THE STRUCTURE AND DYNAMICS OF SYSTEMS SECURITY ECONOMICS

ADAM BEAUTEMENT AND DAVID PYM

ABSTRACT. Structured systems security economics provides a conceptual framework, inspired by macroeconomic models of trade-offs and mathematical systems models, for analysing the structure of security architectures, their policy constraints, and their interactions with users. In this paper, we explore the dynamics of structured systems security economics by considering the representation and functionality of Actors in the framework. We show how a simple representation of Actors' preferences is sufficient to understand the security dynamics of the system and support utility calculations that inform design and investment decisions. Overall, the framework is intended to facilitate the design and implementation of trustworthy security systems.

1. INTRODUCTION: THE SECURITY MANAGEMENT PROBLEM

Organizations deploy systems technologies in order to achieve their business objectives. In many situations, it is critical, from the perspectives of the system's users, operators, and regulators, that the system be trustworthy. In order to achieve this, it will, typically, be necessary for an organization to invest in deploying information security policies, processes, and technologies in order to protect the confidentiality, C, integrity, I, and availability, A, of its business processes. Faced with a finite security budget, in terms of finance and other resources [6], managers must choose how to deploy their resources to protect each of these aspects of their operations; that is, they must accept that there are trade-offs to be made. We have argued elsewhere that utility theory — particularly, but not exclusively, as employed in macroeconomics and financial economics — provides a suitable framework for characterizing and exploring such trade-offs [22, 21], building on pioneering work such as Anderson [3] and Gordon and Loeb [18, 19], among others.

A key issue is to understand how systems managers' policies and preferences, be they motivated by regulators or by business objectives, should influence system design. Our approach, as represented in recent work by the present authors and their collaborations with others [9, 26, 5], is to represent policies and preferences in terms of utility functions and to integrate these representations with executable mathematical systems models [13, 14, 11, 12], so as to produce a framework that is able to provide predictive analyses of the consequences of policy choices.

In restricted settings, such as vulnerability and threat management [8, 9, 2] and identity and access management [26, 2], utility-based methodologies are already being deployed as industrial-strength services. Hewlett-Packard's Security Analytics [30, 2] is one such service which makes essential use of mathematical systems models and economic models to analyse security trade-offs and support security policy and investment decision making. While such approaches do make attempts to include human factors this aspect is represented implicitly and informally and without the fidelity afforded to the technological components of the system. Crucially while strong attempts are made to elicit the preferences of key stakeholders the preferences of the users in the system are not represented. However, other areas of research inform us that it is dangerous to ignore these preferences.

Adams and Sasse [1] found that insecure user behaviours, such as choosing simple passwords, were perceived by the users to be caused by a mechanism designed to increase security — in this case, being asked to frequently change their passwords. Understanding such interactions is crucial for developing trustworthy systems. However, it is only recently that such concerns have begun to be addressed by the security community. Pallas [29] stated that 'security management is all about human behaviour'. He goes on to suggest that co-operation between the members of an organization is essential for security. If this co-operation is not elicited, then security will be less effective or even absent. While we may disagree that security management is *all* about human behaviour, the relationship between policies and the behaviour of the entities to which they are intended to apply is certainly a key issue for managers. Beautement and Sasse's Compliance Budget [7] also informs us that if user effort is ignored when implementing security policy weaknesses will be introduced into the system. A first response to such a lack of compliance may

be to introduce more training and sanctions to educate the user population to the nature of the risks but Herley [20] puts forward compelling arguments for this being a poor choice of action. Rather than users being too uneducated or unmotivated, Herley posits that in fact their rejection of security advice is based on a rational cost–benefit analysis and an understanding of the value of their own time.

Taking these ideas forward it becomes clear that any systems model that aims to support decision making or make predictions about the impact of policy changes must include a strong notion of the user and their preferences. In this paper, we introduce an approach that allows both user preferences and policy design to be represented within the same structure, the architecture of which is explicitly designed to facilitate a smooth transition to systems modelling. Bringing together user preferences and policy decisions under a unified framework allows utility to be calculated for the whole system rather than having to consider, for example, usability and policy separately.

In [5], recapitulated in Section 2, we have presented a conceptual framework — structured systems security economics — for describing and designing security architectures. It has the following key features:

- A clean separation between declarative security objectives, such as CIA or sensitivity and criticality, and the operational mechanisms used to achieve them, such as authentication and back-ups;
- A framework that captures how requirements and preferences (expressed in terms of utility) cascade through a hierarchy of rôles to determine a hierarchy of objectives, together with a corresponding hierarchy of associated implementation mechanisms;
- Associated with these hierarchies are Actors, which populate the rôles identified within the hierarchies. Actors were not the primary focus of [5], which was mainly concerned with structural not dynamic issues;
- The whole analysis is organized in such a way that it maps onto a predictive mathematical systems modelling and economic modelling framework [13, 5]. Moreover, this modelling framework is the same as the one developed by one of us and deployed in Hewlett-Packard's Security Analytics service [2, 30].

In his paper, we expand upon the representation and functionality of Actors, so providing a detailed account of the dynamics of structured systems security economics.

In Section 2, we review and recapitulate the structured systems security economics framework introduced in [5], including a sketch of the underlying mathematical systems modelling approach. In Section 3, we explain the representation and functionality of Actors, the key dynamic component of the framework, giving a discussion of Preference Grids, which represent the Actors' preferences for outcomes at given locations and provide a basis for simple utility calculations. Finally, in Section 4, we summarize the overall methodology we have presented, and mention some directions for future work.

2. Framework: Structured Systems Security Economics

We have argued elsewhere [5] that a key distinction within the concepts of information security is between the declarative and the operational, and that it is important not to confuse the two. In addition, we have argued that the concept of utility — from economics — provides an appropriate mechanism for assessing the value of both declarative objectives and operational implementations.

- The *declarative* concepts of information are those qualities that information security policies and architectures seek to achieve. The key declarative concepts — at least at the usual level of abstraction — are confidentiality, integrity, and availability, and different systems will seek to achieve these qualities to varying relative extents.
- The *operational* concepts of information security are the mechanisms that are used in order to implement the declarative objectives. For example, authentication is used to protect confidentiality, and back-ups are used to protect availability.
- *Utility functions* are used to express preferences between different, possibly conflicting objectives, as well as the form and time-evolution of the objectives themselves.

So, the fundamental declarative concepts of information security are confidentiality, integrity, and availability (CIA). Alongside these notions, also declarative, sits the concept of privacy. We are not concerned with privacy in this paper. Trustworthiness in a system and its operations, however, can be seen as a consequence of adequate standards of confidentiality, integrity, and availability.

Although our motivation for this work derives from information security, it seems clear that the concepts and structures that we introduce are applicable not only to information security but also to physical security. Indeed, it seems that many of the concepts that originate in information security are directly applicable to physical security questions.

2.1. **A Running Example.** Drawing upon [5], we use a running example that illustrates the commonality of information and physical security concepts quite clearly. Specifically, we consider the management of an airport's security process [4, 33, 35, 37, 27]. A key aspect of this is checking passengers and their bags for acceptability to fly. A passenger, with luggage, must navigate from the concourse of an airport's terminal building to a seat on an aircraft; that is, it is an access control process that is predicated on maintaining an *integrity* property of aircraft, and this is achieved by maintaining that property for passengers and their baggage. This integrity property trades-off against other concerns, principally costs, incurred in providing security staff and equipment, and service availability.

From the perspective of information security, the declarative objective of airport security is to maintain appropriate integrity properties of the aircraft. Specifically, to ensure that certain classes of objects and individuals are excluded. There are, however, trade-offs with the availability of the aircraft to its users. For example, extremely rigorous security checks might be imposed, with the consequence that some passengers might be unable to arrive at the airport in sufficient time for them to be completed before access to the flight is closed. Moreover, the operational mechanisms used to implement airport security involve both physical components, such as x-ray machines and body searches, informational components, such as the passenger manifest, and objects that are both physical and informational, such as boarding cards and passports.

2.2. **Systems Economics.** Just as in [5], economics is the driving perspective of this paper. The inspiration for our deployment of the methods of utility theory is drawn from the theory of the macroeconomic management of the key indicators in market economies.

For example (e.g., [32]), in the macroeconomic management of market economies, central banks play a key rôle. For example, the managers of a central bank might be given, by their national governments, targets for certain key economic indicators, such as unemployment (u_t) and inflation (π_t) at time t (time can be either discrete or continuous here). Their task is then to set a (e.g., monthly) sequence of controls, such as their base (interest) rates (i_t) so that the key indicators are sufficiently close to their targets, $\overline{u_t}$ and $\overline{\pi_t}$, respectively. Typically, using this example, the managers' policy is expressed as a utility function

$$U_t \quad = \quad w_1 f_1(u_t - \overline{u_t}) + w_2 f_2(\pi_t - \overline{\pi_t}) \tag{1}$$

together with system equations, $u_t = s_1(i_t)$ and $\pi_t = s_2(i_t)$, expressing the dependency (among other things) of u and π on interest rates in terms of functions s_1 and s_2 that describe the (macro) dynamics of the economy. Two key components of this set-up are the following:

- The weights w_1 and w_2 (typically, values between 0 and 1) that express the managers' preference between the components of the utility function — that is, which they care about more; and
- The functions f_1 and f_2 that express how utility depends on deviation from target. A simple version of this set-up would take the f_is to be quadratic. Quadratics conveniently express diminishing marginal returns as the indicators approach target, but make utility symmetric around target. More realistically, Linex functions [36, 38, 32], usually expressed in the form $g(z) = (exp(\alpha z) - \alpha z - 1)/\alpha^2$ are used to capture a degree of asymmetry that is parametrized by α.

The managers' task, then, is to set a sequence of interest rates i_t such that the *expected* utility, $E[U_t]$, remains within an acceptable range, as u_t and π_t vary, and trade-off against each other, as the sequence of rates i_t evolves. In general, there can of course be as many components as required in a utility function.

This framework can be deployed in the context of information security as follows:

- At the declarative level, the components, or qualities, of information security are confidentiality, integrity, and availability;
- The other key component, at the declarative level, is cost;
- Different organizations will typically make different prioritizations between these qualities for different aspects of their operations, giving rise to the weightings in a utility function for a given aspect of a given organization's operations;

- The choice of functional form for each quality's deviation from target will depend on the managers' preferences: for example,
- At the operational level, utility can be used to express not only preferences between different implementation mechanisms for a given quality (e.g., different access control mechanisms to protect confidentiality), but also the impacts of the choice of mechanism for one quality on other declarative objectives; for example, the choice of two-factor authentication to protect confidentiality might cause (possibly unnecessarily) restricted availability in certain circumstances.

To formulate an analysis of this kind, we follow, *mutatis mutandis*, the prescription outlined above:

- The organization that deploys information security measures exists in an economic and/or regulatory environment. This environment places constraints upon the systems and security architectures available to the organization's managers;
- The managers formulate a utility function that expresses their policy preferences, which will depend upon the nature of their organization. For example, state intelligence agencies and online retailers will have quite different priorities among confidentiality, integrity, and availability; see, for example, [22];
- In a highly complex situation, such as a security architecture, it will typically not be possible to formulate system equations (in terms of functions s_1 and s_2) in the way that is usually possible in, for example, macroeconomic modelling. Typically, though, the key control variables, such as system interconnectivity or investment in various aspects (people, process, and technology) of security operations, will be identifiable;
- Instead, however, an executable system model [13], using the key control variables, can used in order to simulate the the dynamics of the utility function. Systems models are discussed herein in Section 2.6.

2.3. **Framework Architecture.** In this section, we review our conceptual account of security architectures [5] that is structured so as to facilitate direct integration both with the natural structure of executable system models (see Section 3; see [13]) and with the deployment of the macroeconomics-inspired approach to the application of utility theory to express and evaluate the preferences of the security stakeholders.

This section recapitulates a similar section in [5]; its inclusion here, making this paper sufficiently self-contained, is necessary to provide the basic structures with which Actors, the focus of this paper, interact. Throughout this section we use airport security as our running example, chosen to illustrate not only the basic concepts of our approach, but also the commonality of the analysis of information and physical security. The elucidation of the commonality of concepts and analysis for information security and physical security, and the applicability of our approach in both contexts as well as hybrid contexts (such as airport security) is one of the objectives of this work.

There are two key layers in our representation of a security system, the *Framework* layer and the *Instantiation* layer. There is a commonality of organization between these layers although they represent conceptually different parts of the model. Both layers are organized into a hierarchy of rôles with each rôle sub-divided into dependencies, priorities, and preferences.

The hierarchy contains all the relevant rôles that make up the organization being modelled. Rôles are ordered by their ability to influence the security architecture of the system. In other words, they are classified by the toolbox that is available to them for modifying *Security Objects* (that characterize security tasks, defined below). The system accepts multiple and partial orderings. For example, the top level of the model might represent the strategic decision-makers of the organization, such as an airport's security managers or their regulators, while the bottom level might represent an individual employee or user of the organization, such as an airport's check-in staff or a passenger navigating airport security. The rôles represent the possible positions individuals can adopt in the hierarchy. They do not represent any entity themselves. They are instead populated by *Actors*. Actors are key components in the architecture and, building on [5], are the main focus of this paper. They are discussed in some detail in Section 3.

Each hierarchy level contains three sections representing the dependencies, priorities and preferences of that level. For our purposes we define the terms as follows:

- *Dependencies* (strong requirement): Externally enforced requirements that actors populating the rôle must meet all of in order to function within the model. Actors occupying this rôle have no

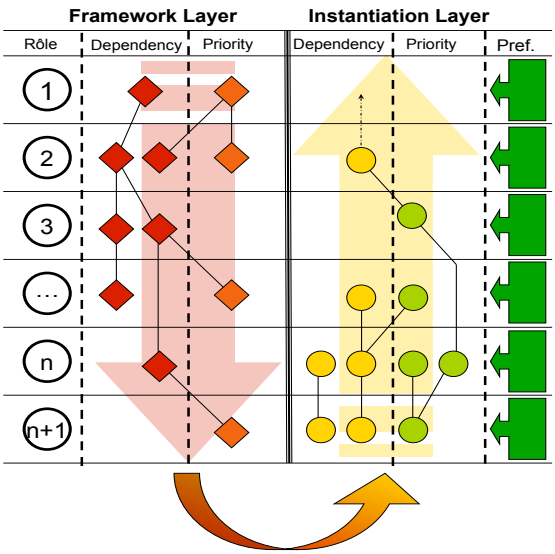
choice in whether or not (and possibly even how) to meet these requirements regardless of how resource inefficient they are. Dependencies will often be informed by the environment within which the hierarchy exists;

- *Priorities* (weak requirement): Externally supplied tasks, as many as possible of which should be met by Actors in the associated rôle. Actors have some choice in which priorities to meet and how they are approached. In a limited resource environment, Actors can select the most resource efficient priorities and methods first. Priorities will often be informed by the rôle that the level represents;
- *Preferences*: Actor-generated tasks that the Actor has decided are worth achieving from its own perspective. These can be generated by the Actor's inclusion in other hierarchies.

Dependencies, priorities, and preferences (DPP) and the hierarchy of rôles structure are found in both the framework layer and the instantiation layer.

The form and construction of the security architecture is illustrated in Figure 1. This architecture is grounded in our underlying approach to mathematical systems modelling [13, 11, 14], described in Section 2.6, which can be captured (sufficiently adequately for our present purposes) in the Core Gnosis modelling tool [13, 17]. Some details of the representation of the airport architecture are given in [5]; we elide the implementation details here.

FIGURE 1. Security Architecture: Framework and Instantiation



The key components of this diagram are the following:

- *The hierarchy of rôles* (far left). Rôles capture the relevant security management structure of the organization being modelled. Rôles are ordered (in the hierarchy) by their ability to influence the security architecture of the system;
- *The Framework Layer* (centre left). The Framework Layer is constructed top-down. Dependencies and priorities at a given level in the hierarchy induce dependencies and priorities at lower levels;
- *Security Objects* (trees within Framework Layer). Security Objects represent the security tasks which, if completed, will satisfy the dependencies and priorities with which they are associated;
- *The Instantiation Layer* (centre right). The Instantiation Layer is constructed bottom-up, starting where the Framework Layer finishes (see below). The Instantiation Layer is a populated image of the Framework Layer;
- *Security Components* (nodes of trees within Instantiation Layer). Security Components perform the operational checks required in order to deliver Security Objects. They do so by returning

boolean values up the tree, towards the root. They enter the architecture when the Framework is instantiated;

- *Actors* (far right). Actors occupy rôles. They insert preferences into the hierarchy of rôles at the Instantiation Layer. Again, Actors are the main focus of this paper and are discussed in Section 3.

A key point here concerns the way in which the dependencies, priorities, and preferences in a model are intimately related to the declarative security concepts. For example, given confidentiality, integrity, and availability as the concerns, the dependencies, priorities — in the Framework Layer — express the policies required to implement the managers' preferences as represent in their chosen utility function for the organization. Specifically, given a utility function with definiens of the form

$$U(C, I, A, K) = w_1 f_1(C - \overline{C}) + w_2 f_2(I - \overline{I}) + w_3 f_3(A - \overline{A}) + w_4 f_4(K - \overline{K}) \tag{2}$$

where C, I, and A denote suitable instances and/or measures (see, for example, [22]) of confidentiality. integrity, and availability, respectively, and where K denotes cost or investment, dependencies will have very high weightings, with very little tolerance for deviation from target, and priorities slightly lower weightings. Preferences (see below) have weightings that are lower still, and may have high tolerance for deviation from target.

The form and function of the Framework and Instantiation Layers, and the interaction between them, will now be discussed in more detail.

2.4. **The Framework Layer.** The Framework Layer represents the underlying structure of the system. It is static (in the sense that the model does not run on this layer) and declarative but informs the construction of the more operational Instantiation Layer. A completed Framework Layer consists in a hierarchy of rôles (see, for example, [4]) with dependencies and priorities assigned to them. As preferences are derived from actors (see below), they do not explicitly appear in this layer because actors appear in the Instantiation Layer. However an implicit notion of the preferences of the system is found in the actions that a perfectly compliant actor would take. This forms the neutral prefernence state of the system, a concept we shall return to later. The dependencies and priorities will each have a *Security Object* (SO) assigned to them. SOs are a unique component of the Framework Layer and represent the security tasks which, if completed, will satisfy the dependencies and priorities with which they are associated.

In the setting of the running example of airport security that we have begun to introduce, examples of Security Objects include the examining of checked luggage, the checking of hand luggage and passengers — to identify and so remove any prohibited contents — and the tracking of the relationship between passengers and checked luggage. These examples are developed below.

SOs are unconstrained with respect to their location within the hierarchy. SOs can only exist in one hierarchy and never populate multiple hierarchies. This is a key difference between Actors and SOs. One of the aims of this formulation is to improve the communication between different stakeholders and eliminate the duplication caused by the failure to understand the connectedness of security concepts between levels. In practice, that means a typical SO will exist at multiple levels and multiple sections (dependency, priority) in the Framework. It will commonly be the case that an SO created at a higher level will transition through and connect (or create) priorities and dependencies lower in the framework.

For the more mathematically minded reader, there are many choices of formalization of SO. Our working choice for the purposes of this paper is, roughly speaking, the following:

- SOs are characterized by (directed) and/or forests[1] [2] (illustrated in Figure 1 by the red/orange trees in the Framework Layer) associated with dependencies and priorities;
- Internal nodes of the trees are labelled with boolean variables, each associated with a dependency or priority, and truth conditions are inherited upwards (towards the root);
- Leaves are nodes for which a boolean instantiation (all components for conjunctions, one component for disjunctions) can be determined at the next level down in the hierarchy of rôles.

SOs do not map cleanly onto any part of a systems model (see Section 2.6) as such a simulation draws more naturally from the Instantiation Layer in which the security requirements are populated with concrete processes and resources.

[1] A (directed) forest is a disjoint union of (directed) and/or trees.

[2] A forest is required because a given SO may, in general, derive from more than one dependency or priority.

The Framework is populated with dependencies, priorities, and security objects through an iterative process that requires design input from an expert source. The criteria under which security objects terminate and by which the Framework can be said to be complete is the same for all frameworks created in this way.

As indicated above, dependencies and priorities are externally generated. In practice, a hierarchy of rôles will not encompass all possible contributors to the framework and will have been bounded at some sensible level. In our example, we have not represented any rôle higher than the airport security manager in our hierarchy. The creation and bounding of the hierarchy of rôles is the first step in creating a framework layer. To populate a framework, it is necessary to determine the dependencies and/or priorities that the top rôle in the hierarchy will inherit from sources external to the hierarchy. At this stage, we will have a complete hierarchy of rôles that is empty of dependencies and priorities except for the top layer. The next step is to assign security objects to these dependencies and priorities that will allow them to be fulfilled; see Table 1.

TABLE 1. SOs Stage 1

Rôle	**Dependencies**	**Security Objects**
Airport Security Manager	Ensure no prohibited materials transit the airport	Scan checked luggage Scan hand luggage and passengers Track relationship between passengers and checked luggage

At this point the iterative process begins. The construction of a framework always proceeds from top to bottom. At each iteration the following are checked:

- Are there any dependencies or priorities without an assigned Security Object?
- Are there any unterminated Security Objects?

The construction of a Security Object terminates once it is possible to return a boolean value from its lowest point. If this is not possible, then the Security Object must be extended to the rôle below in the hierarchy, creating any necessary dependencies and priorities as it does so. The dependencies and priorities created will be informed by the rôle creating them; thus, as the SO descends through the hierarchy, it will become more detailed as the scope of the lower levels is necessarily more limited. In our example above, none of the security objects can return a value and thus need to be extended. Let us extend the 'scan hand luggage' SO. The following (Table 2) would be the result of two iterations, one would generate the dependency from the SO above, the second would find a dependency without an assigned SO and create one:

TABLE 2. SOs Stage 2

Rôle	**Dependencies**	**Security Objects**
Airport Security Manager	Ensure no prohibited materials transit the airport	Scan checked luggage Scan hand luggage and passengers Track relationship between passengers and checked luggage
Airport Security Staff	Examine all passengers and luggage passing through security checkpoint	Identify contents of hand luggage and verify permitted

Note, for example, that the SO 'scan and luggage and passengers' corresponds to a tree (red/orange in Figure 1) in the Framework Layer.

At this point the SO can return a boolean (true/false that the contents of the bag are permitted) and will terminate. The framework is not yet complete, however, as there are still unterminated SOs at the manager layer. Iterating in this fashion would also close those at a suitable point. The final step in a SO is always a compliance step which indicates that at this level and below the rôles in the hierarchy simply comply with the SO and are not involved in its execution. This would add the following line (Table 3) to the framework:

Once the framework is complete under the criteria outlined above the SO's will form a forest with the leaves connecting each dependency and priority in the framework. At this point we can begin to construct the instantiation layer.

TABLE 3. SOs Stage 3

Rôle	Dependencies	Security Objects
Passenger	Comply with SO	

2.5. **The Instantiation Layer.** Whereas the framework layer is static and declarative the instantiation layer is dynamic and operational. Two new parts of the architecture are added during instantiation, Security Components (SC) and Actors. Actors will be discussed in more detail below; for now it is sufficient to know that they occupy rôles and insert preferences into the hierarchy of rôles at the Instantiation Layer. Security components combine together to form the operational counterparts of security objects.

The instantiation layer needs building in the same way that the framework layer did. Again an iterative process is adopted with certain termination criteria. The key difference here is that this layer is built bottom up. SCs lay out the processes and resources needed to perform the boolean checks specified in corresponding SOs. SCs start at the final 'compliance' layer of the SO. Once the processes and resources required at this level are put in place we check to see if they are sufficient to complete the SO. If yes, then the SC terminates. If not then we move up to the rôle above and add additional processes and resources as needed. Again, this process repeats until all SCs are closed. At this point, the Instantiation layer is complete.

A little more formally, corresponding to the slightly more formal view of SOs sketched above, we can describe how SCs are combined to instantiate SOs as follows:

- SCs are combined according to the and/or forest determined by the SO that they instantiate;
- Each SC implements a checking process that applies to Actors at the level below;
- SCs return boolean values that instantiate internal nodes of the corresponding SO.

Working through our example again we start at the passenger level and work upward until we have sufficient processes and resources in place to return a boolean for the statement 'the passenger's possessions and luggage are permitted'. The finished security component in this case would be as follows (Table 4):

TABLE 4. SCs

Rôle	Dependencies	Security Components
Airport Security Manager	Ensure no prohibited materials transit the airport	Provide resources (X-ray machine, metal detector, wands) Provide data on prohibited materials for X-ray comparison
Airport Security Staff	Examine all passengers and luggage passing through security checkpoint	Monitor X-ray machine and inspect results for prohibited items Hand-search suspect luggage Hand-scan suspicious passengers
Passenger	Comply with SO	Place luggage on scanner Walk through detector

In Figure 1, the SCs correspond to the green/yellow nodes in the Instantiation Layer

Note that whereas the SO terminated in a 'compliance' level the SC terminates at a 'provision' level when it reaches a rôle that can sufficiently provide the resources required to execute the SC without recourse to a higher rôle.

In that SCs deal specifically with the processes and resources needed to satisfy the security requirements laid out by the SOs, they strongly reflect the systems modelling approach (see Section 2.6, next) which underlies and motivates this work and their construction doubles as the first step in the modelling process.

The final component of the instantiation layer (and the model) is the Actors, considered in detail Section 3.

2.6. **Mathematical Systems Modelling.** It is useful to argue — see, for example, [13, 14, 11, 5] — that the key structural aspects of systems are the ones discussed below: environment, location, resource, and

process. This point of view is consistent with the classical view of distributed systems, as described, for example, in [15].

More detail of the necessary mathematics is given in an accessible style in [5, 13]. The approach we take is rooted in process algebra and its developments (see, for example, [23, 24, 25]), associated (modal) logics (see, for example, [24, 34]), and substructural logic (see, for example, [28, 31, 14, 11], and references therein). Related work in systems modelling — including work by the groups of Hillston (PEPA), Kwiatkowska (PRISM), and Saunders (Möbius) — is discussed in [13, 5].

Environment. Systems exist within external environments, from which events are incident upon the system's boundaries. Typically, the environment is insufficiently understood and too complex to be represented in the same, explicit, form as the system itself.

For example, in the airport example, the terminal building may be located at the focus of a complex road and rail network that delivers passengers to the airport. However, we might not wish to model the traffic flows in the network in any detail, preferring just to give a stochastic representation of passenger arrivals. Parts of the the internal structure of the system that are not of interest to the modeller can also be treated as part of the environment.

Mathematically, the environment within which a system exists is modelled using stochastic processes to represent the incidence of events at the system's boundary. For example, the the arrival of passengers at the airport's terminal from the outside world might be captured as Poisson process with arrival rate determined by a negative exponential distribution.

Location. In general, the architectures of systems are highly distributed, logically and/or physically. The system's resources are distributed around a collection of places, and these places have (directed) connections between them. The airport locations that are relevant to this paper are illustrated in Figure 2.

Mathematically, location can be captured by structures that are similar to graphs, or hyper-graphs, or topological spaces [11, 13]. A simple leading example is provided by directed graphs.

Resource. The infrastructure of a system, on which the system's processes execute, consists of a collection of resources that may be utilized by the processes in order to achieve their intended purposes. For example, in the airport, resources might include security x-ray machines, check-in desks, and boarding cards.

Mathematically, resources are modelled as pre-ordered monoids satisfying a functoriality condition. This amounts to capturing, with minimal mathematical structure, that resource elements can be combined and compared. A basic example is provided by the non-negative integers, combined using addition (with unit zero) and compared using less-than-or-equals.

Process. A synthetic system exists in order to deliver services, and services can be conveniently understood as processes that execute on the system's architecture. Typically, it will be necessary for many services to execute, concurrently and sequentially, in order for a service to be delivered. Similarly, natural processes execute relative to natural substrates. Our main focus here is on synthetic systems; in particular, large-scale information systems. For example, in the airport, the hand luggage scanning procedure is a process which takes from its environment arrivals of cabin baggage, treated as resources, and applies an x-ray examination to the resource. SCs will typically be modelled as processes.

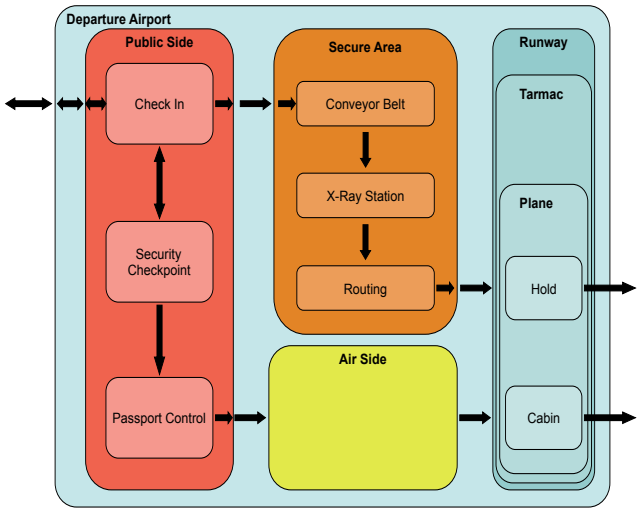
Mathematically, processes are modelled using a process algebra that incorporates, again using minimal mathematical structure, the notions of location and resource described above. Processes are generated by basic actions, and can be combined using combinators such as non-deterministic choice and concurrent composition, and can be defined recursively.

These components provide the semantic basis for a modelling tool — Core Gnosis [16]; *Gnosis, Gk. knowledge* — that incorporates the stochastic model of environment described above. Essentially, the execution of processes relative to resources at locations is initiated by stochastic events that cause actions to occur. Again, see [13] for a summary of the approach. Core Gnosis provides a practical modelling tool, in the spirit of Demos [10], that has been deployed in a range of industrial-strength settings [8, 9]. Both the mathematical framework and the use of the tool to represent the airport example are described in more detail in [5].

Having established, at least conceptually, the the structural framework for security architectures, the economic framework within which preferences are expressed, and the mapping to a mathematical systems modelling framework, we are now ready to provide an account of the the dynamics of security architectures.

The key step is to study in detail how actors — driven by their objectives and preferences — interact with the structure.

FIGURE 2. Airport Locations



Modelling Actors is a quite delicate matter, and depends on the perspective and purpose of the model:

- Actors instantiate rôles (in the Instantiation Layer). An instantiated rôle may occur at several different locations;
- Actors can be modelled naturally either as processes or resources, depending upon the perspective from which the system model is constructed. We shall return to this point in Section 3.

3. DYNAMICS: ACTORS AND ACTIONS

In our previous paper, we outlined the rôle of Actors as the primary element in adding a dynamic component to the hierarchy of rôles conceptualisation but the details of their function were not adequately expressed. A key objective of returning to this work is to extend the previous definition of Actors and expand upon their function within the framework.

3.1. **Actors.** Actors are the final component of the instantiation layer (and the model). Uniquely Actors exist independently from any single security hierarchy. They represent entities that transition between hierarchies, this being the key difference between Actors and SOs. They can interact with any and all hierarchies present, simultaneously if necessary. Actors exist solely as a collection of tags corresponding to their attributes. When an Actor interacts with a hierarchy and seeks to populate one of its rôles, the hierarchy examines the Actors tags (some or all of which may be unreadable by the hierarchy). It assigns the Actor to a rôle based on its tags and clones a copy of that rôle for the Actor to inhabit for the duration of its lifecycle in that system. This clone inherits the relevant dependencies and priorities of the rôle. Its preferences are obtained by interrogating the Actors tag-cloud. These preferences are used to generate Preference Grids which are covered in more detail below.

Some Actors will have no preferences. Such Actors represent inanimate physical components of the system (e.g., a passenger's bags in the airport example) or data stores (e.g., a baggage handling label in the airport example). They can be passed between hierarchies (the departure and destination airports will have separate security systems and this will be considered separate hierarchies) but have no intentions of their own. Once a clone is created the Actor no long directly interacts with the hierarchy for the duration of its lifecycle. The Actor clone is now a full part of the hierarchy and can interact with its associated Security Components, updating its dependencies, priorities, and preferences as necessary. The associated Preference Grids will determine the paths and states of the hierarchy it passes through. These in turn decided the exit

point of the actor clone from the hierarchy. Each possible access point has a set of tags associated with it that are then passed back to the core Actor to replace the set used to interact with the hierarchy. These may be identical or contain new or changed statuses. For example, a hierarchy designed to check the citizenship of an unknown actor may return tags, which it did not previously possess, identifying it as a national or an illegal alien. The introduction of Actors adds a third dimension to the framework. Multiple actors can exist in any one layer of the hierarchy (and, typically, the lower levels will have more actors assigned to them). This also means that Security Objects can span multiple actors as well as multiple levels.

To understand the governing mechanisms that inform the behaviour of Actors it is necessary to recall certain key points from the hierarchy conceptualisation. First, that the lowest level in the hierarchy of rôles is occupied by Zero Preference Actors (ZPAs). This level is essentially a (possibly extensive) list of the ZPAs allowed within, or understood by, the security hierarchy. In the case of the airport example this list would for example include

- hand luggage,
- checked luggage,
- passport,
- ticket/booking documentation,
- minor offensive items (such as liquids over the amount allowed in hand luggage),
- major offensive items (such as weapons and narcotics).

Second, that the hierarchy of rôles is ordered by the level of control each rôle has over the security system. This means that rôles further up the hierarchy have authority over the lower levels.

Returning to the question of how to model Actors, the following is a useful guide:

- A model that is written from the point of view of a given Actor (e.g., passenger's view of his or her passage through the airport) would normally treat that Actor as a process.
- Other Actors (e.g., for example the airport security staff, as part of a model written from the point of view of a passenger) would normally be treated as resources — to be used by the subject, or processes called by the subject, in order to navigate the system.

3.2. **Preference Grids.** Actors interact with one other, and with (other) resources in the system as they navigate a system. From the perspective of a system model that reflects, for example, the point of view of a passenger, navigating a security checkpoint is

Within the hierarchy, then, Actors will often have cause to engage and interact with one another. Both to motivate and to mediate such interactions and determine their outcomes, it is necessary to represent within the system a notion of the intentions of the various Actors. This is achieved by using Preference Grids. As well as their tag clouds Actors have a set of Grids that relate to themselves and all Actors in the layers of the hierarchy below their own, including ZPAs. These Grids are tables containing a set of Preference Values representing the intentions of that Actor toward the other dynamic components of the security hierarchy. Grids are location-dependent as the preferences of the Actors toward each other will vary as they move through the system. Referring to the airport location map in Figure 2, it is clear that a passenger will react very differently to a security guard attempting to search their bag at the check-in desk as opposed to the security checkpoint. This location map will be used as the basis for discussing the relationship between Actors and locations in the following section. Grids are, evidently, close related to utility, and we will return to this point in Section 3.3.

The Grids assign weightings to 'C, I, A', expressed as a value between -1 and 1. The CIA dimension can be extended as necessary with other components such as actions. Each Grid — for example, one expressing the intentions of a passenger toward hand luggage — is in fact a family of Grids representing the intentions of the passenger toward all instantiations of 'hand luggage' they have interacted with in the current state of the system. Where the Actor has an existing preference toward the Actor or ZPA in question these values will be used to populate the Grid. However, as not all interactions can be predicted in advance (and indeed to do so would bloat the information content of the Actor's tag cloud significantly) interactions can occur in which the Actor has no preference to draw upon. In such cases, a 'neutral' Grid will be generated from information contained in the Framework Layer. Neutral in this case does not mean containing values of 0 but containing values aligned with the goals of the security hierarchy rather than the individual Actor. In this sense, where the Actor has no preferences of its own, it inherits the preferences of the system.

Although it will be necessary to generate many of these Grids in a typical instantiation of the system, they are computationally light to handle and thus incur little overhead. These Grids, and the attitudes they represent, form the intention map for each actor. Do they want explosives to be in the passenger lounge or not? How concerned are they with the confidentiality of the contents of their luggage? As Actors can also have preferences regarding actors lower down the hierarchy and not just ZPAs we allow interaction between groups of intentioned actors as well. For example, security guards may detain or search passengers according to their preferences. This allows us to capture the intentions of passengers, smugglers, terrorists and security personnel with equally fidelity using the same approach. The system can remain agnostic of the goals of each Actor and does not require special mechanisms to deal with 'enemy' Actors. This allows us to create different types of actor very easily by configuring their tag clouds to contain different preferences regarding the components of the system.

To illustrate the flexibility of the Grid system we will pick up our running airport example and compare the Preference Grids of a passenger attempting to smuggle a disassembled ceramic firearm onto a plane and the airport staff present at the various locations the passenger passes through. Although a full system would require many more Actors and ZPAs to be covered by the Grids for simplicity and space we will focus on the passenger and his hand luggage only. Additionally, we will assume that once the weapon is airside a security breach has occurred so the relevant locations for this example are the check-in desk, security checkpoint and passport control.

At check-in, the passenger will not tolerate anyone examining his or her bag, and the check-in staff will have no interest in it at that stage, so their grids will be aligned. Moreover, the check-in staff, being concerned with the business processes associated with the airport, will actively want the passenger to be there. At this stage both parties are interested in the availability of the passenger and neither intend for the integrity or confidentiality of the bag to be breached. The Grids associated with check-in are shown below. The first column shows the factors to which the preference values are being assigned, in this case CIA. The column headings left to right show the members of the hierarchy of rôles about which the Actor in question has preferences. ZPAs, being the lowest level of the hierarchy, will therefore be in the rightmost columns of a Grid. It is important to understand that the Preference Values contained in the Grid represent those of an Actor about the entities contained in the Grid column headings. Thus the first grid below shows us the preferences of the passenger in our example towards the CIA state of himself and his hand luggage while located at the check-in desk. The values chosen here are intended only to be indicative. In practice, identifying values would require a preference elicitation process (there is an extensive literature; see [9]).

TABLE 5. Passenger (left) and Staff (right) Grids at Check-in

	Passenger	Hand Luggage
C	0.8	1
I	1	1
A	1	1

	Staff	Passenger	Hand Luggage
C	1	0.8	1
I	1	1	1
A	1	1	1

The slight drop in the confidentiality of the passenger is caused by the requirement of the staff to ask them questions during check-in and the willingness of the passenger to respond.

Significant changes occur when the passenger reaches the security checkpoint. Now the passenger must be willing to subject both himself and his bag to scrutiny, necessarily contravening their confidentiality to some extent. However, while our passenger does not mind his or her bag being x-rayed, relying on the ceramic components of the weapon to shield it from detection, he or she would be unwilling to have his or her bag manually searched. This is reflected by his preferred level of confidentiality being higher than that selected by the security guard doing the examination. The Grids for the passenger, x-ray operator, and bag searcher are given in Table 6.

The lowered scoring for the passenger's preference toward his own confidentiality implies that he is willing to be searched during the process. His attitude toward his bag is in alignment with the requirement of the x-ray operator but his preferences conflict with the bag searcher who wishes to reduce the confidentiality of his bag below the level he is comfortable with. The neutral weightings of the bag searcher and x-ray operator indicate that while they are not actively seeking to take items from the bag they have no interest in preserving its contents either. Their actions in this regard will be dictated by other factors

TABLE 6. Passenger (left), X-ray Operator (centre), and Bag Searcher (right) Grids at Security Checkpoint

	Passenger	Hand Luggage
C	0.5	0.6
I	1	1
A	1	1

	Staff	Passenger	Hand Luggage
C	1	0	0.6
I	1	0	0.6
A	1	-1	1

	Staff	Passenger	Hand Luggage
C	1	0	0.4
I	1	1	0
A	1	1	1

such as the legality of its contents. It is also interesting to note that the Grids allow us to express less directly relevant security concerns such as the x-ray operator not wishing to have passengers in or around the machine as shown by the negative weighting for passenger availability.

Whether or not this passenger's bag is actually searched will depend on the degree of suspicion aroused by the x-ray scan combined with the need of the security checkpoint to maintain a certain level of throughput. Assuming the passenger makes it through the security checkpoint successfully, he or she will then (following the map in Figure 2) enter a final passport check before they are allowed airside. Now they are through the security check their Grids will again change to reflect a preference for less intrusive inspections of themselves and their luggage as shown below.

TABLE 7. Passenger (left) and Staff (right) Grids at Passport Control

	Passenger	Hand Luggage
C	0.9	1
I	1	1
A	1	1

	Staff	Passenger	Hand Luggage
C	1	0.9	1
I	1	1	1
A	1	1	1

This situation now mirrors that found at check-in: the passenger is willing to answer some questions (although is expecting less this time round) but is again highly reluctant to allow searches of his bag. The Grids can be used in this way to express a wide variety of behaviours simulating many types of Actor at the same level in the hierarchy. Intra-Grid interactions also reveal behavioural preferences. For example if a passenger has higher integrity than availability at check-in it means he is unwilling to remove items from his bag (if it is overweight for example) even it if means they will not be allowed to check in.

Preference Grids also have a key early-stage rôle in systems models. The preferences expressed in the Grids are directly analogous to preferences for certain processes to be executed or resources allocated. By supplying these requests for resources and processes, Actors provide the dynamic force that drives the system model forward, and are usually modelled in Core Gnosis as processes, though occasionally it is convenient to treat them as resources.

3.3. **Utility Revisited.** The utility functions described in Sections 2.2 and 2.3 are quite complex mathematical constructions, and it may not always, for a given system, be possible to formulate them. Indeed, it may not be necessary.

A measure of an Actor's utility can be calculated using Preference Grids. By comparing the outcome of any action with the initial intention of the Actor, as expressed in its Grids, a value for how well the Actor's preferences have been met can be found. These values will be weighted by how strongly the Actor preferred the course of action in question. These utility values will also serve as a measure of the friction between the primary process of the system (as represented by the intentions of the populating Actors) and the security process. In the case of smugglers and terrorists, utility can be calculated in the same way, but resultants will measure how effective their organizations are at defeating airport security.

In the airport example, we can see Preference Values as crude representations of utility, encapsulating both weighting and deviation from target. For example, in the situations discussed above, we have

$$U(C, I, A) = P^*_C + P^*_I + P^*_A \tag{3}$$

Here, each of P^*_C, P^*_I, and P^*_A is a deviation from a target Preference Value, P_C, P_I, and P_A, each of which represents an anticipated outcome. These can be compared to the actual outcomes that occur, and so the deviations from targets, P^*_C, P^*_I, and P^*_A, determined. Moreover, the size of the initial value can

be interpreted as a crude measure of the weighting assigned by the Actor, and we could enrich the utility expression by introducing functions (i.e., f_is) of the components, as in Equation 2.

4. Discussion: The Value of the Approach and Outstanding Challenges

By introducing Actor's preferences into the same unified structure as management preferences, we allow utility calculations to take place that capture the full range of security-related needs within the system. While management preferences inform the shape of the Framework Layer, user preferences are represented in the Instantiation Layer through the use of Preference Grids. As previously mentioned, a 'neutral' Grid is one in which the Actor has no preferences, and is derived directly from the preferences contained in the Framework Layer. In this way, the Preference Grid system facilitates the interaction of both preference sets and forms the basis for unified utility calculations. Additionally, as a foundation for these calculations, it offers significant flexibility. By choosing either Grids representing the policy preferences or an individual Actor's Grids as the target state, it is possible to calculate the utility with respect to either the policy design or the goals of the Actor chosen. By examining the level of conflict between policy preferences and Actor Grids, a measure of the friction between the security and business processes can also be found.

The motivating force behind this work is not to create a purely theoretical framework but to work toward the development of a toolset that can effectively support decision making, initially in the field of security, although the applicability of this approach does not necessarily end there. As in the Security Analytics service [30, 2], we aim to use systems models for simulation and prediction as a key component of this toolset. With major sections of the framework already expressed in terms of location, process, and resources, the creation of such models is facilitated by the architecture of the system. Data constructs within the framework such as Security Objects and Components can be exported with relative ease into a systems model ensuring that the expert knowledge used in their creation is preserved in the model. As previously discussed both Actor and policy preferences are represented in the framework ensuring that any model, and resulting simulation, deriving from it will be both accurate and grounded in real-world considerations. Through simulation an understanding of the trade-offs between factors in the decision-making process can be reached. With the Grid system mediating the interaction between the dynamic components of the system, this approach ensures that user preferences are given the appropriate weight when such trade-offs are being considered. From this model, predictions can be made about the effectiveness or impact of proposed policy changes allowing a security decision maker to explore safely options at low cost without the requirement to implement or deploy technologies on a trial basis. Additionally, such models can be generated during the design phase of a new system to explore the implementation options available. Again, the inclusion of a more detailed notion of Actor preferences can offer increased accuracy when attempting to predict the behaviour of a new system when it 'goes live' and interacts with a large population of users, including those with harmful intentions, such as terrorists or malicious insiders.

By iterating this process, in the established method of mathematical modelling, refinements can be made to the model, reducing discrepancies between output and observed data, and also increasing the accuracy of the model's predictions. As a refinement to the toolset currently being deployed through Security Analytics, our approach unifies the previously disparate preference sets of users and security managers.

Future work includes the following: multiple and interacting hierarchies (as discussed in [5]); develop a systematic account of system models for hierarchies; work with partners in industry to develop case studies such as power generation and distribution systems, banking systems, and other (than air) transport systems; creation of templates for Core Gnosis models for classes of examples.

5. Acknowledgements

We are grateful to our colleagues in HP's Cloud and Security Lab and in the Trust Economics project, partially funded by the UK's Technology Strategy Board.

References

[1] A. Adams and M.A. Sasse. Users are not the enemy: Why users compromise security mechanisms and how to take remedial measures. In *Communications of the ACM, 42 (12)*, pages 40–46, 1999.

[2] Security Analytics. `http://h20195.www2.hp.com/V2/GetPDF.aspx/4AA3-2046EEW.pdf`.

[3] R. Anderson. Why information security is hard: An economic perspective. In *Proc. 17th Annual Computer Security Applications Conference*, pages 358–265. IEEE, 2001.

[4] N. Ashford, H.P.M. Stanton, and C.A. Moore. *Airport Operations*. McGraw-Hill, 1997.
[5] A. Beautement and D. Pym. Structured systems economics for security management. In Tyler Moore, editor, *Proceedings of the Ninth Workshop on the Economics of Information Security (WEIS)*, 2010. Available at: `http://weis2010.econinfosec.org/papers/session6/weis2010_beautement.pdf`.
[6] A. Beautement and M. A. Sasse. The compliance budget: The economics of user effort in information security. *Computer Fraud and Security*, 2009:8–12, 2009.
[7] A. Beautement, M.A. Sasse, and M. Wonham. The compliance budget: managing security behaviour in organisations. In *Proceedings of the 2008 Workshop on New Security Paradigms*, pages 47–58. ACM Digital Library, 2009.
[8] Y. Beres, J. Griffin, S. Shiu, M. Heitman, D. Markle, and P. Ventura. Analysing the performance of security solutions to reduce vulnerability exposure window. In *Proceedings of the 2008 Annual Computer Security Applications Conference*, pages 33–42. IEEE Computer Society Conference Publishing Services (CPS), 2008.
[9] Y. Beres, D. Pym, and S. Shiu. Decision support for systems security investment. In *Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP*, pages 118–125, 2010. doi: 10.1109/NOMSW.2010.5486590, ISBN: 978-1-4244-6037-3, INSPEC Accession Number: 11502735.
[10] G.M. Birtwistle. *Discrete Event Modelling on SIMULA*. Springer-Verlag, 1987.
[11] M. Collinson, B. Monahan, and D. Pym. A logical and computational theory of located resource. *Journal of Logic and Computation*, 19(6):1207–1244, 2009. doi:10.1093/logcom/exp021.
[12] M. Collinson, B. Monahan, and D. Pym. A discipline of mathematical systems modelling. To appear: College Publications, London, 2010.
[13] M. Collinson, B. Monahan, and D. Pym. Semantics for structured systems modelling and simulation. In *Proc. Simutools 2010*. ICST: ACM Digital Library and EU Digital Library, 2010. ISBN: 78-963-9799-87-5.
[14] 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.
[15] G. Coulouris, J. Dollimore, and T. Kindberg. *Distributed Systems: Concepts and Design*. Addison Wesley, 2005.
[16] Gnosis. `http://www.hpl.hp.com/research/systems_security/gnosis.html`.
[17] Core Gnosis. `http://www.hpl.hp.com/research/systems_security/gnosis.html`.
[18] L.A. Gordon and M.P. Loeb. The Economics of Information Security Investment. *ACM Transactions on Information and Systems Security*, 5(4):438–457, 2002.
[19] L.A. Gordon and M.P. Loeb. *Managing Cybersecurity Resources: A Cost-Benefit Analysis*. McGraw Hill, 2006.
[20] C. Herley. So long and no thanks for the externalities: The rational rejection of security advice by users. In *New Security Paradigms Workshop*, 2009.
[21] C. Ioannidis, D. Pym, and J. Williams. Information security trade-offs and optimal patching policies. Manuscript, 2009.
[22] C. Ioannidis, D. Pym, and J. Williams. Investments and trade-offs in the economics of information security. In Roger Dingledine and Philippe Golle, editors, *Proc. Financial Cryptography and Data Security '09*, volume 5628 of *LNCS*, pages 148–166. Springer, 2009. Preprint available at `http://www.cs.bath.ac.uk/~pym/IoannidisPymWilliams-FC09.pdf`.
[23] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25(3):267–310, 1983.
[24] R. Milner. *Communication and Concurrency*. Prentice Hall, New York, 1989.
[25] R. Milner. *The Space and Motion of Communicating Agents*. Cambridge University Press, 2009.
[26] M. Casassa Mont, Y. Beres, D. Pym, and S. Shiu. Economics of identity and access management: Providing decision support for investments. In *Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP*, pages 134–141, 2010. doi: 10.1109/NOMSW.2010.5486588, ISBN: 978-1-4244-6037-3, INSPEC Accession Number: 11502733.
[27] R. De Neufville and A.R. Odoni. *Airport systems: planning, design, and management*. McGraw-Hill Professional, 2003.
[28] P.W. O'Hearn and D.J. Pym. The logic of bunched implications. *Bull. of Symb. Logic*, 5(2):215–244, 1999.
[29] F. Pallas. Information security inside organizations: A positive model and some normative arguments based on new institutional economics. PhD Thesis, University of Berlin, 2009.
[30] David Pym and Simon Shiu. Security analytics: Bringing science to security management. *IISP Pulse*, 4:12–13, 2010.
[31] D.J. Pym, P.W. O'Hearn, and H. Yang. Possible worlds and resources: The semantics of BI. *Theoretical Computer Science*, 315(1):257–305, 2004. Erratum: p. 285, l. -12: ", for some $P', Q \equiv P; P'$ " should be "$P \vdash Q$".
[32] Francisco J. Ruge-Murcia. Inflation targeting under asymmetric preferences. *J. of Money, Credit, and Banking*, 35(5), 2003.
[33] Norman E.L. Shanks and Alexandre L.W. Bradley. *Handbook of Checked Baggage Screening: Advanced Airport Security Operation*. WileyBlackwell, 2004.
[34] Colin Stirling. *Modal and Temporal Properties of Processes*. Springer Verlag, 2001.
[35] Transport Security Administration. Recommended Security Guidelines for Airport Planning, Design, and Construction. `http://www.tsa.gov/assets/pdf/airport_security_design_guidelines.pdf`, 2006.
[36] H. Varian. A bayesian approach to real estate management. In S.E. Feinberg and A. Zellner, editors, *Studies in Bayesian Economics in Honour of L.J. Savage*, pages 195–208. North Holland, 1974.
[37] A.T. Wells and S. Young. *Airport Planning and Management*. McGraw-Hill Professional, 2004.
[38] A. Zellner. Bayesian prediction and estimation using asymmetric loss functions. *J. Amer. Stat. Assoc.*, 81:446–451, 1986.

UCL, LONDON
E-mail address: `a.beautement@cs.ucl.ac.uk`

UNIVERSITY OF ABERDEEN
E-mail address: `d.j.pym@abdn.ac.uk`