the approach may require more effort when security mechanisms are put into place to protect data access.

The isolation offered by separated databases is very strong, because there is no database-level path between the different applications/domains. In this case, the isolation is only likely to be compromised by a poorly-implemented higher-level service. Separated schemas also offer strong isolation, if the higher-level services are suitably constructed, but are subject to compromise through the database management system. In particular, backups and other tools for managing dependability will typically have access to all of the tenants' data. A shared schema requires the greatest care during implementation and the greatest reliance on fine-grained access controls, and so the isolation must be considered inherently fragile.

4.1.1 Sticky Policies

The 'sticky policy' paradigm was introduced by Karjoth et al. [KSW03] in order to ensure that end-users' preferences on how their data is to be accesed can be enforced wherever the data flows. A sticky policy is a data structure in which primary data is associated with a policy (typically an access-control policy) so that the two are transmitted and stored together and the latter can be applied to the former regardless of the processing platform (or even if the data migrates to another enterprise [KSW03]). Applied to trust domains, sticky policies can permit policies, specifying how data should be accessed by participants, to be continuously enforced. This is needed when data accessed in a trust domain is used outside the domain, in participants' organisations for instance. They could also help enforce that when participants participate in multiple trust domains, data that is accessed in different trust domains remains isolated on the participants' computing platform.

When introducing the sticky-policy paradigm, Karjoth et al. highlighted that for sticky policies to be enforced policies should always be associated with the data they apply to. Kounga and Chen [KC12] highlighted that two additional requirements should be fulfilled: (1) Data should only be accessed under the conditions specified by the associated policies, and (2) on a computing platform, data should only be accessed by applications that are able to enforce the policy. These additional requirements imply that computing platforms which received data should be continuously monitored to evaluate whether the conditions in which data is accessed fulfil the associated policies.

Many approaches have been proposed to enforce sticky policies, Karjoth et al.'s approach does not enforce that data is accessed only as specified by the policies, instead, it is a 'best effort' approach. To avoid this, Tang [Tan08] and Casassa Mont et al. [CPB03] propose to use cryptographic techniques in order to bind policies to the data they apply to. These approaches allow data to be securely sent to a computer platform which fulfils specific requirements. However, they do not ensure that access on the destination platform is effectively constrained by the policy. In order to solve this limitation, Kounga and Chen [KC12] propose an approach which relies on the Trusted Platform Module (TPM) and virtual machine technologies to provide an end-to-end enforcement of sticky policies. The latter makes it possible to run applications in isolation and to provide access to data only to those applications that satisfy the associated policy. In order to achieve this, applications are continuously monitored and are authorised to access data if they are in a valid integrity state and if they have a conformance certificate demonstrating their capability to use data as specified by policies. Before Kounga and Chen, Sandhu and Zang [SZ05] also proposed a similar approach. The main difference between both is that Kounga and Chen's approach also relies on the application's capabilities in order to determine whether access to data can be granted.

4.2 Monitoring

A trust domain is composed of entities that 'have an expectation of and exhibit shared and predictable behaviour to protect the resources'. This definition highlights the need to evaluate the behaviour of the entities composing (or which are expected to compose) a trust domain in order to identify whether they exhibit the expected behaviour. This further highlights that:

1. Each entity participating in a trust domain should generate evidence that can be verified by other entities
2. Each entity participating in a trust domain should generate evidence that characterises its capability to protect resources
3. Each entity participating in a trust domain should generate evidence that can be compared to expected characteristics to evaluate the capability of an entity to behave as expected
4. Mechanisms should be in place to collect the evidence needed to evaluate whether an entity behaved as expected in the past and is currently behaving as expected in order to protect the trust domain's resources

5. Mechanisms to ensure that only entities exhibiting the expected behaviour can access the trust domain's data must be put into place.

Evidence generated in trust domains should be non-repudiable, that is, entities should not be able to deny having sent some evidence. In order to achieve this, each entity participating in a trust domain should have a public/private key pair, and mechanisms for managing keys should be put into place. Trusted Computing may help with this task; furthermore, Trusted Computing may also generate evidence itself.

Trusted Computing To provide the assurance that the evidence generated by an entity properly describes its behaviour, it is important to verify that this entity has not been tampered with. Trusted-Computing technologies provide the means to achieve this. Trusted computing is a paradigm developed by the Trusted Computing Group (TCG) 'to enforce trustworthy behaviour of computing platforms by identifying a complete chain of trust' [LM10]. This is achieved using Trusted Computing technology centred around the Trusted Platform Module (TPM). TPMs are tamper-resistant computer chips capable of securely storing cryptographic material (encryption keys, certificates and passwords), generating cryptographic keys, digital signatures, cryptographic hashes and performing encryption. TPMs also provide three roots of trust: the Root of Trust for Measurement (RTM), the Root of Trust for Storage (RTS) and the Root of Trust for Reporting (RTR). The RTS is a trusted implementation of a shielded location for storage of integrity measurements (in the form of cryptographic hashes over data) and cryptographic keys [Mar08, Tru07], whereas the RTR is a trusted implementation of a shielded location for reliably reporting the information held by the RTS [Mar08, Tru07]. The RTM is the computing engine trusted to make reliable integrity measurements [Tru07]. It is also the root of the chain of transitive trust, where transitive trust designates the process of extending trust to a second group of functions if the description of that second group by a trusted group of functionality is valid. The authenticated boot process relies on this principle to allow a program to evaluate its integrity as well as the integrity of the components it relies on [Mar08]. In order to do this, all pieces of code that are executed on a platform from start-up are measured by hashing their code.

Provenance Provenance can be described as 'information that helps determine the derivation history of a data product, starting from its original sources' [SPG05]. Therefore, provenance can help audit the processing of data in

a trust domain. In [APM08] Aldeco-Perez and Moreau propose a provenance-based architecture for auditing as well as a provenance-based application for auditing the processing of private data. They highlight that to establish such provenance, additional information describing what occurred at execution time should be collected by applications. They propose a methodology for creating provenance-aware applications [APM08]. Lyle and Martin [LM10] demonstrate how trusted computing technologies can be used to provide trusted provenance. In a trust domain, trusted provenance might be needed to demonstrate that provenance data was generated by trusted entities. Namiluko and Martin [NM12] also use trusted computing technologies. They propose a 'provenance-based model for reasoning about a system's ability to satisfy trust properties of interest'. This work could help the definition of models for trust domains. Moitra et al. [MBCD09] consider information assurance and specify a framework allowing 'wrappers,' containing provenance information, to be added to messages, e.g. a message identifier, a timestamp, or the hash of the message and the algorithm used to generate the hash. In a trust domain, Moitra et al.'s approach might help generate evidence demonstrating the manner in which messages where exchanged between participants.

4.3 Hypervisor-based Monitoring

Throughout the discussion in this chapter we have repeatedly observed that trust domains require mechanisms to monitor the behaviour of components. Monitoring can be implemented in various ways, depending on the usage scenario, but should generally work with as little influence from the system as possible. If an operating system (OS) goes wrong during its lifetime or becomes tampered, such that security properties are not met, it may no longer be considered trusted, and neither would any monitoring that relies on the OS.

Suspicion is usually aroused when we see something that we do not consider to be normal or expected. Computer systems typically do not spend time to reflect on whether what they are doing is expected by the user. The field of anomaly detection has concentrated on being able to spot inconsistencies within systems and networks that do not match 'normal' behaviour. However, many approaches have been riddled with a large probability of false positives. A small set of anomalies and outliers may be insignificant, but correlated with other types of anomalies can provide a more informative view of system behaviour. In the past there has been work in 'symptoms of malicious behaviour' which represent indicators of unusual behaviour [HBA+12]. More observable

signals could provide more confidence in establishing that a particular event has occurred, but more importantly it must help tell whether the system is still safe to use.

The focus here is on a subset of system integrity where select properties of the system can be observed and measured repeatedly during the lifetime of the system. If a client machine on a network is not operating as expected, this may not necessarily be a big indicator of compromise. Nevertheless from an administrative point of view someone would want to know about this type of behaviour as soon as possible when it occurs. This notion can even be extended to the users of a trust domain, by providing a service that monitors the status of each client machine and thus both informs the users and complements the security of the underlying architecture.

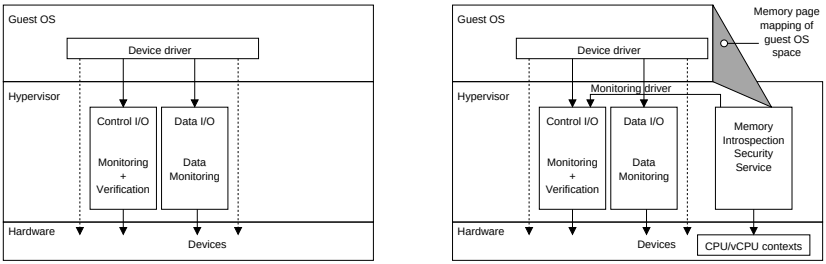
We consider two kinds of changes during runtime: Some changes be *reversible* to regain trust. If a hash of a particular application's executable code changes, this indicates that the program has been altered, and the level of trust should drop until the code is restored to its original state. A further example could be a process that must always be running in order to keep the trust level high. In contrast, an *irreversible* change could be tied to particular system-level components where integrity is critical and of the upmost importance, such as kernel modules or drivers. Once these have changed, the trust level can never go back up unless system is restored to a well-known 'good' state. Reversible and irreversible changes can be driven by policy per component.

In this section we outline a specialised hypervisor that provides secure introspection and monitoring of virtual machines. Hypervisor technology is especially suitable for the following usage scenario: For some sensitive data the only acceptable way of sharing may be through a virtual 'clean room', i.e. a locked-down virtual machine that only allows access to specific programs and services. This virtual machine can be distributed to the user along with the monitoring component implemented as part of a hypervisor, and the monitoring component can then be used to detect compromises to the virtual machine and to report whether the correct virtual machine has been started.

The main practical properties of such a component should include:

- A small Trusted Computing Base (TCB) to minimise the number of security vulnerabilities and for better verifiability
- Strict isolation between monitored OS and security software
- Reliable network communication for reporting level of trust

In order to provide monitoring, we need a secure base to build upon and use it to be able to detect system behaviour without compromising the isolation



(a) Architecture for endpoint hypervisor monitoring.

(b) Architecture with added out-of-band monitoring for physical memory and devices.

Figure 4.1: Endpoint hypervisor architectures.

between the operating system and the TCB. To work with existing Virtual Machine Manager (VMM) software, we design a new component which operates alongside hypervisors to perform monitoring of system behaviour on a platform, whilst keeping a small TCB that can be verified. The hypervisor uses different ways to passively monitor a guest OS using Virtual Machine Introspection techniques, described later. Keeping the hypervisor small has been seen as an effective approach to improving reliability of the hypervisor and the TCB (Trusted Computing Base) [MMH08].

4.3.1 Foundations for Secure Monitoring

Mainstream processors include extensions to increase throughput and performance of virtualised platforms. These extensions help in the scenario when the OS requires executing a privileged operation or access to a sensitive area of memory, which causes an event trigger. Execution is then switched momentarily to the hypervisor to handle the request.

Some processors include a pre-emption mode built into the processor, which after setup by the hypervisor, allows for switching between an untrusted execution environment and the secure execution mode based on the amount of time spent executing the guest OS. This feature can be used to switch to hypervisor mode periodically so that the monitoring software can run without the possibility of being disabled. This allows for the guest OS to run with full resources and keeps the monitoring running reliably through regular instantia-

tion without requiring a whole separate virtual CPU core. The trade-off could potentially be explored in future work to see if pre-emption in the processor fully satisfies isolation requirements.

Several monitoring drivers have been implemented, which allow for monitoring of raw control I/O and data I/O channels of devices, such as a disk drive and network card (see Figure 4.3). Since the information of these channels is raw, the monitoring drivers are small and do not contain high level logic or analysis which would affect performance of the monitored system. Furthermore, this could be used on devices that are not fully virtualised, and could yield better overall throughput than a fully virtualised device model. One of the open research questions is to reduce the large semantic gap between the raw data and high-level behaviour. Another goal is to not impact performance of the guest OS running on the machine. Monitoring is easily prone to be extremely CPU and disk intensive, so performance is critical in order to not affect the experience in the guest OS. We have built an experimental hypervisor upon which the results of this work are based.

4.3.2 Introspection and I/O monitoring

Introspection is a mechanism used by a trusted component on a platform to monitor at runtime another system running on the same platform, enabling the trusted component to safely and reliably extract dynamic properties of that running system. The memory spaces and execution contexts of these components are isolated and enforced by the hypervisor, which use to perform runtime integrity checking whilst protecting the trusted component.

We want to perform monitoring of activity within a guest VM. Using introspection we can achieve this goal due to having no dependency on the observed system and due to being outside the attack range of a malicious user. The definition of introspection is based loosely on the ability to reflect on oneself. In this case, the machine is looking at itself. The system boundary of the machine covers several sub-components, one of which is looking at the others looking for problems. The term originated with work by Garfinkel and Rosenblum, who created the first host-based IDS (intrusion detection system) based on Virtual Machine introspection [GR03]. This was the first platform that was able to observe the running state of a guest OS from outside its VM. Here, we leverage this technology to enhance reflection of trust in trust domain platforms.

Runtime introspection generally covers complete read access to memory and the contents of processor registers. Network and disk traffic in certain

cases can also be monitored. Since the field of introspection can cover a wide range of devices and use cases, we focus on the design of two subcomponents of our hypervisor implementation. These components perform memory analysis and monitoring of disk activity.

Memory Analysis. We introduce a Memory Introspection Security Service as a sub-component to perform secure logging and measurement of OS state. Mainstream operating systems keep symbol tables to help driver and kernel developers debug the system. Each symbol corresponds to a kernel virtual address in memory.

When initialised, the introspection service waits for a short period of time as the OS boots in order for the kernel to load all the symbols into memory. This is the entry point into the OS context. The introspection service then scans for the symbol table in memory. It is worth noting that some specific operating systems may leave a pointer to the symbol table in a CPU register during boot. Upon finding the symbol table, the introspection service needs the ability to translate virtual addresses to physical addresses in memory. This is because the introspection service is not running in the OS it is monitoring, and thus it must translate every virtual address from the point of view of the OS, which can be either a kernel virtual address or a specific userspace virtual address. On x86 systems, the guest CR3 register holds a physical address pointer to the root of the OS page tables. Upon finding a virtual address, the introspection service traverses the page table to find the corresponding physical address. If a physical address is not present in the table, then the virtual address is either invalid, or the OS has paged out the memory. A modern day OS include separate address spaces between processes and thus the monitor must be able to translate according to the context.

The symbol table contains pointers to internal data structures of the running OS, such as the table of currently running processes and the list of loaded kernel modules. When traversing internal system data structures in a complex system, the introspector is given a subset of the context required to get to its destination. In ‘black box’ systems where there is little semantic information about the OS available, this indeed represents a challenge for monitoring as there exists a semantic gap.

It should be noted that traversing data structures in memory is a slow process. For every virtual address pointer, a full page-table lookup is required. This can be overcome by caching the address space before monitoring of the platform begins. However, this increase in performance does mean that some

parts of the cached table may be invalidated during the lifetime of the system as the load on paging continues.

Disk-activity monitoring Although memory contains what is currently loaded and executed, it is still a single data source with an incomplete picture of the system. We thus augment this by a subcomponent of the hypervisor for measuring access times to the primary hard disk. Furthermore, we explore if it is possible to distinguish raw disk activity to see if it possible to measure whether there are expected variances in behaviour that can be checked, with the remote possibility to identify some runtime behaviour properties by eliminating system noise. We define expected runtime as execution trajectories.

Some hypervisors include filesystem drivers that contain functionality to mount and interact with common file-systems. In our model all filesystem logic is kept to the driver that lies in the guest OS. However, since the hypervisor logs access to the disk resource, it is able to monitor control and data channels to the disk. In particular, features like the inter-arrival times, the size, and the address patterns of disk accesses can be monitored and analysed.

4.3.3 Example Applications

Trust characteristics of a system can be very subjective and will vary according to different organisational perspectives and platforms. The following working examples illustrate different scenarios:

Example 1: Process Launch In order to detect process launches, the process table is continuously polled by the integrity monitor from boot-time onwards. If an application is launched that has not previously been seen before, then this will be logged and the reflection of trust would be updated based on policy. A policy might state no programs other than pre-installed programs should be started, or that no two specific applications be run at the same time. There is no enforcement to stop the user from executing an unknown program, but the hypervisor is aware as and when it does occur. In the event that this happens, a log is made and sent to update the reflection of trust tied to that client.

Example 2: Integrity of Executable Code The hypervisor can produce hashes of the code segments of the operating system, drivers, and application programs, and compare these to hashes of known good copies. Thus, if a section of memory is tampered with or altered by a malicious user to bypass

security measures, this can be detected by the underlying hypervisor. If the user of the OS decides to unload a kernel module or uninstall a device driver, then the hypervisor would be aware of this event and update the level of trust.

Virtual platforms that run in userspace, such as parts of the Java Virtual Machine, could be checked for code integrity. An open research question this poses is how to extract additional context about a userspace program, such that the context can be captured and placed inside a monitor independent of the noise of the operating system, with methods such as tracing and memory analysis.

The same technique can also be used to detect an attack known as 'process hollowing', where a malicious program overwrites the virtual-memory contents of an existing legitimate process in order to evade detection by security software ([LAHR11], pp. 496–499).

Example 3: Unexpected disk activity If additional patterns are detected other than the usual boot trajectory, then this may indicate that a modification to the system has been made to execute additional services or applications on startup. This could lower the level of trust and more monitoring may be needed in the form of memory analysis, for example.

* * *

In this chapter we have identified technologies that can support trust domains by enabling monitoring and enforcement on both processing elements and on data. We have provided a survey of existing solutions and described our hypervisor for enabling secure monitoring of client devices.

This chapter is based on and uses material from the Trust Domains Deliverables D4.1b [KM12] and D4.2 [Sha12]. Note that the survey of existing technology given in Deliverable D4.1b has been shortened considerably for the presentation here. For the full survey we refer the reader to the original source.

Chapter Five

Modelling for Trust Domains

In practice, a trust domain comprises a complex network of interactions between organisations and individuals, which are governed and supported by social norms, expectations, and technical infrastructure. Modelling is used to explore this situation, helping to understand the implications of particular decisions, and thus ensuring that important properties hold. Modelling can support trust, as it can give guarantees on some properties, and it can support designers and users in making optimal decisions, both when the system is designed and when the system is in use. As illustrated in Table 5.1, we broadly distinguish modelling approaches based on the nature of their outcomes and on the point at which they are applied in the system lifecycle:

Modelling for verification: The goal of modelling for verification is to verify that something can or cannot happen, i.e. if the rules that govern the system's behaviour allow certain outcomes to occur. For instance, modelling for verification could prove that an attacker cannot gain access to some information. A system, or components of a system, for which such a proof has been obtained can then be trusted.

Modelling for evaluation: The goal of modelling for evaluation is to obtain insights into quantitative properties of the system under the assumption of stochastic behaviour. Quantitative properties of interest could be leakage rates of information or the time it takes to perform certain operations. Modelling for evaluation thus helps us to understand the impact of decisions on operational properties. This insight is helpful in optimising parameters when there are tradeoffs between e.g. performance and security.

	Verification	*Evaluation*
Design-time	PoliVer, StatVerif	Gnosis++ Automated Model-Generation Tool (AMG)
Run-time	—	Online Model-Generation Tool (OMG)

Table 5.1: Modelling tools for trust domains, structured by the aim of modelling (Verification or Evaluation) and the time of application (Design-time or Run-time).

We also distinguish according to the point in time at which modelling is applied:

Design-time modelling: Modelling at design-time is performed in order to optimise the design of the system before the system goes on-line. Modelling decisions to be made here include the type and number of technical components, policy settings, and operational parameters of components. Typically, design-time modelling is not time-critical, and thus approaches that have a high overhead in terms of constructing the model or in obtaining results from the model can be applied.

Run-time modelling: Modelling at run-time helps to support decisions during the operational phase of the system by assessing and illustrating the impact of user decisions on properties such as security or performance. Run-time modelling is particularly important in the complex settings typical for most trust domains, where decisions may have far-reaching unintended results. As run-time modelling is applied to support the user in operational decisions, it must be possible to return results fast.

In this chapter we discuss the various modelling methods that have been developed in the Trust Domains project. These modelling methods differ from the semantic model discussed in Chapter 3 in that their purpose is not communication about the trust domain, but rather the derivation of properties of the domain. We start by discussing verification methods and then discuss approaches for evaluation. We conclude the chapter by addressing the mathematical foundations underlying these modelling approaches.

5.1 Modelling for Verification

In modelling for verification the goal is to establish the possibility or impossibility of performing certain actions or achieving certain goals in a given setting. This allows us to verify if a security protocol fulfils its requirements, i.e. is robust against specific attacks, or if the policies defined for a system prevent users of that system from performing undesirable actions. Modelling for verification thus helps with the task of determining if a complex system of rules, such as a system policy or a secure communication protocol, accurately reflects the intentions of its designer.

In modelling for verification the system is described in terms of states and actions of agents that change the states. The modeller then queries the model if, starting from a given state or set of states, another specified state or set of states can be reached. Typically the tool will report not only whether the target state can be reached, but also return a sequence of actions that lead to the target state. This sequence helps to identify potential attack paths and points to disrupt them, in order to prevent the attack from succeeding.

In the Trust Domains project two modelling tools support specific needs of the project by verification methods: The PoliVer tool [KR11] provides policy verification, while the StatVerif tool [ARR11, Sta] enables verification of security protocols. We describe these tools in the following sections.

5.1.1 Policy Verification using PoliVer

Policies support trust domains by formalising permitted and prohibited actions of agents in a system and enabling the automated enforcement of these permissions. In this way, trust can be placed in a system, if it is known that the system does not permit actions that would break the trust. The design of policies for any system of realistic size is tremendously complex, as the designer has to ensure that the policies capture all cases and forbid all unintended actions. The PoliVer tool helps the designer in this task by automated checking if a system of policies fulfils its requirements. In the following we give a brief summary of the approach, intended to provide an understanding of the basic concepts. For full detail we refer the reader to [KR11].

With PoliVer, the designer first specifies the states of the system and the actions that can be performed by agents in that system to change the state. Then, the designer specifies the policies which restrict the set of actions that can be performed in each state.

The system *states* are specified by variables, and by predicates and logical formulae defined on the set of variables. Predicates defined on variables are referred to as atomic formulae, and logical combinations of atomic formulae (which may include existential and universal quantifiers) are referred to as logical formulae. Formulae may evaluate to `true` or `false`. Agents are represented by variables.

The system state can be modified by *action rules*. An action rule α, specified by

$$\alpha(\vec{v}) : E \leftarrow L, \text{ where } \vec{v} = (v_1, \ldots, v_n),$$

describes an action that can be performed by the agent v_1 if the logical formula L evaluates to `true`. The effect of the action is given by E; in particular, E may contain the effects $+w(\vec{v})$ and $-w(\vec{v})$, which specify that the value of $w(\vec{v})$ will be set to `true` or `false`, respectively.

Furthermore, *read-permission rules* can be given. These specify under which conditions an agent can read the value of an atomic formula. Read-permission rules have the form

$$\rho(u, \vec{v}) : w(\vec{v}) \leftarrow L,$$

where u refers to the agent that performs the read, $w(\vec{v})$ is the atomic formula to be read, and L is a logical formula that describes the condition under which $w(\vec{v})$ is allowed to be read by agent u.

Access control policies are then described by sets of action rules and read-permission rules on given predicates.

After describing the system and the access-control policies, the designer formulates a query to the PoliVer system. The query takes the form

$$\{W_s\} \rightarrow G,$$

where $\{W_s\}$ is an initial state and G is a target state. Intuitively, the semantics of the query are: 'Starting from the given initial state, does the access-control policy permit a sequence of actions through which the target state can be reached?' In general, the target state is an undesirable state, such as an agent being able to access data that should not be accessible to them. Given this query, the PoliVer tool applies a search backwards from the target state, establishing sets of predecessor states until either no further states can be discovered or the initial states are found. If the latter is the case, then the policy allows a sequence of actions that leads to the undesirable state; i.e. the policy is not effective. The

tool also returns the sequence of actions, which provides an example of the path an attacker would take. This example can guide the designer in deciding how to amend the policy in order to make it more effective.

5.1.2 Verification of Stateful Protocols using StatVerif

StatVerif [ARR11, Sta] is a tool for the evaluation of protocols in systems with global state. StatVerif extends the ProVerif tool [BAF08, Pro] such that it supports global state.

In the basic ProVerif language, the protocol designer describes the protocol using *processes* that exchange messages. Messages are modelled by *terms*, i.e. names and variables. Processes can create values and send and wait for messages. Furthermore, processes can run in parallel, can depend on conditionals (defined on messages) and can be replicated such that multiple copies of a process can be running at the same time.

ProVerif supports primitives for security protocol evaluation. In particular, cryptographic operations are modelled by reductions. For instance,

$$\mathrm{senc}(x,y) \to c$$

denotes that the encryption of the plaintext x using the key y results in the ciphertext c, while

$$\mathrm{sdec}(c,y) \to x$$

describes the decryption of the ciphertext c using key y. Encryption is an example of a *constructor*, whereas decryption is a *destructor*.

StatVerif extends this basic formalism by the notion of a *cell*, which represents global state. A cell s with initial value M is denoted by

$$[s \mapsto M].$$

Cell values can be read, evaluated in conditionals, and assigned to. Furthermore, access to cells can be locked such that the process that has locked access has exclusive access to the global state cells until the cells are unlocked.

The model (described in terms of processes) is then translated to Horn clauses, i.e. clauses of the form

$$H_1, H_2, \ldots, H_n \to C,$$

where $H_1, \ldots, H_n$ are *hypotheses* and C is a *conclusion*. These clauses describe what can happen (the conclusion), given the hypotheses. In StatVerif, the user specifies hypotheses and conclusions with the predicates $\texttt{attacker}(\tilde{M}, N)$ and $\texttt{message}(\tilde{M}, N, K)$. The predicate $\texttt{attacker}(\tilde{M}, N)$ means that the process can reach a state where the state variables $\tilde{s}$ have the values $\tilde{M}$ and the attacker knows the value N. Similarly, $\texttt{message}(\tilde{M}, N, K)$ implies that there exists a reachable state where the variables $\tilde{s}$ have the values $\tilde{M}$ and the value K is available on channel N.

In order to verify system properties, the user queries the StatVerif tool for the existence of a certain property. The tool will then create the appropriate clauses from the process and check if they admit this property. For instance, the user may ask whether there is a way for an attacker to gain access to a certain piece of data, if a specific security protocol is followed, and StatVerif will respond with `true` or `false`, depending on whether this state exists.

5.2 Modelling for Evaluation

In modelling for evaluation we assume stochastic behaviour of the entities involved in a system. This allows us to capture situations where entities can make choices; in particular, it enables us to model the case where human agents may choose between different ways of performing an action, each of which may have different impact on security, and thus on the trust that can be placed in that agent. Furthermore, based on the assumption of stochastic behaviour, we can represent mistakes made by human agents, and failures of technical components. We then obtain values for quantitative properties of the system, such as the time until a security incident occurs, a measure of the trustworthiness level, or the time that an action will require when security measures are applied. These values help to make informed decisions with respect to parameters that affect tradeoffs in system design and system operation.

In modelling for evaluation we can use different levels of detail. Models with a high degree of abstraction represent the system and its properties using mathematical equations and stochastics. They produce general results and can often be evaluated with little computational effort. On the other hand, such models omit many details, capturing them in stochastic behaviour instead, and may therefore misrepresent crucial aspects of the system. Furthermore, they may require considerable effort in abstraction and translation on the part of the user in order to choose appropriate parameters and in order to apply the insights gained from the model in practice. Models with a low degree of abstraction

represent the system and its properties using abstractions of the processes and resources in the system. With these concepts, a much higher level of detail is achieved, as the behaviour of entities can be modelled using a notation that is similar to a programming language. This allows the modeller to capture important properties, and also makes it easier to parameterise the model, as parameters can be reflected directly, without the need for abstraction. On the other hand, the results from the models are specific to the modelled situation, and evaluation of these models typically requires simulation, which is computationally expensive.

High-level and low-level modelling complement each other, and the application of approaches from both abstraction levels can help in different aspects of the evaluation of trust domains. In the Trust Domains project we have developed three modelling approaches to support decision-makers both at design-time and at run-time: Modelling with Gnosis++ enables low-level modelling using an extension of the Gnosis [Cor] modelling formalism to explicitly support information flows. The Automated Model-Generation Tool (AMG Tool) and the Online Model-Generation Tool (OMG Tool) are based on stochastic modelling of information flows.

5.2.1 Gnosis++

Gnosis++ is an extension of the Gnosis modelling tool [Cor]. Gnosis++ is built as a library on top of the Ruby scripting language. Gnosis++ models are implemented in Ruby, which simplifies the modelling process. Gnosis++ enables low-level, high-detail modelling of system behaviour and thus helps the system designer in assessing the impact of a wide range of changes on the security and trustworthiness of a trust domain. The underlying formalism is based on the following primitives:

Processes describe arbitrary behaviour and are used to model the behaviour of human agents and system components. Processes operate on resources and information, and can modify these. Processing times are modelled as delays, whose lengths can be described by probability distributions.

Processes are represented as objects with their own local state. Each instance of a process is thus independent of other instances of the process. Furthermore, process behaviour can be inherited, which allows the modelling of similar, but not identical agents.

Resources model physical resources such as hardware tokens, devices, or the position of an agent. Resources can be moved between locations (if enabled by the location graph), but can neither be created nor deleted.

Information is similar to resources in that it can be moved between locations. However, information can also be created, deleted, and copied. Therefore, the information primitive can be used to model information flows.

Locations and Links describe where resources and information can reside, and the possible flows of both between locations. Locations and links define the location graph, which describes the permitted flows, for instance in modelling containment and isolation mechanisms.

Given these primitives, a system is modelled by describing the actions of agents in the system using processes operating on resources and information, whose flow is constrained by the location graph. Times required for processing etc. are modelled by delays in the process. In general, the model is built only to a certain level of detail. Behaviour that is not captured by this detail is abstracted away from using stochastic models, such as in delays or in random choices between different actions.

The system is then simulated for a specified length of time. Throughout this time the number of copies of information items and resources in specific locations as well as the number of specific events is counted, and from these statistics measures on the spread of data throughout and beyond the boundaries of the system can be computed.

Example of a Gnosis++ model Gnosis++ has been designed to make it easy to model information flow scenarios and run simulations to compare different architectural and behavioural constraints, albeit at an abstract level. Here we give a simple worked example based on part of the Mergers and Acquisitions (M&A) scenario described in Chapter 1. We develop the scenario through four simple examples starting with a team exchanging data, then adding in an attack process, then adding in data flow controls such as enterprise DRM and finally adding in a more directed attack process aimed at defeating these controls. Along the way we also discuss how the model can be extended to deal with a more detailed M&A workflow process.

We start by saying there is a team of people who work on an M&A deal, and each will have a computer system that they use and on which they store and work on data. We start as the deal is initiated and model the phase of data acquisition. Thus we have a number of people within the M&A team, each of

which is receiving or creating data of a given class relating to the M&A deal and storing it on their device. We also have an outside world that the team will sometimes communicate with. Here we do not differentiate between different people on the outside as we are only interested in how much deal data may reach this 'outside' group.

Within our initial model we start by creating a type of deal data and an outside location. To do this we have a Ruby class called `InfoShare` which represents a type of data that is named. We also have a `Location` class which represents a location that can be connected with other locations via a graph and at which information and resources can reside:

```
$dealData = InfoShare.new("Deal Data")
$outside = Location.new("outside")
```

We then define a person working in the team, how they fit into the overall system, and their behaviours. Here we are interested in general behaviours associated with the M&A team and how these behaviours influence the overall flow of information. We define a member of the M&A team as a class, allowing us to create multiple similar team members as object instances.

Each individual will have their own location to represent the repository where they store information. For example, the location could represent their laptop. We could enrich the example by adding multiple locations that represent different individual devices and storage repositories such as laptops, personal corporate storage, and smartphones, but here we keep it simple with each person having a single location. Locations also form part of a graph structure where information is allowed to flow across the arcs of the graph. This structure can be used to model networks, OS containment or other architectural constraints. As well as having a Location structure we can place locations into groups or `LocationSets`, which makes it easier to specify the connections. We define a `LocationSet` for the team and place all members of the team within the set, in order to allow them to communicate. We also link the `outside` location into this location set so that there are no architectural constraints to information flow in this first version of the example.

We now model the way each individual interacts with the information. The M&A problem includes multiple phases starting with an acquisition phase where information is collected, followed by an analysis phase, and then the production of a deal book. To keep this example simple here we just model the acquisition phase. Hence we have each team member collecting information from an external source and then sharing it with some of the other team members. We can write this as two processes. The first process is the acquisition

process where information is gathered, creating information at that person's location. In absence of any more information about the acquisition protocols we have a process where new information is created at a random interval drawn from an exponential distribution. The second process allows for the sharing of information between the team members. As with the acquisition process we assume that information is shared at intervals described by an exponential distribution. In order to share information, a second team member is chosen as the recipient at random, and then information is copied from the source to the destination, such that one copy of the information ends up in both locations.

We now add a third process where information is shared erroneously with those outside of the group, as can happen, for instnace, when the wrong email address is included in an information transfer. In each case, we could produce more complex versions of these processes that model the way people work to a higher degree of accuracy, but this would require more information about how the process operated. Lastly, we allow each team member to be customised with a different acquisition rate, data sharing rate, and error rate. This would allow us to model individuals or different classes of people. Thus we add these rates as parameters to the creation of the team members. We obtain the following model with a `Team` class that sets up the team:

```
class Team

  # Create a team with number members with default
  # information creation, sharing, and error sharing
  # rates
  def initialize(number, createRate, shareRate,
      oShareRate)
      @teamLocs = LocationSet.new("Team")
      @teamLocs.link($outside)
      @teamSize = number
      @team = []
      for i in 0..(number -1)
        @team[i] = Person.new(i, self, createRate,
            shareRate,
                oShareRate)
        # Connect the location graphs
        @teamLocs.addChild(@team[i].location)
        @team[i].location.link(@teamLocs)
      end
  end

  #start simulation for each member of the team
  def start()
```

```
  # Select a member of the team but not the requesting
  # person
  def selectMember(person)
    ...
end
```

We also have a `Person` class that acts as each member in the team and contains the processes that they use to interact with data:

```
class Person
  # Distributions for people processes
  @@timeToCreate=Negexp.new
  @@timeToShare=Negexp.new
  @@timeToMisshare= Negexp.new
  @@infoCreated=0

  # create a person and related location
  def initialize(pId, team, createRate, shareRate,
      oShareRate)
      @location = Location.new("laptop #{pId}")
  end

  # Launch all processes for this person
  def start()
    launch {createInfo()}
    launch {distributeInfo()}
    launch {distributeOutside()}
  end

  # Acquisition process for creating information.
  # Wait for a time to get/create a document, add it
  # to this location, then wait again.
  # Record how much information is created.
  def createInfo()
    while true
      hold(@@timeToCreate.nextM(@createRate))
      $dealData.createInfo(1, @location)
      @@infoCreated=@@infoCreated+1
    end
  end

  # Share information. Wait for a random time then
  # copy to a new location
  def distributeInfo()
    while true
      hold(@@timeToShare.nextM(@shareRate))
```

```
        $dealData.copyInfo(1, @location,
            @team.selectMember(self)location())
      end
    end

    # Share information. Wait for a random time then
    # copy to a new location
    def distributeOutside()
      while true
        hold(@@timeToMisshare.nextM(@oShareRate))
        $dealData.copyInfo(1, @location, $outside)
      end
    end
  end
```

These classes are standard Ruby code but we introduce additional constructs that come into play when we execute the code as co-routines within a simulation. Of significance here is the `launch` statement in the start function. This launches a new Gnosis++ process that runs concurrently with the other processes. The use of co-routines means that unlike threads these processes have a well-defined execution order, and that only one process ever runs at a time. This means we do not need to worry about synchronisation or how shared variables are managed. The simulation run-time will run one process until it comes across a statement such as a `hold` or an access to a Gnosis resource or an information resource. Control then passes to the simulation engine which decides on the next process to be run based on the available resources and a time queue. The `hold` statement is used in the processes above, and this has the effect of stopping the process for a certain amount of time. So, for example, the process for creating information will wait for the time till the next piece of information needs to be created and then be scheduled at this time. Gnosis++ simulations are discrete-event simulations so that there is a time queue with the times when events happen. When the simulation has finished everything at the current time it then looks for the next event in time and moves the clock to that time.

The second set of commands worth noting here are the creation of locations and links between them. Information can be associated with each location and then be copied between the locations (see the `distributeInfo` process). Gnosis++ is intended to describe a process-based system and then allow simulations of that system to be run based on the environment being represented by stochastic variables. In this case, the environment is simply the team size, and the rates of information creation, distribution and erroneous distribution. As

we run a simulation we can run through as if the team exists and then record how much information is created, copied to different locations and lost to the outside world. The Gnosis++ model is thus a stochastic model based on a number of random variables that represent the variability of the environment. As such we use the simulation system to run a large number of experiments so that we can understand how variable the outputs are to the randomness within the environment. Any model is an abstraction of a problem and as such it will not necessarily make accurate predictions, but a model will often show the shape of the solution space. As such we may want to use the model to explore the effects of changing the environment. So we could for example rerun our model assuming staff are more careful and less likely to send documents to the wrong people. Alternatively we can start to add in additional threat actors and architectural constraints.

Here we start be exploring the idea of adding in an additional threat of a computer system getting infected with a Trojan allowing information to be removed. Here we view information that can be removed as exposed rather than trying to assess the motivations and skills of those who may get the data. We could explicitly model an attacker, but instead we add an infection process to our `Person` class. Here we introduce a new external location of a hacker with the idea of seeing how much information can be transferred to them. We model the infection process by having a time to infection and a time to clean up along with a period where the computer is infected. This data could be estimated by looking at helpdesk calls, the antivirus system and other security monitoring facilities. Most computers will never get infected but all run this process with the mean time to infection being much larger than the length of the simulation. Within our example we keep the infection code very simple and say that there is a time till an infection at which point data becomes exposed. When it becomes exposed we create a link between the bad location and the location within the infected instance of the `Person` class allowing information to be removed. We then remove information to the bad location so that it can be accounted for:

```
$bad = Location.new("bad")

# Share information. Wait for a random time then
# copy to a new location.
def infection()
  while true
    hold(@@timeToInfect.nextM(@meanTimeToInfect))
    @location.link($bad)
```

```
      @infected = true
      dat = $dealData.dataItems(@location)
      if (dat != nil)
         $dealData.copyInfo(dat, @location, $bad)
         hold(@@timeToCleanup.nextM(
             @meanTimeToCleanup))
         @location.unlink($bad)
         @infected = true
    end
  end
```

Now that we have two simplified loss processes within the model we can start to look at different countermeasures. One such measure would be to use an enterprise DRM system to encrypt the deal data so that its access is linked to the authentication of the user. We can do this by creating an additional class of unencrypted deal data which is the form data is in most of the time. We also use this idea to model architectural constraints provided by the encryption into the system. Here we limit what data can flow over the location graphs. In actual fact the encrypted data could flow, but without keys it cannot be recovered, and hence the underlying information cannot be extracted. Note that links between the team allow the flow of the deal data since access is managed by the DRM system, hence as a member of the team is linked into the team graph all information can flow. However, the outside location is linked in in such a way that only unencrypted data flows. Here when we form links we specify what information can flow and hence change lines in the set-up of the team:

```
$unencDealData = InfoShare.new("unencDealData")

class Team

  # Create a team with number members with default
  # information creation, sharing and error sharing
  # rates
  def initialize(number, createRate, shareRate,
      oShareRate)
      @teamLocs = LocationSet.new("Team")
      @teamLocs.link($outside, $unencDealData)
      for i in 0..(number -1)
             @team[i].location.link(@teamLocs)
      end
  end
```

Similar changes are made in the `Person` class where a link is used so that only unencrypted data can flow. Note that the link statement between the locations

has an additional parameter containing the data that is allowed to flow. So far in this example the unencrypted data does not exist and so we can extend our usage model to include the person accessing data and making the unencrypted form available. As we modify our usage model with this additional process we can take account of infected machines. Hence we have a new `useData` process that allows data to be extracted. The old infection process and erroneous email processes will no longer extract data, modelling the architectural constraints brought in by using enterprise DRM. This process then interacts with the infection process allowing unencrypted data to be extracted (exposed) as it is created. However the Trojan running on the infected machine here is now assumed to be more complex in that it has the ability to extract data whilst it is being used. Hence here we may wish to separate out the different forms of infections:

```
# Share information. Wait for a random time then
# copy to a new location
def useData()
  while true
    hold(@@timeTillUse.nextM(@meanTimeTillUse))
      # create a decrypted copy
      if ($dealData.dataItems(@location)>1)
              @unencDealData.create(1, @location)
              if (@infected )
                      $encDealData.copyInfo(1,
                          @location, $bad)
              end
              hold(@@useTime.nextM(@meanUseTime)
              @unencDealData.create(1, @location)
       end
  end
end
```

We can look at the results in terms of the histograms representing the proportions of data exposed in different versions of the model. For example the histogram shown in Figure 5.1 comes from the first version of the model where data is only exposed through being sent to the wrong person. This is done assuming a team size of 10 and set rates for information creation, distribution, and an erroneous email rate of on average 2 bad emails a year. Hence we have a histogram with very small proportion of losses in terms of the loss rate per document. However the zero loss bar is relatively small at 104 out of 1000 simulations with no loss.

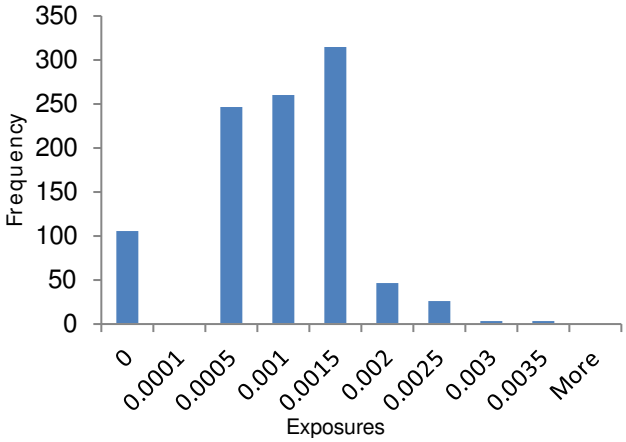


Figure 5.1: Histogram showing the distribution of loss events over 1000 simulation runs.

Here we have shown how a very basic Gnosis++ model can be developed to model a situation. The Gnosis++ constructs in the Ruby language allow processes to be expressed for how information is moved and how threats to the leakage of information also happens. The extensions allow models to be run as a discrete event simulation allowing different architectural choices to be compared. Within the language itself we have added support for information to be moved between locations with a graph structure being used to specify possible paths for all or particular types of information. We also introduce the notion of information resources into the language. Unlike physical resources (which are only movable), information resources can be created, copied and deleted. The incorporation of Gnosis concepts into the Ruby language allows us to have an object orientated modelling language making it easy to model multiple entities whilst retaining the mathematically sound properties of gnosis. The example given is relatively simple but provides a framework that can be expanded to include specific processes concerning how information is handled.

5.2.2 Automated Model-Generation Tool (AMG Tool)

The Automated Model-Generation Tool (AMG Tool) helps system designers in evaluating the impact of policy settings and performance characteristics on

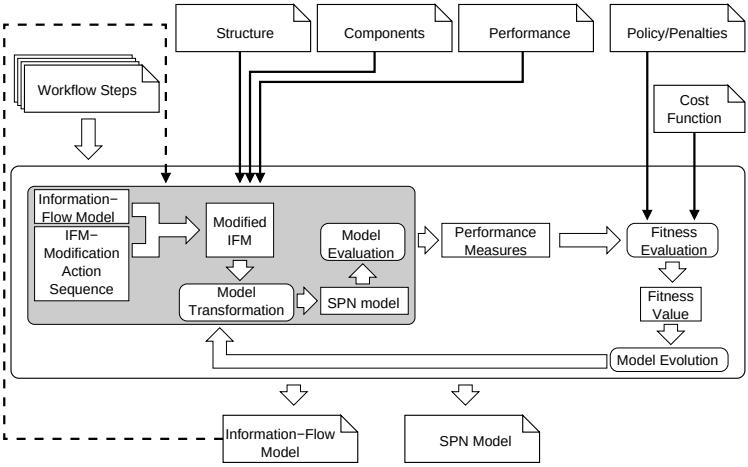


Figure 5.2: Operation of the AMG Tool.

the behaviour of human agents affected by the policies, and on the security on the system. The tool uses an information-flow abstraction, that is, it models the system in terms of information flows between source and destination nodes in a graph. The behaviour of human agents is modelled as establishing or removing connections, which then enable or disable flows. Performance characteristics are evaluated as the time it takes for information to flow from the source node to the destination node. The security of the system is assessed by the time it takes for the information to flow to an attacker node.

Figure 5.2 illustrates the operation of the tool: The user of the tool describes a workflow by splitting it into information-exchange steps. In each of these steps, data needs to be transmitted from a source to a destination. The data transfer can be achieved using different technical means (e.g., the data might be shared using a Cloud service or printed and posted using a courier service). The source, the destination, and the means of transport are modelled as nodes in a directed graph. In order to form a graph that connects the source and the destination, the nodes must be connected by links. For each of the links the modeller can specify the speed and a penalty associated with using that link. The penalty models the policy, i.e., modes of transport that are discouraged can be associated with a high penalty. Initially, only the properties of these links are given, but no links are established. The tool then applies a

genetic algorithm to find a graph that connects the nodes, such that the graph is optimal with respect to a user-specified cost function. The cost function can capture aspects such as risk-affinity, by trading off penalties against performance. Furthermore, the cost function can also be used to model the impact of a timeout. The graph produced by the tool serves two purposes. First, it illustrates the flow of information between the source and the destination. Second, it can be used as an input to the modelling of attacker behaviour. In this step, the genetic algorithm modifies the graph in order to direct information to the attacker's node. This model then illustrates attack points.

In the background, the genetic algorithm operates on sequences of actions that add or remove connections in the graph; that is, the genomes of the algorithm are sequences of such actions. A sequence's fitness is evaluated based on the model that results from applying the sequence to the graph model, as follows: Penalty costs are accumulated from the actions in the sequence. Performance costs are computed by transforming the graph model to a Stochastic Petri-Net (SPN), from which the distribution of times for moving data from the source to the destination can be derived as a phase-type distribution.

The output of this approach is thus two-fold. On the one hand, it produces results that are immediately useful for assessing the impact of choices at design-time. On the other hand, the tool automatically generates Stochastic Petri-Net (SPN) models of the system (optionally including attacker behaviour) from high-level specifications of performance characteristics and penalties. These models can be used in further evaluation of various properties. The advantage of automatic generation of models over manual modelling is that the former may include details that could be overlooked by a human modeller. For instance, a combination of policy settings and performance characteristics might favour the use of an insecure way of transport, but this might not be obvious from the specifications, and thus would be omitted in a manually created model.

It should be noted that, although less demanding than low-level simulation, this approach is computationally expensive and thus is typically applied at design-time, rather than at run-time.

In the following, we describe the AMG Tool in detail. We start by describing the models employed in the tool and the transformations between them. We then describe how a realistic model is generated in an automated way by an optimisation process built upon these models and model transformations.

5.2.2.1 Models and Model-Transformations

The AMG tool employs four different types of models: The *Information-Flow Model* (IFM) represents information-flows between entities or devices in a system, the *Action Model* (AM) models the actions of users or attackers, the *Workflow Model* (WFM) models the workflow, and *Stochastic Petri-Nets* (SPN) are used as the underlying modelling formalism for evaluation of the models. In the following we first describe these models and then discuss the transformations between the information-flow model and stochastic Petri-net model that form the basis of the AMG and OMG tools.

The Information-Flow Model (IFM). An information-flow model is used to describe the information flow between entities or devices. The abstraction proceeds as follows: We assume that data is transferred between a source and a destination, using intermediate steps. The source and destination are typically devices such as computers. The intermediate steps are modes of transport, such as postal mail, e-mail, or Cloud services. As such, sources, destinations, and modes of transport comprise locations in which information may reside, either transiently or permanently. Transient locations are, for instance, e-mail servers that store the message only until it has been transmitted, while a storage service in the Cloud can be considered permanent. In the information-flow model we model these locations as nodes. Information-flow is then modelled by directed edges between the nodes. Edges have three types of attributes:

Speed of data-flow The speed attribute describes the speed at which data can flow along this edge. This attribute is used to model the fact that different modes of transport have different impact on performance. For instance, sending information by post is certainly slower than sending it by e-mail. The speed attribute also captures performance aspects such as setup costs involved with using a specific mode of transport, e.g. the impact of setting up an encrypted connection. At present, the speed attribute is a single numeric value that specifies the rate λ of an exponential distribution. The extension to phase-type distributions [Neu81] is straightforward and would enable the modelling of arbitrary distributions, as any distribution can be approximated by a phase-type distribution [TBT06].

Penalty The penalty attributes give the penalties that a user has to pay when they use this edge as a connection to transmit information, or when they

remove this edge. This attribute is used to model policies: Adding connections to/from nodes whose use is undesirable from an organisation's point of view will be assigned a high penalty, in order to discourage users from using such connections. Equivalently, removing connections to/from nodes that the organisation considers desirable will have a high penalty. Note that the penalty attributes only affect legitimate users. These attributes are two numeric values.

Attack costs The attack-costs attributes specifiy the costs that an attacker has to bear when establishing or removing an edge. This attribute models the security of devices, i.e. connections to/from more secure locations are assigned higher attack costs than those from insecure locations. Typically, these attributes will be based on assumptions and general knowledge about the relative security of different modes of transport. The attack-cost attributes are two numeric values.

The Action Model (AM). Building on the abstraction provided by the information-flow model, the action model describes the behaviour of users (or attackers) of a system, as follows: The goal of users is to transfer information between a source and a destination (the goal of an attacker is analogous, as the attacker wants to transfer data to their own locations). In order to achieve this, some means of transport need to be used. The action model describes how users or attackers utilise the means of transport that are available to achieve their goals. An action model operates on an information-flow model by adding or removing connections between nodes in the model.

An action model consists of a sequence of `Add` (i, j) and `Remove` (i, j) actions that are executed consecutively on an IFM. `Add` (i, j) adds a connection between nodes i and j in the IFM, whereas `Remove` (i, j) removes a connection. Each action incurs some costs, as specified in the IFM. If the action model describes the actions of a user, the action costs are given by the penalties in the IFM; conversely, if the action model describes the actions of an attacker, the costs are the attack costs from the IFM. The sum of these costs comprises the total cost of the action model.

The Workflow Model (WFM). The workflow model combines a sequence of information-flow models to model a workflow. In this model, each IFM model describes one information exchange. In a workflow model, the individual IFM models may influence each other, in order to model the fact that steps

in a workflow are not independent. In particular, the time taken in one step of a workflow may influence later steps, if that workflow is executed under time constraints. A workflow model consists of the following parts:

Information-Flow Models A WFM includes a sequence of information-flow models. Each information-flow model models one information exchange necessary in order to complete the workflow. It is assumed that these steps have to be executed in sequence.

The cost function The cost function is used in the model-generation phase to optimise each IFM (see Section 5.2.2.1).

Workflow parameters Arbitrary workflow parameters are used in the cost function to determine the quality of an IFM. In particular, these parameters usually include a timeout for the workflow, after which the results of the workflow would be considered useless. This aspect can be used to model time-constraints, which are often present and and affect decision-makers.

Stochastic Petri-Nets (SPNs) Stochastic Petri-Nets (SPNs) are a widely-used modelling formalism that has both a solid mathematical foundation and a graphical representation. The following introduction to SPNs is limited to the aspects required for understanding their use in the AMG and OMG tools; further information may be found in e.g. [Hav98].

SPNs consist of *places*, *transitions*, *arcs*, and *tokens*:

Places Places in an SPN are locations that store *tokens*. In a graphical representation, a place is typically represented a a circle with the number of tokens on this place written in the circle.

Arcs are directed edges connecting a place to a transition or a place to a transition. Each place may be connected by multiple arcs to multiple transitions, and each transition may be connected to multiple places. We distinguish between *normal* and *inhibitor* arcs; their semantics are explained in the following. Normal arcs are graphically represented as arcs with an arrow denoting their direction. Inhibitor arcs carry an empty circle in the place of the arrow.

Transitions are connected to places by arcs. Each transition has a number of incoming arcs and a number of outgoing arcs. Each incoming arc connects a place to the transition; each outgoing arc connects the transition to a place. We refer to the places connected by incoming arcs as

input places, while the places connected to by outgoing arcs are *output places*. Transitions carry as a parameter a single positive numerical value that specifies the rate of an exponential distribution. Transitions are typically represented as empty rectangles, with the rate annotated next to the rectangle.

The state of an SPN, typically referred to as the *marking*, is the number of tokens on each of the places of the SPN. Transitions provide the means to tranform the model from one state to another: A transition is considered *enabled* if all of the input places that are connected to the transition by a normal arc have at least one token on them and none of the input places connected by an inhibitor arc has a token on it. When a transition becomes enabled, a random delay is drawn from an exponential distribution with the rate of that transition. If the transition is still enabled after that delay has elapsed, then the transition removes one token from each of its input places and puts one token on each of its output places, thus producing a new marking.

Given an SPN, various interesting measures can be computed on it. In our context, transitive measures such as the probability of there being a specific number of tokens on a particular place at a given time t and the time until a specific number of tokens are on specific places are of particular interest, as they capture the flow of information. The computation of these measures is based on the transformation of the SPN to a Continuous-Time Markov Chain (CTMC). In this CTMC, each marking of the SPN is a state of the CTMC, and the transitions are given by the rates of the transitions that transform the SPN between markings. Further details on the state-space generation process for SPNs may be obtained from e.g. [Hav98]. The SPNs generated as part of the AMG tool are absorbing, because they model the flow of information from a source node to a destination node. Consequently, the CTMCs generated from them are of finite size and absorbing, and thus the time until a particular state of the system is reached is described by a phase-type distribution.

In order to compute metrics on an information-flow model, we transform it to a stochastic Petri-net, on which we can then compute the metrics. This transformation represents locations in the IFM as places. The representation of edges depends on the types of the nodes they connect. Any edge connecting simple transient nodes or a simple transient node to a permanent node is transformed into a simple transition with the rate of the edge. An edge connecting a permanent node to any type of node is modelled by a transition and an additional node *B*. Upon firing, the transition transfers the token representing

the data from the input place to the output place. Additionally, the transition also restores the token to the input place, in order to represent the fact that the associated node stores the data forever. Furthermore, the transition also puts a token on the place *B*. This place, when non-empty, prevents the transition from firing again. This construct ensures that the associated CTMC is finite; without it, the transition would keep on firing and producing new tokens, i.e. new markings and therefore an infinite number of states. This would result in difficulties in computing measures on the CTMC.

5.2.2.2 Automated Model-Generation

In the automated model-generation tool a stochastic model is generated that allows the evaluation of a number of properties. The resulting model illustrates likely information flows and gives measures for the performance and security of the modelled system. Furthermore, it enables the evaluation of other measures that might be of interest. Automated model-generation proceeds from a workflow model (consisting of several information-flow models) and uses a genetic algorithm to optimise the information flow in each information-flow model, subject to the constraints imposed by the cost function and of the workflow model, most notably the timeout for the workflow.

Genetic Algorithm. Genetic algorithms are a widely-used optimisation method that employs biological metaphors (an introduction to genetic algorithms can be found in [Mit98]). The optimisation goals are expressed by a fitness function that computes the fitness of a choice of parameters. The parameters are considered the genome, and can be modified by crossing-over, mutation, extension, and truncation. Genetic algorithms operate on a population of parameter settings (called the population). In each step, a genetic algorithm creates a new population based on the previous one by modifying individuals from the old population. Fitter individuals are more likely to survive or contribute to the new population, and thus the overall fitness increases. It should be noted that genetic algorithms need not necessarily converge to an optimum; often, a good solution can be obtained by just performing a large number of steps.

Our genetic algorithm operates on a population of action models and uses the cost function of the workflow model in the fitness evaluation. The modifications are straightforward: Two action models are crossed over by choosing a random cross-over point and swapping the action sequences behind that point. Extension and truncation add randomly-generated actions or remove

randomly-chosen actions from the AM. Mutation chooses a random action in the AM and modifies it, e.g. by changing `Add` to `Remove`, or by changing one of the nodes that the action refers to.

We apply the algorithm as follows: First, a set of random action models is generated. We then run the algorithm for a few hundred rounds, evaluating the fitness of each individual in each step and choosing fitter individuals with higher probability. In order to ensure that a good individual is not lost, the new population is seeded with the fittest individual from the current population.

Fitness Evaluation. In order to evaluate the fitness of an action model, we apply the action model to the information-flow model, collecting the costs C incurred by the actions. We then transform the IFM to a stochastic Petri-net. In the IFM, the data is initially located at the source location. This is transformed to the initial marking of the SPN. The target marking represents the state when the data is at the destination. We then compute $E[T]$, the mean time until the first time the target marking is reached, when starting from the initial marking. This time corresponds to the time required for transferring the data from the source to the destination. If the data cannot flow from the source to the destination, then $E[T] = \infty$ will be reported. The overall fitness is computed from the costs and the mean time for transfer, as specified by the cost function in the workflow model.

The genetic algorithm is applied in each step of the workflow model, resulting in a sequence of action models. Each of these action models describes the likely information flows in the respective workflow step. Furthermore, each IFM then specifies a stochastic model for the respective step.

In general, these steps are the same for modelling the user and the attacker. However, for the user the penalty attributes will be used, whilst for the attacker the attack-costs attributes are used. Furthermore, for the user we start from an empty information-flow model. This models the situation that the user wants to establish connections to perform the work. When modelling for the attacker, we start from a model already created for modelling a user. This models the situation that the attacker attacks an existing interaction and tries to exploit weaknesses that might have occured just as a result of user decisions, e.g. by using an insecure mode of data transfer at some point in the process.

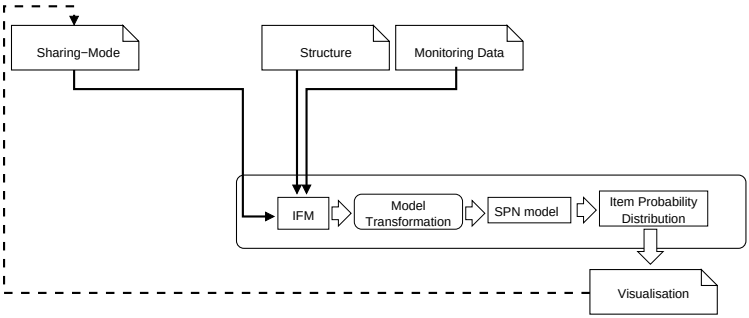


Figure 5.3: Operation of the OMG Tool.

5.2.3 Online Model-Generation Tool (OMG Tool)

The Online Model-Generation Tool (OMG Tool) applies principles of stochastic modelling to help users make informed decisions at run-time. In this approach, we consider the following situation: A user wants to share data with one set of users, but does not want another set of users to gain access to the data. Recipients of the data may vary in their trustworthiness with respect to not sharing the data, and the users' devices may also be more or less likely to leak the data. The data may be shared in various ways with different security properties, e.g. on some sharing methods it may be possible to prevent leakage of the information from the recipient's device. We assume that there is some monitoring data available on the trustworthiness of users and their devices, and we want to support the sharing user in making an optimal choice of recipients and sharing mode, such that it is unlikely that unintended recipients receive the data.

The operation of the tool is illustrated in Figure 5.3: Potential recipients of the data are abstracted as nodes in a directed graph. The edges of the graph model information flows. Each edge has a parameter that specifies the rate at which data flows along this edge. The rates for data-flow from the sharing user to the intended recipients are set to reflect the speed at which data can be shared. The outgoing information-flow rates on recipient nodes are parameterised based on a combination of trustworthiness monitoring data and the security of the selected sharing-method. The graph model is transformed to a Stochastic Petri-Net, from which the probabilities of each node having seen the data before a given time t are computed. These probabilities can then be dis-

played graphically, e.g. as a terrain map or as a colour-coded visualisation of the graph. Based on this information, the user can revise and adjust decisions accordingly, in order to reduce the probability of unintended parties receiving the data.

The underlying equations for this approach can be solved quickly, and therefore the method is used at run-time.

5.2.3.1 Modelling for Online Evaluation

We employ the information-flow model abstraction described in Section 5.2.2.1 to represent information flows for online evaluation. Potential recipients of the data are abstracted as locations. Edges model connections between recipients (e.g. colleagues may be connected). In the OMG tool, only the speed attribute is used, as no modification of the model structure takes place. The speed attribute here represents the leakage rate of data along this edge.

In the OMG tool we assume that data, once it has been at a node, will be stored there forever. This reflects the worst-case assumption that someone who has had access to some data will not forget or delete it. For this reason, only permanent nodes are used in the IFM model.

5.2.3.2 Solution

In order to obtain the probabilities of data being in a certain location at a time t we transform the model to the SPN, as described in Section 5.2.2.1, and then derive the continuous-time Markov chain. From the CTMC we can compute the probability of each possible marking at any time t. From the construction of our model out of permanent nodes it follows that this value directly gives the probabilities of each node having seen the data up to that point.

5.3 A Note on Mathematical Foundations for Modelling

Our modelling approaches use various notions of processes, resources and locations to model agents and their actions within a system. Furthermore, they typically employ the concept of a cost function, or, equivalently, utility, to describe preferences in the behaviour of agents, and they abstract away from factors outside the system. In this section we discuss the mathematical foundation for applying these modelling methods. We can only give a very brief summary here; more detail may be found in the papers referenced throughout the section.

An agent, situated within a system that contains also other agents, may establish a part of the system, or a collection of other agents within the system, that it trusts. Similarly, a system's designer or manager might establish a collection of parts of the system such that, within any given part, the agents trust one another. We shall refer to such a part of the system, or such a collection of agents, as a 'trust domain'.

It follows that, in order to model trust domains mathematically, it is necessary to establish an appropriate mathematical handling of them within a framework that models the underlying system. In the Trust Domains project, we have chosen to build upon the systems modelling framework that has been established mathematically in [CMP12, CMP10, CP09, CMP09] and deployed as a practical decision-support tool, as described in, for example, [BCG$^+$08, BPS10, CPW14].

The notion of process has been explored in some detail by the semantics community. Concepts like *resource* and *location* have, however, usually been treated as second class ([Mil09] is a partial exception). Whilst there are some good theoretical reasons to do this, in [CP09, CMP09, CMP10, CMP12] we explore what can be gained by developing an approach in which the structures present in modelling languages are given a rigorous theoretical treatment as first-class citizens. We ensure that each component – location, resource, and process – is handled compositionally. In addition to the structural components of models, we consider also the environment within which a system exists:

Environment: All systems exist within an external environment, which is typically treated as a source of events that are incident upon the system rather than being explicitly stated. Mathematically, environments are represented stochastically, using probability distributions that are sampled in order to provide such events [CP09, CMP09, CMP10, CMP12].

The key structural components are considered, drawing upon classical distributed systems theory — see, for example, [CDK00]:

Location: Places are connected by (directed) links. Locations may be abstracted and refined provided the connectivity of the links and the placement of resources is respected. Mathematically, the axioms for locations [CMP12] are satisfied by various graphical and topological structures, including simple directed graphs and hyper-graphs [CP09, CMP09, CMP10, CMP12];

Resource: The notion of resource captures the components of the system that are manipulated by its processes (see below). Resources include things like computer memory, system operating staff, or system users, as well as money. Conceptually, the axioms of resources are that they can be combined

and compared. We model this notion using *(partial commutative) resource monoids* [OP99, CMP12]: structures $\mathbf{R} = (\mathbf{R}, \sqsubseteq, \circ, e)$ with carrier set $\mathbf{R}$, preorder $\sqsubseteq$, and partial binary composition $\circ$ with unit e, and which satisfies the *bifunctoriality condition*: $R \sqsubseteq R'$ and $S \sqsubseteq S'$ and $R \circ S$ is defined implies $R' \circ S'$ is defined and $R \circ S \sqsubseteq R' \circ S'$, for all $R, S, R', S' \in \mathbf{R}$;

Process: The notion of process captures the (operational) dynamics of the system. Processes manipulate resources in order to deliver the system's intended services. Mathematically, we use algebraic representation of processes based on the ideas in [Mil83], integrated with the notions of resource and location [CP09, CMP09, CMP10, CMP12].

Let Act be a commutative monoid of *actions*, with multiplication written as juxtaposition and unit 1. Let $a, b \in \mathrm{Act}$, etc., so that their multiplication is written ab, etc.. The execution of models based on these concepts, as formulated in [CMP12], is described by a transition system with a basic structural operational semantics judgement [Mil83] of the form $L, R, E \xrightarrow{a} L', R', E'$, which is read as 'the occurrence of the action a evolves the process E, relative to resources R at locations L, to become the process E', which then evolves relative to resources R' at locations L''. The meaning of this judgement is given by a structural operational semantics [Mil83]. The basic case, also know as 'action prefix', is the rule

$$\frac{}{L, R, a : E \xrightarrow{a} L', R', E'} \quad \mu(L, R, a) = (L', R').$$

Here μ is a 'modification' function from locations, resources, and actions (assumed to form a monoid) to locations and resources that describes the evolution of the system when an action occurs. Suppressing locations for now, a partial function $\mu : \mathrm{Act} \times \mathbf{R} \to \mathbf{R}$ is a *modification* if it satisfies the following conditions for all a, b, R, S: $\mu(1, R) = R$; if $R \circ S$ and $\mu(a, R) \circ \mu(b, S)$ are defined, then $\mu(ab, R \circ S) = \mu(a, R) \circ \mu(b, S)$.

There are also rules giving the semantics to combinators for concurrent composition, choice, and hiding — similar to restriction in SCCS and other process algebras (e.g., [Mil83]) — as well for recursion. For example, the rule for synchronous concurrent composition of processes is

$$\frac{L, R, E \xrightarrow{a} L', R', E' \qquad M, S, F \xrightarrow{b} M', S', F'}{L \cdot M, R \circ S, E \times F \xrightarrow{ab} L' \cdot M', R' \circ S', E' \times F'},$$

where we presume, in addition to the evident monoidal compositions of actions and resources, a composition on locations (here written as $\cdot$). The rules for the other combinators, with suitable coherence conditions on the modification functions, follow similar patterns [CMP12]. Note that our choice of a synchronous calculus retains the ability to model asynchrony [Mil83, dS85] (this doesn't work the other way round).

For another example, a key process construct for this paper is non-deterministic choice or sum:

$$\frac{L,R,E_i \xrightarrow{a_i} L',R',E'}{L,R,E_1+E_2 \xrightarrow{a_i} L',R',E'} \quad i=1,2.$$

For example, suppressing location and resource, a process $(a{:}E)+(b{:}F)+(c{:}G)$ will evolve to become E, F, or G depending on the next action being a, b, or c.

Along with the transition system described here comes, in the sense of Hennessy–Milner [HP80, CMP12], a modal logic [CMP12] with basic judgement

$$L,R,E \models \phi, \text{where the proposition } \phi \text{ expresses a property}$$

where the proposition ϕ expresses a property of the process E executing with respect to resources R at location L. This logic includes, in addition to the kinds of connectives and modalities usually encountered in process logics, a number of substructural connectives and modalities that are helpful in reasoning compositionally about resource-bounded systems [CMP12, CP09]. This modal logic can also be extended to the stochastic world. An account of this logic and its extensions is beyond our present scope, but will be of interest in further work.

In order to model trust domains, we work with a version of this modelling framework that is enriched in two ways:

- First, rather than consider a simple process evolution, $E \xrightarrow{a} E'$, we consider processes that evolve in a given surrounding context, C, which also evolves: essentially, $C(E) \xrightarrow{a} C'(E')$ (see [ACP13b, ACP13a] for details);
- Second, and critically, we enrich the calculus with a notion of utility, so that associated with an evolution $C(E) \xrightarrow{a} C'(E')$ is cost, K, determined by the agent E's utility function, K_E, which records the cost associated

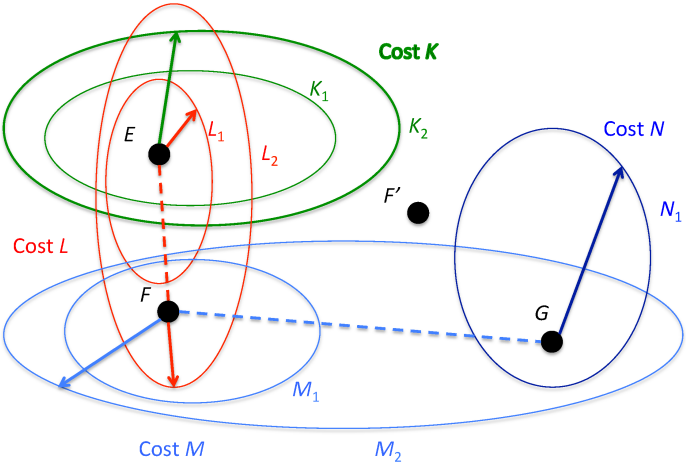


Figure 5.4: Iso-utilities and Trust Domains.

with each choice made during the execution. The theory of this process-utility calculus is presented in [ACP13b, ACP13a].

So, within a system model, where an agent is represented as a process, at any given point in the agent's execution, the process is associated with a location (which we suppress for now) within the system and has access to a collection of resources. That is, the agent has a state. As described above, the agent is also associated with a utility function. Here we interpret the utility function as a loss function, associating a *cost* $k_E(a_i)$ with each choice a_i that is made as a process executes, so that the trace σ of the process that describes agent E gives the total cost K of an agent E's execution:

$$K_E(\sigma) \quad = \quad \sum_{\sigma=a_1,\ldots,a_k} k_E(a_i). \tag{5.1}$$

For now, we consider just finite traces.

Figure 5.4 illustrates the intuition underpinning the concept of a trust domain using these concepts: The agent E may be given one of two different choices of cost (utility) function. If $K_E = K$, then F is not within E's trust domain at either the K_1 or K_2 levels. If, however, $K_E = L$, then F is within

E's trust domain at the L_2, but not at the L_1 level. Agent F's cost function, M, includes agent G at the M_2 level, but not at the M_1 level ($M_1 \leq M_2$). F' is in no-one's domain at any of the given levels of utility.

The formal definition of a trust domain is set up, using the process-utility calculus, for an agent E together with a property ϕ required (by the agent or by the designer/manager) of the part of the system or collection of agents that is to be trusted, the agent's utility function K_E, which assigns values to choices made as the agent executes, and a bound K on the total cost of the trace, which characterizes the total acceptable cost to the agent in reaching or interacting with other parts of the system or other agents within it.

The trust domain is then constructed as a collection of contexts within which the agent may evolve whilst maintaining the properties by which it determines trust. Two properties are required to establish a viable definition. First, a bound K on the cost that E is prepared to incur. Second, a propositional assertion ϕ about the state to which E can evolve within that cost constraint. So, if R is the resource initially associated with E, then — greatly simplifying for this report — we can define, where σ is a trace or sequence of actions,

$$\begin{aligned} \mathrm{TD}(E,\phi,K_E,K) \quad = \quad \Big\{ C \mid & \text{there exist closed } F \text{ and trace} \\ & \sigma \text{ such that } R,E \xrightarrow{a} S,F \\ & \text{and} \quad S,F \models_{C'} \phi \\ & \text{and} \quad K_E(\sigma) \leq K \Big\}, \end{aligned} \tag{5.2}$$

where the resource S is that which is derived from R by the the trace σ and $C(E) \xrightarrow{\sigma} C'(F)$. The details may be found in [ACP13b, ACP13a].

* * *

In this chapter we have discussed how modelling helps in the development and evaluation of trust domains. We have seen how modelling for verification can help to ensure desired properties of a protocol, of components, and of complete systems, and how it can help to ensure that no undesired properties exist. We have then discussed how evaluation can help to improve decision-making, both in the design phase and in the run-time phase of a system. In the design phase, modelling helps to optimise within the tradeoff between performance and security, by illustrating the consequences of choices on the information flow within the system and on performance and security metrics. At run-time, modelling helps to understand the consequences of decisions with respect to

sharing data. Our modelling illustrates the information flows that are likely to result from sharing data with other parties, and thus helps the user to avoid decisions that may result in unintended flows.

This chapter is based on the Trust Domains deliverable D3.2 [d3.13] and on Milestone M5 [RBA13], as well as on numerous academic publications, referenced throughout. As we only give an overview here, we advise the reader to consult the original publications for the details.

Chapter Six

Trust Domains in Practice

In this chapter we discuss approaches for supporting and implementing trust domains. The approaches discussed here span a broad range, from non-technical communication to technical frameworks for specific use-cases of trust domains. In particular, we discuss the following:

- Game-based communication of issues, requirements, and tradeoffs of trust domains.
- Evaluation of security and performance issues in a setting without specific technical support for trust domains.
- Support for trust domains in the setting of conference organisation.
- Support for trust domains in a general collaboration setting.

We may distinguish application-scenarios for these approaches as illustrated in Table 6.1: On the one hand, we may consider whether the workflow is structured and well-known in advance, or whether it is inherently unstructured. On the other hand, we can differentiate between scenarios where the members of the trust domain are determined in an ad-hoc fashion at run-time or where they are known in advance.

Figure 6.1 illustrates the general flow of the chapter and the questions that the approaches address: The Game, as presented in Section 6.1, helps to understand and explore the principal/agent problem inherent in supporting trust domains by security measures measures in general situations without particular reference to any technology. Model-based evaluation is applied in Section 6.2 to support high-level decision-making when building systems that require trust

	Structured Workflow	*Unstructured Workflow*
Ad-hoc Membership	—	Trustworthy Collaboration System
Pre-set Membership	The Game, AMG ConfiChair	—

Table 6.1: Support for Trust Domains, classified by the nature of the workflow (structured/unstructured) and the nature of domain memberships (ad-hoc or pre-set).

domains. Sections 6.3 and 6.4 then present software packages that specifically support trust domains in two application domains.

6.1 Communication and Customer Engagement: The Game

Trust domains often need to be established in complex scenarios comprising various levels of hierarchy in different organisations. Technology is deployed to address concerns that might undermine trust, yet ultimately trust is a subjective assessment made by the parties involved.

In order to establish trust and trustworthiness, organisations typically set policies on the handling of data and processes. However, such policies are rarely enforced, usually only in exceptional cases, which leads employees to interpret policy according to personal priorities and obligations, for example the need to share information quickly. This creates tension between the employee and the organisation when, with the best intentions in mind, employees take 'shortcuts' that place the organisation at risk. Understanding how these risks arise, and more importantly the likely extent and impact they may have on an organisation, is key to managing information security, whether that involves setting effective policy or deploying technology as control points that enforce certain behaviours.

Viewed from an economics perspective, the relation between organisation and employees is a classic example of the principal-agent problem [MCWG95]. In the worst case tensions between the principal and the agent create asymmetries that exhibit moral hazard behaviours [Hol79], where agents subject the principal to undesirable (i.e. excessively risky and untrustworthy) actions sim-

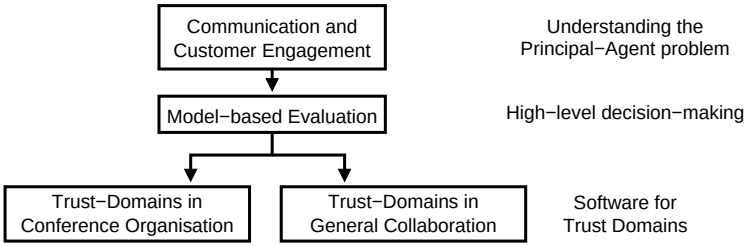


Figure 6.1: General Structure of the Chapter.

ply because they do not adequately understand the effect they have on the principal. Likewise, the impact of policy settings may not be immediately obvious to the principal. The empirical study in Section 2.4 (Chapter 2) illustrated that the principal-problem affects trust domains. Employees typically mean well, but conflicting priorities may make it difficult to perform tasks both securely and efficiently.

Gamification is an emergent approach that utilises principles from game-playing to solve practical problems (for a recent overview, see [HKS14]). By abstracting away from particular technical aspects, gamification helps to understand the general features of the problem, and to find new solutions. In the following we present a simple game developed to illustrate the issues, requirements and tradeofs of establishing and enforcing trust domains, particularly addressing the principal-agent problem. This game is used by HP Enterprise Services (HPES) to walk the customer through the decision-making process and explore the role of positive intervention. This helps the customer better understand their situation and allows them to discover and explore tradeoffs in an intuitive manner without necessarily referring or constraining to technical solutions.

The game is based on classical board-game principles in which a token moves between fields on a board, the goal being to successfully reach the end field. Each field represents a step that corresponds to a workflow action targeted on the information-sharing goal. Each field carries with it a certain probability that a security breach may occur when the token is on the field. The agent controls the token and receives a reward for every successful traversal of the board in which they reach the end goal (field). The principal is also rewarded for successful traversals by the agent. Additionally, however, the principal pays a penalty every time a security breach occurs. The principal can

try to control the agent's behaviour by either blocking fields or by specifying penalties that the agent has to pay should a security breach occur.

Both the agent and the principal try to maximise their rewards, but their incentive-structures and their options differ. The agent is primarily concerned with traversing the board quickly (in order to maximise reward) and can control his actions directly, whereas the principal must take into account penalties arising from actions by the agent, which he can only control indirectly. The customer (acting as the player) takes the role of the principal, while the agent is simulated by an ant-routing algorithm [GSB02] that searches for an optimal route across the board. By choosing different cost-functions and weights in the routing algorithm, different agent personas (e.g. risk-averse, risk-affine) can be implemented and their interaction with controls be explored.

The version of the game discussed in the following is computer-based. The game can also be performed as a traditional board game. In this version, the roles of the principal and the agents are played by human players, and random events are simulated by dice or by a computer. The board-game version provides for more intuitive interactions and may be more appropriate in some contexts. On the other hand, the computer-based version is more efficient at exploring different options.

The game can be tailored to the customer's specific application-domain. In the following, we present its application in a Bring-Your-Own-Device scenario. Scenarios of this kind, where employees use their own devices to access company data, have become increasingly popular in recent years, but introduce additional security risks [Inf].

The Game in the Bring Your Own Device (BYOD) Information-Sharing Scenario In this example, employees of a company need to send important messages with highly sensitive content. The company allows different ways of accessing its infrastructure: First, the employee can send the message from the company machine, which is located within the company's office building and connected to the wired company network. Second, the employee may use their own laptop, either by logging into the webmail interface of the company or by connecting to the company's mail server through the company's virtual private network (VPN). Finally, the employee may also use their own smartphone to send the message, either by logging into the company's webmail interface, or by sending the message using their own private mail account. These means of transmitting messages have different probabilities of leaking the transmitted data. It is reasonable to assume that the traditional way of sending the message

within the company building is most secure, whereas the private mail account is likely to be compromised. On the other hand, the more secure options require access to the company building and thus involve a higher time overhead than the least secure option. Therefore, the company may be considering implementing a strategy that allows employees to utilise their own devices, but needs to understand the impact of such a strategy on the security of its information.

In this scenario the *agent* is the employee, and their goal is to transmit the message as fast as possible, while also avoiding potential penalties by their supervisors. The choices of the agent are displayed by lighting-up of the paths that they are most likely to take. The *principal* is the company owner, who tries to optimise the company's profit. The profit is affected by the employee's performance in transmitting messages (i.e., better communication will increase the company's income), but also by penalties imposed by the authorities, e.g. the UK's Information-Commissioner's Office (ICO). The authorities are assumed to issue a financial penalty each time a message's contents are leaked, and this financial penalty is substracted from the company's income. The principal may set policies that discourage the employees from using certain modes of transmitting information, and enforce these policies by penalising them for digressions. Furthermore, the principal can also block some ways of data transmission by technical means, e.g. not accept mail from employee's private devices. The game allows the player to take the principal's point of view and employ both of these mechanisms to explore their impact on the behaviour of employees and on the company's financial performance.

6.2 Model-based Evaluation

In this section we illustrate how the modelling approaches described in Section 5.2.2 can be used in the evaluation of the tradeoffs between security and performance when establishing trust domains in a scenario where there is no specific technology support. That is, while we assume that there are mechanisms for securing and restricting information-flows, unlike with the approaches described in Sections 6.3 and 6.4 these mechanisms are not integrated in a solution that revolves around trust domains principles.

We use the Mergers and Acquisitions (M&A) example, as introduced in Chapter 1: Two companies want to merge, and are supported by a bank, which employs the services of external accountants and lawyers to help with some

steps of the process. From the point of view of the information flows between different organisations, this process can be broken up into seven steps:

1. The bank contacts the companies to obtain required data.
2. The companies respond with the data.
3. The bank forwards the data to a external analysts.
4. The external analysts respond with their reports.
5. The bank contacts a lawyer and requests a contract to be drawn up.
6. The lawyer responds with a contract.
7. The bank forwards the contract to the companies to be signed.

For simplicity, we assume that in each of these steps the sender has the choice between the same three means of transmitting the data:

Print and Post. With this mode, the sender first prints the documents and then uses a courier service to deliver it to the recipient. Whilst the transmission is secure from leakage, this mode is also very slow.

Public Cloud Document-Sharing Service. With this mode, the sender uploads the documents to a service in a public Cloud, such as DropBox or Google-Docs, whence the receiver can download them. This mode is very fast, but can be expected to be very insecure, as the provider of the public service has access to the data, and attackers might also be able to gain access.

Virtual Clean-Room via Locked-Down Virtual Machine. With this mode, the data is stored on a server at the bank. The server can only be accessed by a virtual machine provided by the bank. This virtual machine is locked down such that data cannot be exfiltrated from it by an attacker. This mode is very secure, but involves additional start-up costs for installing and running the virtual machine, and is therefore relatively slow, when compared to the Cloud service.

The properties of the three transmission modes are reflected in the parameter settings on their outgoing edges: Highly secure modes have higher attack costs than insecure ones, and faster modes have higher information-flow rates than slower modes.

We furthermore assume that the workflow has to be finished within a certain timespan. We then want to find policy settings that discourage users from using insecure modes, whilst also ensuring that the process is finished within

the allotted time. The penalty settings on the arcs are used to express different policy choices, i.e. we assign high penalties for connections to/from nodes that are undesirable. The cost function is defined such that it returns very high costs if the time available for the workflow would be exceeded, and a weighted sum of the expected time to completion and the penalties otherwise.

Figure 6.2 illustrates the resulting information flows for one setting. In all steps, the bank is on the left-hand side, while the other party is on the right-hand side. The policies are set such that the virtual clean-room solution is preferred. We observe that in Steps 1–6 the parties follow this policy. However, in Step 7 the data is being transmitted through the Cloud service. This is caused by the fact that in this step the virtual clean-room solution is too slow to enable the completion of the workflow, and a faster solution is required. This can be interpreted as the case where the employees run out of time and therefore use a faster method, even though it presents a higher risk. We could try to enforce compliant behaviour by more restrictive policies. In this scenario, however, we would not be able to find a better solution, as the timespan for the process is too short to be finished by exclusive use of the virtual clean-room.

6.3 Trust Domains in Conference Organisation: ConfiChair

In this section we describe the ConfiChair conference reviewing system. Our discussion follows the presentation in [ABR12]. In [ABR13] the approach has been generalised to a wider range of competition problems, such as procurement or employment.

The ConfiChair approach addresses the problem of managing the review process for scientific conferences using an untrusted Cloud provider for storage and processing. The goals of the system are to ensure that the Cloud provider cannot access the contents of papers and reviews stored on its platform, and that it cannot infer reviewer-author relationships, i.e. that the Cloud provider cannot infer which reviewer reviewed which paper. At the same time, the Cloud provider should be able to provide storage and processing, such as the forwarding of data and the computation of statistics. The system achieves this by employing strong encryption on contents of papers and reviews and by using anonymised identifiers that are mixed between steps. The process is described in detail in [ABR12, ABR13]; here, we consider it from the perspectives of the users of the system:

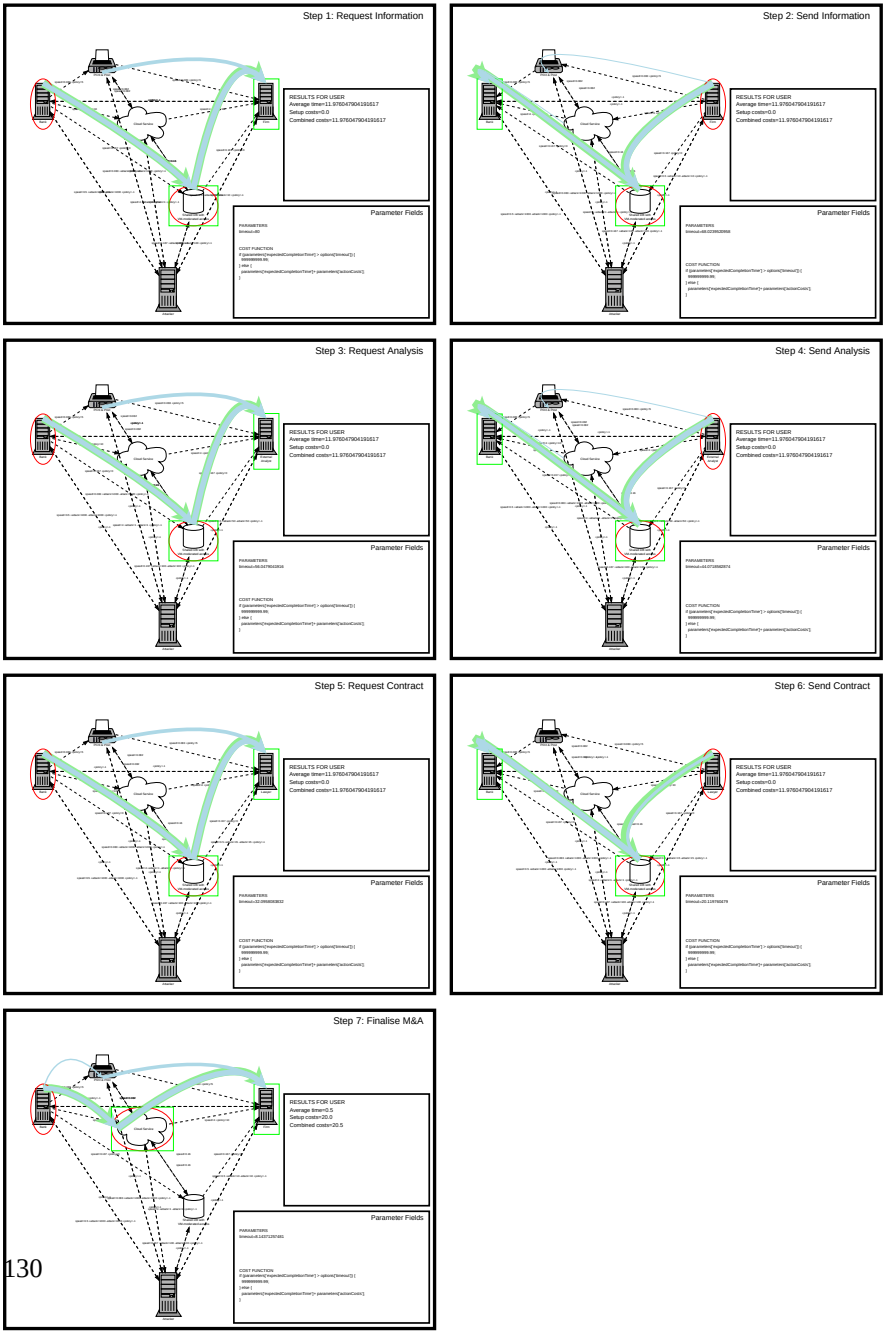


Figure 6.2: Information flows in the M&A setting, modelled by the AMG tool.

The *conference chair* first creates a symmetric key K_{conf} and a public/private key pair for the conference and sends K_{conf} to the reviewers through a channel that cannot be accessed by the Cloud provider. The chair then waits for papers to be submitted by the authors. Each *author* encrypts their paper with their own key and encrypts this key with the public key of the conference. Both the encrypted paper and the encrypted key are then uploaded. During the paper submission phase, a database of paper keys encrypted with the public conference key and a database of papers encrypted with the paper keys are created at the Cloud. Each paper in the database is assigned a unique identifier λ. In order to initiate the review phase, the *chair* downloads the key database, decrypts its contents using the private conference key, assigns a new identifier μ to each identifier λ, and re-encrypts the contents with the symmetric conference key K_{conf}. The modified database is then uploaded to the Cloud. Referring to the entries of this modified database, the chair assigns reviewers to papers using the identifiers μ. The Cloud then notifies reviewers of the papers that have been assigned to them and forwards the encrypted keys for these papers. The *reviewers* download the database of encrypted papers, decrypt the encrypted paper keys (using the shared conference key), and decrypt the papers assigned to them. After reviewing, they encrypt their reviews with the conference key and upload them again. The reviews are referred to by the identifiers μ, and hence the Cloud cannot infer which paper (identified by identifier λ) a particular reviewer has accessed. In the final stage, the *chair* creates a notification for each paper λ, encrypts it with the paper's key, and requests the Cloud to forward the notification to the appropriate author.

Formal Verification of Security Properties ConfiChair aims to protect contents of papers and reviews from being accessed by the Cloud, and to prevent the Cloud from linking authors and reviewers. These properties are formally verified in [ABR12], using the ProVerif tool [BAF08, Pro]. Here, we give a very short summary of the underlying modelling method

In the proof of the required properties, the concept of *observational equivalence* [RS11] is applied. That is, it must be shown that two executions of the protocol with different inputs are indistinguishable from the point of view of the Cloud. For instance, in order to prove that the contents of papers cannot be accessed by the Cloud, it must be proven that the Cloud provider cannot distinguish an instance where an author has submitted the paper P from an instance where the same author has submitted a different paper P'.

In order to prove the desired properties, the ConfiChair protocol is modelled using the ProVerif process calculus. In this calculus the evolution of the system is described in terms of processes, messages and term evaluations. The resulting model, along with the desired property (i.e., observational equivalence), is then submitted to the ProVerif tool, which returns whether the property holds.

In the ConfiChair system, trust is established by the users trusting in the security of the protocol, and trust in the security of the system is based on trust in the validity of the proof. The security of the protocol has been proven by formal verification in [ABR12]. ConfiChair thus enables the creation of trust domains with guaranteed security properties. Note, however, that in order for the proof to be valid the workflow must be known beforehand, and it must be guaranteed that all participants follow this workflow. If the former does not hold, then the model cannot be formulated, if the latter is not satisfied, then the protocol that is executed is not the same as the one represented by the model, and therefore the statements derived from the proof do not hold. ConfiChair and the modelling applied to prove its properties thus support trust domains in scenarios where there is a fixed, well-known workflow.

6.4 Trust Domains in General Collaboration: TCS

In this section we discuss the Trustworthy Collaboration System (TCS), a system for general, light-weight collaboration between individuals and organisations. The TCS supports the user in sharing data in a trustworthy way by providing mechanisms for secure sharing and model-based decision support both at design-time and at run-time of the system. In contrast to the ConfiChair system discussed in Section 6.3, the Trustworthy Collaboration System assumes an open scenario where only very little control can be exerted on users. In such a system the establishment of trust is vital to the operation of the system; at the same time, the flexibility inherent to the system proves challenging.

The collaboration platform offers elegant sharing options together with an attractive user interface that can be used across a wide range of stationary and mobile devices, thus providing the user with an easy, efficient way of sharing data and documents. The platform complements these attributes with explicit support for trusted sharing of information, made possible by the following features:

1. Model-based decision support during setup. Users wishing to set up a collaboration will be offered a number of different options and also need to decide on membership criteria. Users are supported in their choice of enabled context groups by a model-based approach that predicts and displays the performance, dependability, and security properties arising from a particular choice of context groups and anticipated user characteristics.
2. Model-based decision support during operation. During operation, user's choices of who to share documents with when and using which method are supported by model-based predictions of the resulting trust properties. Users can therefore make informed decisions based on whether the predictions are sufficient to meet their expectations.
3. Monitoring of the security state of client devices. The collaboration platform aims to monitor the security state of client devices, and uses the state information to display the current security properties of the system. Using this information, users can assess whether the system is sufficiently trustworthy to be used for the specific task at hand. Furthermore, monitoring data is used both in establishing a provenance trail and in predicting the impact of decisions on the trust properties of the system

In the following we first describe the scenario we use for illustrating the principles of the Trustworthy Collaboration System and then discuss the operation of the system.

6.4.1 Scenario Description

In our discussion of the TCS we focus on the following application-scenario: An individual giving a presentation wants to present some documents and data for access via a web browser, as some parts of the audience may not be present in the same location. The presenter does not trust everyone in the audience to handle all of the presented material securely, and therefore would like to be able to restrict access to some parts of the presentation. Furthermore, the presenter wants to be able to base the decisions on who to share with on a measure of the trustworthiness of that person.

6.4.2 TCS Workflow

Figure 6.3 illustrates the workflow when using the Trustworthy Collaboration System: First, the creator of the trust domain (who need not be identical to

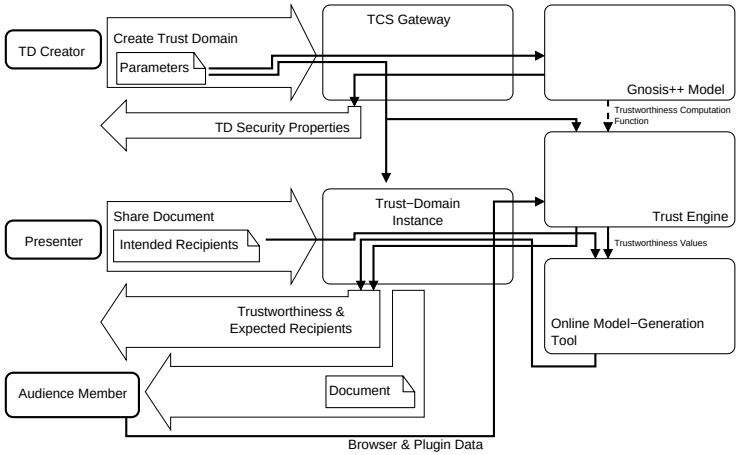


Figure 6.3: General schema of the TCS System.

the presenter) connects to the TCS Gateway and requests a new TCS instance. The creator can then customise the instance by selecting the *access-control level* and by parameterising the *trustworthiness computation functions*. The access-control level can be chosen between public, semi-private and private. The trustworthiness computation function is used in the *Trust Engine* to compute the trustworthiness of connected users. The function bases the trustworthiness of the user on the browser they are using and on the list of browser plugins. The creator parameterises the function by assigning trustworthiness values to a selection of browsers and plugins. These values can be based on the creator's subjective assessment of the trustworthiness of each of these choices, and are generally guided by the policy of the organisation. For instance, an organisation may stipulate that only the Firefox Web browser without the Flash plugin is considered secure, and the creator would then set the trustworthiness values accordingly.

In the next step the creator requests feedback on the security of the chosen settings. This data is generated by evaluating a Gnosis++ model in the backend. The Gnosis++ model represents attendees as agents, using templates to describe classes of attendess, and simulates the behaviour of the agents during the meeting and for a time afterwards. Agents may share the data, or they may leak it inadvertantly. The likelihood of leakage is influenced by the chosen

parameters, i.e. a more secure choice will lower the likelihood of leaks occurring. From evaluation of the model the distribution of the number of leakages is obtained and displayed to the creator.

At run-time the presenter connects to the newly created TCS instance and invites audience members, by communicating the URL of the instance to the audience. The URL may be communicated in text form, or it may be shown as a QR code, which enables easy sharing of the URL information when audience members are in the same location. Audience members connect using their browser and are then shown the shared material. As the presenter progresses through the material, the display is being updated on the audience's devices automatically. The TCS monitors the browser and the browser plugins employed by the user and stores this data in the Trust Engine. This data can easily be obtained from the HTTP headers sent by the browser.

For each page or slide the presenter may select which audience members it is to be shared with, on an individual basis. This choice is made based on the trustworthiness of the recipients and on predictions of the information flow resulting from sharing the document. The trustworthiness of the recipients can be requested from the TCS system and is displayed in a colour-coded way. The value is computed by the Trust Engine, using the trustworthiness computation functions defined earlier. That is, if a user is using a browser/plugin combination that is not considered trustworthy, their trustworthiness value will be low. The information-flows resulting from a choice are displayed by a colour-coded representation of the expected probability of an entity getting access to the data. This data illustrates to the presenter the impact of their sharing decision, i.e., the presenter will be shown who the data may be flowing to as a result of their decision, and can accordingly revise their decision. The information flow is computed using the online-modelling generation tool (OMG Tool), which we described in Section 5.2.3. The model considers three sets of users: The presenter, the set of intended recipients, and the set of unintended recipients. It is assumed that data can flow directly from the presenter to the intended recipients. Each intended recipient is also connected to the set of unintended recipients. The rate of data-flow along each of these latter edges reflects the trustworthiness value of the originator: Intended receipients with a high trustworthiness value have a low rate of leakage, while low trustworthiness implies high leakage rate. By solving the resulting OMG model, likely data flows are identified. These are illustrated by the likely distribution of data. Based on the displayed output, the presenter may then decide not to share a page of the document with a specific member of the audience, or to require them to first

change their browser, such that it complies with the requirements (resulting in an increase in trustworthiness).

* * *

In this chapter we have discussed several practical approaches for trust domains. These span the range from informal methods aimed at improving communication and enabling exploration to systems developed specifically for supporting trust domains in practical scenarios.

Bibliography

[ABR12] Myrto Arapinis, Sergiu Bursuc, and Mark Ryan. Privacy Supporting Cloud Computing: Confichair, a Case Study. In *Proceedings of the First International Conference on Principles of Security and Trust*, POST'12, pages 89–108, Berlin, Heidelberg, 2012. Springer-Verlag.

[ABR13] Myrto Arapinis, Sergiu Bursuc, and Mark Ryan. Privacy-supporting cloud computing by in-browser key translation. *Journal of Computer Security*, 21(6):847–880, 2013.

[ACP13a] G. Anderson, M. Collinson, and D. Pym. Trust domains: An algebraic, logical, and utility-theoretic approach. In *Trust and trustworthy Computing*, volume 7904 of *LNCS*, pages 232–249, 2013.

[ACP13b] G. Anderson, M. Collinson, and D. Pym. Utility-based Decision-making in Distributed Systems Modelling. In Burkhard C. Schipper, editor, *In Proc. 14th TARK*, Chennai, 2013. Computing Research Repository (CoRR): http://arxiv.org/corr/home. ISBN: 978-0-615-74716-3.

[AM14] Nalin Asanka Gamagedara Arachchilage and Andrew Martin. A Trust Domains Taxonomy for Securely Sharing Information: An Empirical Investigation. Submitted to International Symposium on Human Aspects of Information Security & Assurance (HAISA 2014), 2014.

[ANM13] Nalin Asanka Gamagedara Arachchilage, Cornelius Namiluko, and Andrew Martin. A taxonomy for securely sharing information among others in a trust domain. In *Information Science and Technology (ICIST), 2013 International Conference on*, pages 296–304, March 2013.

[APM08] Rocío Aldeco-Pérez and Luc Moreau. Provenance-based Auditing of Private Data Use. In *Proceedings of the 2008 International Conference on Visions of Computer Science: BCS International Academic Conference*, VoCS'08, pages 141–152, Swinton, UK, UK, 2008. British Computer Society.

[ARR11] Myrto Arapinis, Eike Ritter, and Mark D. Ryan. StatVerif: Verification of Stateful Processes. In *Proceedings of the 2011 IEEE 24th Computer Security Foundations Symposium*, CSF '11, pages 33–47, Washington, DC, USA, 2011. IEEE Computer Society.

[BAF08] Bruno Blanchet, Martin Abadi, and Cedric Fournet. Automated Verification of Selected Equivalences for Security. *Journal of Logic and Algebraic Programming*, 75(1):3–51, 2008.

[Bai02] T. Bailey. On trust and philosophy. The philosophy of trust. http://www.open2.net/trust/on_trust/on_trust1.htm, 2002. (Not available anymore.).

[BB08] P. Brantingham and P. Brantingham. Crime Pattern Theory. In R. Wortley and L. Mazerolle, editors, *Environmental Criminology and Crime Analysis*. Collumpton: Willan, 2008.

[BCG+08] A. Beautement, R. Coles, J. Griffin, C. Ioannidis, B. Monahan, D. Pym, A. Sasse, and M. Wonham. Modelling the Human and Technological Costs and Benefits of USB Memory Stick Security. In M. Eric Johnson, editor, *Managing Information Risk and the Economics of Security*, pages 141–163. Springer, 2008.

[Ber10] W. Bernasco, editor. *Offenders on Offending*. Willan., Collumpton, 2010.

[BHM+04] David Booth, Hugo Haas, Francis Mccabe, Eric Newcomer, Michael Champion, Chris Ferris, and David Orchard. Web services architecture. Technical report, W3C, Web Services Architecture Working Group, February 2004.

[Bie09] James L. Bierstaker. Differences in attitudes about fraud and corruption across cultures: Theory, examples and recommendations. *Cross Cultural Management: An International Journal*, 16(3):241–250, 2009.

[BME09] M.L. Benson, T.D. Madensen, and J.E. Eck. White-collar crime from an opportunity perspective. In S. Simpson and D. Weisburd, editors, *The criminology of white-collar crime*, pages 175–193. Springer Science+Business Media, New York, 2009.

[BP07] S. Ba and B. Pavlou. Evidence Of the Effect of Trust Building Technology in Electronic Markets: Price Premiums and Buyer Behavior. *MIS Quarterly*, 26(3):243–268, September 2007.

[BPS10] Y. Beres, D. Pym, and S. Shiu. Decision Support for Systems Security Investment. In *Proc. Business-driven IT Management (BDIM) 2010*. IEEE Xplore, 2010.

[BW06] K. Bussmann and M. Werle. Addressing Crime in Companies: First Findings from a Global Survey of Economic Crime. *British Journal of Criminology*, 46(6):1128–1144, 2006.

[CB05] L. Carter and F. Bélanger. The utilization of e-government services: citizen trust, innovation and acceptance factors. *Information Systems Journal*, 15(1):5–25, 2005.

[CC86] D Cornish and R. Clarke. *The Reasoning Criminal and Rational Choice Perspectives on Offending*. Springer-Verlag, New York, NY, 1986.

[CCW06] Frederick Chong, Gianpaolo Carraro, and Roger Wolter. Multi-Tenant Data Architecture. http://msdn.microsoft.com/en-us/library/aa479086.aspx, 2006.

[CDK00] George Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed Systems: Concepts and Design*. Addison Wesley; 3rd edition, 2000.

[CF79] L.E. Cohen and M. Felson. Social Change and Crime Rate Trends: A Routine Activity Approach. *American Sociological Review*, 44:588–605, 1979.

[CG12] Stephen Crane and Martin Gill. TSB Trust Domains Project Work Package 1: Framework and Usage Scenarios for Data Sharing Deliverable D1.3: Trust Domain Guide Interim Report. Technical report, Perpetuity Research and Consultancy International and HP Labs, 2012.

[Cla95] R. V. Clarke. Situational crime prevention. In M. Tonry and N. Morris, editors, *Building a safer society: Strategic approaches to crime prevention*, pages 91–150. University of Chicago Press, Chicago, IL, 1995.

[CMP09] M. Collinson, B. Monahan, and D. Pym. A Logical and Computational Theory of Located Resource. *Journal of Logic and Computation*, 19(b):1207–1244, 2009.

[CMP10] Matthew Collinson, Brian Monahan, and David Pym. Semantics for Structured Systems Modelling and Simulation. In *Proc. Simutools 2010*. ACM Digital Library, ISBN 78-963-9799-87-5, 2010.

[CMP12] M. Collinson, B. Monahan, and D. Pym. *A Discipline of Mathematical Systems Modelling*. College Publications, 2012.

[Cor] Core Gnosis. http://www.hpl.hp.com/research/systems_security/gnosis.html.

[Cor93] D. Cornish. Theories of Action in Criminology: Learning Theory and Rational Choice Approaches. In R.V. Clarke and M. Felson, editors, *Routine Activity and Rational Choice.Advances in Criminological Theory, Vol. 5*. Transaction Press, New Brunswick, NJ, 1993.

[Cor94a] D. Cornish. The Procedural Analysis of Offending and its Relevance for Situational Prevention. In R. Clarke, editor, *Crime Prevention Studies Volume 3*. Criminal Justice Press, Monsey New York, 1994.

[Cor94b] D. B. Cornish. Crimes as scripts. In D. Zahm and P. Cromwel, editors, *Proceedings of the International Seminar on Environmental Criminology and Crime Analysis*, pages 30–45, Coral Gables, FL:, 1994. University of Miami, Florida Criminal Justice Executive Institute.

[CP09] M. Collinson and D. Pym. Algebra and logic for resource-based systems modelling. *Mathematical Structures in Computer Science*, 19:959–1027, 2009. doi:10.1017/S0960129509990077.

[CPB03] M. Casassa 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 Labs Bristol, March 19 2003.

[CPW14] T. Caulfield, D. Pym, and J. Williams. Compositional security modelling: Structure, economics, and behaviour. In *Proceedings of the Foundations, Tools, and New Concepts in Trusted Computing Track, 2nd International Conference on Human Aspects of Information Security, Privacy and Trust, HCI International 2014, Heraklion, June 2014*. LNCS, Springer. Preprint at: http://www.cs.ucl.ac.uk/staff/D.Pym/CaulfieldPymWilliams.pdf, 2014.

[Cre50] D.R. Cressey. The criminal violation of financial trust. *American Sociological Review*, 15(6):738–743, 1950.

[Cre53] D.R. Cressey. *Other People's Money: A Study in the Social Psychology of Embezzlement*. Glencoe: The Free Press, 1953.

[d3.13] Confidence indicators and representation of trusted components and trust domains. Trust Domains Deliverable D3.2. Technical report, 2013.

[DG07] M. Duffin and M. Gill. Staff Dishonesty: A Report for Procter and Gamble. Perpetuity Research and Consultancy International, 2007.

[DG08] J.J. Donegan and M.W. Ganon. Strain, differential association, and coercion: Insights from the criminology literature on causes of accountant's misconduct. *Accounting and the Public Interest*, 8:1–20, 2008.

[DHP07] C. Dwyer, S. Hiltz, and K. Passerini. Trust and Privacy Concern Within Social Networking Sites: A Comparison of Facebook and MySpace. In *Americas Conference on Information Systems (AMCIS) AMCIS 2007 Proceedings*, 2007.

[Dri78] J. W. Driscoll. Trust and participation in organizational decision making as predictors of satisfaction. *The Academy of Management Journal*, 21(1):44–56, 1978.

[dS85] R. de Simone. Higher-level synchronising devices in Meije-SCCS. *Theoretical Computer Science*, 37:245–267, 1985.

[Ekb10] P. Ekblom. The Conjunction of Criminal Opportunity Theory. In *Sage Encyclopedia of Victimology and Crime Prevention*. Sage, 2010.

[Ekb11] P. Ekblom. *Crime Prevention, Security and Community Safety Using the 5Is Framework*. Palgrave, Basingstoke, 2011.

[ENS07] R. G. Eccles, S. C. Newquist, and R. Schatz. Reputation and Its Risks. *Harvard Business Review*, 85(2):104–114, February 2007.

[Far05] David B. Farber. Restoring Trust after Fraud: Does Corporate Governance Matter? *The Accounting Review*, 80(2):539–561, 2005.

[Fel02] M. Felson. *Crime and Everyday Life*. Sage: Thousand Oaks, 2002.

[Fuk95] Francis Fukuyama. *Trust: The social virtues, and the creation of prosperity*. The Free Press, New York, 1995.

[GC13] Martin Gill and Stephen Crane. TSB Trust Domains Report. Work Package 1: Framework and Usage Scenarios for Data Sharing. Deliverable D1.x1: Offenders' approaches to offending: what we can learn about threats to trust domains from criminology, crime prevention and criminals. Technical report, Perpetuity Research and Consultancy International and HP Labs, January 2013.

[GC14] M. Gill and S. Crane. Generators of and impediments to trust in an information sharing environment: thinking about the creation of trust domains. (unpublished), 2014.

[GGW10] M.L. Gill and J.E. Goldstraw-White. Theft and Fraud by Employees. In F. Brookman, M. Maguire, H. Pierpoint, and T.H. Bennett, editors, *Handbook of Crime*. Willan, Uffculme, Devon, 2010.

[GGW12a] M. Gill and J Goldstraw-White. Motives. In A. Doig, editor, *Fraud: The Counter-Fraud Practitioner's Handbook*. Gower, Farnham, 2012.

[GGW12b] M. Gill and J. Goldstraw-White. Why people commit fraud. In A. Doig, editor, *The Handbook of Fraud Investigation*. Gower, London, 2012.

[Gil05a] M. Gill. Learning From Fraudsters. Available from www.perpetuitygroup.com/prci/publications.html, 2005.

[Gil05b] M. Gill. Reducing the capacity to offend: Restricting resources for offending. In N Tilley, editor, *The Handbook of Crime Prevention and Community Safety*. Willan, Uffculme, Devon, 2005.

[Gil07] M. Gill. Learning From Fraudsters: Reinforcing the message. Available from www.perpetuitygroup.com/prci/publications.html, 2007.

[Gil11a] M. Gill. Learning From Offenders' Accounts of their Offending. *Prison Service Journal*, (194):27–32, March 2011.

[Gil11b] M. Gill. Should we expect more frauds in a recession? Views from both fraud managers and fraudsters. *International Journal of Law, Crime and Justice*, 39(3):204–214, 2011.

[GMF+03] J. Gennari, M. Musen, R. Fergerson, W. Grosso, M. Crubezy, H. Eriksson, N. Noy, and S. Tu. The evolution of Protege-2000: An environment for knowledge-based systems development. *International Journal of Human-Computer Studies*, 58(1):89–123, 2003.

[GR03] Tal Garfinkel and Mendel Rosenblum. Virtual Machine Introspection Based Architecture for Intrusion Detection. In *Proceedings of the Network and Distributed Systems Security Symposium*, pages 191–206, 2003.

[GSB02] Mesut Güneş, Udo Sorges, and Imed Bouazizi. ARA - The Ant-Colony Based Routing Algorithm for MANETs. In *ICPPW '02: Proceedings of the 2002 International Conference on Parallel Processing Workshops*, Washington, DC, USA, 2002. IEEE Computer Society.

[GW12] J. Goldstraw-White. *White Collar Crime: Accounts of Offending Behaviour.* Palgrave, Basingstoke, 2012.

[Har06] R. Hardin. *Trust*. Polity Press, Malden, MA, 2006.

[Hav98] B. R. Haverkort. *Performance of Computer Communication Systems: A Model-Based Approach*. John Wiley & Sons, Chichester, UK, 1998.

[HBA+12] Keith Harrison, Behzad Bordbar, Syed T.T. Ali, Chris I. Dalton, and Andrew Norman. A Framework for Detecting Malware in Cloud by Identifying Symptoms. In *Proceedings of the 16th International Enterprise Distributed Object Computing Conference*, pages 164–172, 2012.

[HKS14] J. Hamari, J. Koivisto, and H. Sarsa. Does Gamification Work? A Literature Review of Empirical Studies on gamification. In *Proceedings of the 47th Hawaii International Conference on System Sciences*, Hawaii, USA, 6–9 January 2014.

[HLMS10] Fang Hao, T. V. Lakshman, Sarit Mukherjee, and Haoyu Song. Secure Cloud Computing with a Virtualized Network Infrastructure. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, pages 16–16, Berkeley, CA, USA, 2010. USENIX Association.

[Hol79] Bengt Holmstrom. Moral Hazard and Observability. *Bell Journal of Economics*, 10(1):74–91, Spring 1979.

[HP80] M. Hennessy and G. Plotkin. On Observing Nondeterminism and Concurrency. In *Proceedings of the 7th ICALP*, volume 85 of *Lecture Notes in Computer Science*, pages 299–309. Springer-Verlag, 1980.

[Inf] Information Commissioner's Office. Bring your own device (BYOD). http://ico.org.uk/for_organisations/data_protection/topic_guides/online/byod.

[JG05] D. Johnson and K. Grayson. Cognitive and affective trust in service relationships. *Journal of Business Research*, 58(4):500–507, April 2005.

[JOP12] Venkata Josyula, Malcom Orr, and Greg Page. *Cloud computing: Automating the Virtualized Data Center*. Cisco Press, 2012.

[KC12] Gina Kounga and Liqun Chen. Enforcing Sticky Policies with TPM and Virtualization. In *Proceedings of the Third International Conference on Trusted Systems*, INTRUST'11, pages 32–47, Berlin, Heidelberg, 2012. Springer-Verlag.

[KM12] Gina Kounga and Andrew Martin. TrustDomains: A Framework for Modelling and Designing E-Service Infrastructures for Controlled Sharing of Information. Deliverable D4.1b: Framework for Trust Domain Technology. Technical report, Oxford University, August 2012.

[KMR04] H. Knublauch, M. A. Musen, , and A. L. Rector. Editing description logics ontologies with the Protege OWL plugin. In *International Workshop on Description Logics*, Whistler, BC, Canada, 2004.

[KMV+12] A. Kirschenbaum, M. Mariani, C Van Gulijk, S. Lubasz, C. Rapoport, and H. Andriessen. Airport security: An ethnographic study. *Journal of Air Transport Management*, 18:68–73, 2012.

[KR11] Masoud Koleini and Mark Ryan. A Knowledge-based Verification Method for Dynamic Access Control Policies. In *Proceedings of the 13th International Conference on Formal Methods and Software Engineering*, ICFEM'11, pages 243–258, Berlin, Heidelberg, 2011. Springer-Verlag.

[KSW03] Günter Karjoth, Matthias Schunter, and Michael Waidner. Platform for Enterprise Privacy Practices: Privacy-enabled Management of Customer Data. In *Proceedings of the 2Nd International Conference on Privacy Enhancing Technologies*, PET'02, pages 69–84, Berlin, Heidelberg, 2003. Springer-Verlag.

[LAHR11] Michael Hale Ligh, Steven Adair, Blake Hartstein, and Matthew Richard. *Malware Analyst's Cookbook and DVD*. Wiley Publishing Inc., 2011.

[LCP06] Hazel Lacohée, Stephen Crane, and Andy Phippen. Trustguide Final Report. www.trustguide.org, October 2006.

[Lev08] Michael Levi. *The phantom capitalists : the organization and control of long-firm fraud*. Ashgate, Aldershot, 2 edition, 2008.

[LM10] John Lyle and Andrew Martin. Trusted Computing and Provenance: Better Together. In *Proceedings of the 2Nd Conference on Theory and Practice of Provenance*, TAPP'10, Berkeley, CA, USA, 2010. USENIX Association.

[LT03] J. Lacoste and P. Tremblay. Crime and Innovation: a script analysis of patterns in check forgery. In *Theory for practice in situational crime prevention*, volume 16. Criminal Justice Press, New York, 2003.

[Mar08] Andrew Martin. The ten-page introduction to Trusted Computing. Technical Report CS-RR-08-11, Oxford University, 2008.

[MBCD09] Abha Moitra, Bruce Barnett, Andrew Crapo, and Stephen J. Dill. Data Provenance Architecture to Support Information Assurance in a Multilevel Secure Environment. In *Proceedings of the 28th IEEE Conference on Military Communications*, MILCOM'09, pages 2076–2082, Piscataway, NJ, USA, 2009. IEEE Press.

[MC10] Haralambos Mouratidis and Piotr Cofta. Practitioner's challenges in designing trust into online systems. *Journal of Theoretical and Applied Electronic Commerce Research*, 5(3), December 2010.

[MCWG95] Andreu Mas-Colell, Michael D. Whinston, and Jerry R. Green. *Microeconomic Theory*. Oxford University Press, June 1995.

[Mil83] R. Milner. Calculi for Synchrony and Asynchrony. *Theoretical Computer Science*, 25(3):267–310, 1983.

[Mil09] R. Milner. *The Space and Motion of Communicating Agents*. Cambridge University Press, 2009.

[Mit98] M. Mitchell. *An Introduction to Genetic Algorithms*. A Bradford book. MIT Press, 1998.

[MKBN12] John Mace, Dirk Kuhlmann, Adrian Baldwin, and Cornelius Namiluko. Trust Domains Work Package 3: Taxonomy, Description, Confidence Indicators. Deliverable D3.1: Taxonomy, Description, Modelling Options, Confidence Indicators. Interim Report. Technical report, HP Labs and Oxford University, 2012.

[MMH08] Derek G. Murray, Grzegorz Milos, and Steven Hand. Improving Xen Security through disaggregation. In *Proceedings of the 4th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, pages 151–160, March 2008.

[MZD92] C. Moorman, G. Zaltman, and R. Deshpande. Relationships between providers and users of market research: The dynamics of trust within and between organizations. *Journal of Marketing Research*, 29:314–328, August 1992.

[Nat02] National Consumer Council. A Matter of Trust, 2002.

[Neu81] Marcel F. Neuts. *Matrix-Geometric Solutions in Stochastic Models. An Algorithmic Approach*. Dover Publications, Inc., New York, 1981.

[New98] J. Newman. The Dynamics of Trust. In A. Coulson, editor, *Trust and Contracts*. Policy Press, Bristol, 1998.

[NM12] Cornelius Namiluko and Andrew Martin. Provenance-Based Model for Verifying Trust-Properties. In Stefan Katzenbeisser, Edgar Weippl, L. Camp, Melanie Volkamer, Mike Reiter, and Xinwen Zhang, editors, *Trust and Trustworthy Computing*, volume 7344/2012 of *Lecture Notes in Computer Science*, pages 255–272. Springer Berlin/Heidelberg, 2012.

[OP99] P.W. O'Hearn and D.J. Pym. The Logic of Bunched Implications. *Bulletin of Symbolic Logic*, 5(2):215–244, June 1999.

[Ora] Oracle. *Configuring VLANs in Solaris 10 3/05*.

[Ora10] Oracle. *Oracle VM Server for SPARC Enabling a Flexible, Efficient IT Infrastructure*, 2010.

[Ora12] Oracle. *Oracle VM Server for SPARC 2.2 Administration Guide*, 2012.

[Par98] D. Parker. *Fighting Computer Crime*. Charles Scribner's Sons, New York, 1998.

[PCPP03] S. Pearson, L. Chen, D. Plaquin, and G. Proudler, editors. *Trusted Computer Platforms*. Prentice Hall, New Jersey, 2003.

[PG11] K. Pease and M. Gill. *Home Security and Place Design: Some Evidence and its Policy Implications.* Perpetuity Research and Consultancy International., Leicester, 2011.

[Pro] The ProVerif Tool. http://prosecco.gforge.inria.fr/personal/bblanche/proverif/. Last checked 7 March 2014.

[RBA13] Philipp Reinecke, Adrian Baldwin, and Gabrielle Anderson. TSB Trust Domains Project Milestone 5: Theoretical approaches for modelling data flow, processes, technical factors and threats synchronised with design methodology and updated technical framework for Trust Domains. Technical report, HP Labs and University of Aberdeen, 2013.

[Rob96] S. L. Robinson. Trust and Breach of the Psychological Contract. *Administrative Science Quarters*, 41:574–599, 1996.

[Rot67] Julian B. Rotter. A new scale for the measurement of interpersonal trust. *Journal of Personality*, 35(4):651–665, 1967.

[RS11] Mark Ryan and Ben Smyth. The Applied Pi Calculus. In Véronique Cortier and Steve Kremer, editors, *Formal Models and Techniques for Analyzing Security Protocols*, chapter 6. IOS Press, 2011.

[Sak98] Mari Sako. Does Trust Improve Business Performance? In Christel Lane and Reinhard Bachmann, editors, *Trust within and between Organizations*, pages 88–117. Oxford University Press, 1998.

[SAL+07] Angela Sasse, Debi Ashenden, Darren Lawrence, Lizzie Coles-Kemp, Ivan Fléchais, and Paul Kearney. Human Factors Working Group White Paper: Human Vulnerabilities in Security Systems Knowledge Transfer Networks, 2007.

[Sas06] A. Sasse. A Human Factors Approach to Understanding Insider Attacks. (Technical report for QinetiQ), 2006.

[Sha12] Adrian Shaw. TSB Trust Domains Project Work Package 4: Abstract and extended properties and building blocks for Trust Domains. Deliverable D4.2: Design of Components for Trust Domains. Interim Report. Technical report, HP Labs, December 2012.

[SL13] A. Schuchter and M. Levi. The Fraud Triangle Revisited. *Security Journal*, February 2013. (online publication).

[Spe12] P. Speight. *Why Security Fails*. Protection Publications, Osset, West Yorkshire, 2012.

[SPG05] Yogesh L. Simmhan, Beth Plale, and Dennis Gannon. A Survey of Data Provenance in e-Science. *SIGMOD Rec.*, 34(3):31–36, September 2005.

[Sta] The StatVerif Tool. http://markryan.eu/research/statverif/. Last checked 7 March 2014.

[Sut47] E. Sutherland. *Criminology*. Lippincott, Philadelphia, 1947.

[Sut49] E. Sutherland. *White collar crime*. Holt, Reinhart and Winston, New York, 1949.

[Sut10] M. Sutton. Understanding and tackling stolen goods markets. In F. Brookman, M. Maguire, H. Pierpoint, and T.H. Bennett, editors, *Handbook of Crime*. Willan, Uffculme, Devon, 2010.

[SZ05] Ravi Sandhu and Xinwen Zhang. Peer-to-peer Access Control Architecture Using Trusted Computing Technology. In *Proceedings of the Tenth ACM Symposium on Access Control Models and Technologies*, SACMAT '05, pages 147–158, New York, NY, USA, 2005. ACM.

[Tan08] Q. Tang. On using encryption techniques to enhance sticky policies. Technical report, University of Twente, 2008.

[TBT06] Axel Thümmler, Peter Buchholz, and Miklos Telek. A Novel Approach for Phase-Type Fitting with the EM Algorithm. *IEEE Trans. Dependable Secur. Comput.*, 3(3):245–258, 2006.

[TCG] TCG Infrastructure Working Group. Trusted Multi-tenant Infrastructure Specifications. https://www.trustedcomputinggroup.org/developers/trusted_multitenant_infrastructure/specifications. (Retrieved 15 February 2013).

[Tru07] Trusted Computing Group. *TCG. TCG Specification Architecture Overview. Revision 1.4.*, revision 1.4 edition, 2007.

[Tun11] M. Tunley. Need, greed or opportunity? An examination of who commits benefit fraud and why they do it. *Security Journal*, 24(4):302–319, 2011.

[VMW] VMWare. *VMWAre. VMware Virtual Networking Concepts Information*.

[vR] Reinout van Rees. Clarity in the Usage of the Terms Ontology, Taxonomy and Classification. Available at http://www.reinout.vanrees.org/_downloads/2003_cib.pdf (10 March 2014).

[WE05] Y. Wang and H. Emurian. An overview of online trust: Concepts, elements, and implications. *Computers in Human Behavior*, 21:105–125, 2005.

[Wil06] R. Willison. Understanding the perceptions of employee computer crime in the organisational context. *Information and Organisation*, 16:304–324, 2006.

[WS09] R. Willison and M. Siponen. Overcoming the insider: reducing employee computer crime through situational crime prevention. *Communications of the ACM*, 52(9):133–137, 2009.

[Zha10] S. X. Zhang. Talking to Snakeheads: methodological considerations for research on Chinese Human Smuggling. In W Bernasco, editor, *Offenders on Offending*. Willan, Collumpton, 2010.