Formalizing Physical Security Procedures

Catherine Meadows[1] and Dusko Pavlovic[2]

[1] Naval Research Laboratory, Washington, DC, USA
Email: meadows@itd.nrl.navy.mil
[2] Royal Holloway, Oxford and Twente
Email: dusko.pavlovic@rhul.ac.uk

Abstract. Although the problems of physical security emerged more than 10,000 years before the problems of computer security, no formal methods have been developed for them, and the solutions have been evolving slowly, mostly through social procedures. But as the traffic on physical and social networks is now increasingly expedited by computers, the problems of physical and social security are becoming technical problems. From various directions, many security researchers and practitioners have come to a realization that the areas such as transportation security, public and private space protection, or critical infrastructure defense, are in need of formalized engineering methodologies. Following this lead, we extended Protocol Derivation Logic (PDL) to Procedure Derivation Logic (still PDL). In contrast with a protocol, where some principals send and receive some messages, in a procedure they can also exchange and move some objects. For simplicity, in the present paper we actually focus on the security issues arising from traffic of objects, and leave the data flows, and the phenomena emerging from the interaction of data and objects, for future work. We illustrate our approach by applying it to a flawed airport security procedure described by Schneier.

Keywords: formal security protocol analysis, physical procedure analysis, physical security, security policies

1 Introduction

It is well known that the use of security protocols goes well beyond their application to communication between electronic devices. Any procedure that gives rules for interaction between a set of principals in order to provide security against misbehavior can be considered a security protocol in a broader since, whether the principals are human beings, computers, hand-held or embedded devices, or some mixture. Thus Ellison [10] proposed the idea of "ceremonies", which extend the notion of computer-to-computer security protocols to the human end users who interact with them. In [2] Blaze proposes an even further extension of the notion of security protocols: those that cover scripted interactions in general, for example, interactions between passengers and authorities in airports, railroad seat checks, and denial of service in burglar alarms.

Blaze's example of a flawed airport security procedure gives an idea of the kind of things that can go wrong. It arose from the fact that passengers arriving in

the US from overseas must clear their luggage through customs at their first point of entry, even if they are connecting to another flight within the US. Originally, passengers were not required to be checked through security again after clearing customs. They simply re-checked their luggage. This caused a problem when rules changed shortly after September 11, 2001, after which passengers could no longer carry knives aboard planes, but were allowed to put them in checked luggage. Until the rules were modified to require another security check, a passenger could circumvent the new policy by packing knives in his checked luggage, picking it up upon arrival in the US, transferring the knives to his person, passing through customs, dropping off his now knifeless luggage, and catching the next flight.

Another flawed airport security procedure was documented by Bruce Schneier in [20].Shortly after the "shoe bomber" incident, a new policy went into effect in many locations requiring shoes to be screened, but many airport security scanners could not screen passengers' feet. For a short while the following policy was implemented at Heathrow Terminal 3. A passenger would go through security, handing his carry-on to be scanned and himself passing through a security scanner, and pick it up upon emerging from the other end. Then he would pass to another station in which he would hand over his shoes to be scanned. Schneier points out that that a passenger could easily circumvent this procedure by hiding contraband items in his shoes, and packing another pair of shoes in his carry-on. After passing through the body scanner, the passenger could then switch shoes and proceed to the shoe scanning station, where his now innocent shoes would be given a clean bill of health.

These procedures, and the class of airport security procedures in general, belong to a larger class of procedures that govern the motion and location of principals and objects. Any procedure governing the access to a secure facility such as an airport or an office building falls into this category. As we see from the above examples, it is as easy for flaws to creep in as it is for cyber security protocols, especially when, as is often the case, new procedures are hastily implemented in response to a changed security situation.

In this paper we show how the logical framework we developed in [19] can be extended to reasoning about physical access procedures. This framework, called the Procedure Derivation Logic (PDL), extends our Protocol Derivation Logic (also PDL) [13, 3, 16, 1, 14, 18, 17], and the earlier ideas of Protocol Composition Logic (PCL) [9, 5, 4], to reason about the network interactions where principals may control complex network configurations, consisting of multiple nodes representing diverse devices, objects and data, with different channels between them, providing different security guarantees. Such a network configuration, as a concrete realization of the capabilities of a principal, is what we call an *actor*, with a respectful nod to Actor Network Theory [12], in an attempt to take into account some social interactions in security.

In [19] we formalized and analyzed some multi-factor, multi-channel authentication and key agreement procedures, one involving smart cards and card readers, and the other biometric devices and physical sources of randomness, together with the humans and the standard internet nodes. Both procedures

involved some authentic visual and social channels, together with the standard, insecure internet links. In the present work, we focus on the procedures where the humans control hierarchical configurations of physical objects, such as those that arise when we travel, packing our luggage, tickets and documents to satisfy complex security and safety requirements. Such procedures are effectively described as interactions between the actors, which include not only the passengers with their luggage, but also the various authorities and service providers, with their control devices, used for screening and transportation.

There are several important issues that these procedures bring forward. The first one is the *dynamics* of actors' configurations: e.g., during the check-in procedure, a passenger divests himself of some luggage, which becomes a part of another configuration; during security check, the passenger passes his carry-on luggage, and in some case his shoes, under the control of the screeners, who may or may not return these objects later on in the procedure. The next important issue is the *compositionality* of security procedures: e.g., the airport procedures usually come about as combinations of several simpler procedures, previously introduced to address different security concerns. A passenger interacts with various other actors in stages, where he receives his boarding pass at one stage, checks his luggage at another, then enters the screening area where he hands his carry-on over to the screener, and passes through a body scanning device with a certain probability, and so on. The problems with the airport security procedures tend to arise not because their individual components are implemented incorrectly, but because the properties that they guarantee are not adequate for the composite contexts into which they are introduced. Thus, the task of procedure design is often inseparable from the problem of procedure composition. While the same problem is familiar from protocol design, and the incremental approach of Protocol Derivation Logic usefully extends to Procedure Derivation Logic, the reasoning templates developed for security protocols do not suffice for analyzing the complex interactions of the heterogenous security components in networks of actors. The question then arises: at what level should we specify and reason about their compositions?

The answer to this question becomes clearer when we take a closer look at what airport security procedures and policies are regulating. They are concerned not only with authenticating a passenger's identity, and the integrity of his luggage, but also with constraining his movements, depending on the configuration of the objects and data that he controls. Moreover, the passenger's interactions with the airport authorities largely consist of movements: handing over luggage and ID, proceeding from one place to another when indicated. Finally, the breaches and circumventions of the procedures consist of movements as well, if a passenger finds a way to bring a weapon or a prohibited object into a secure area by finding a way to move them from one place to another along an unforeseen path. This situation, in which an attacker is deemed capable of performing any combination of a relatively small number of actions, should be familiar to those acquainted with the formal protocol analysis literature. It is exactly the kind of approach taken in the Dolev-Yao model [8, 7] in which the attacker is

assumed to be able to read, alter, and redirect traffic, as well as perform a small number of operations such as concatenation, deconcatenation, encryption, and decryption. This gives a regular structure to the attacker model which makes security protocols amenable to both logical analysis and model checking.

In this paper we show how we use these insights to develop a logical system for the analysis of policies on of moves, with an application to airport security procedures. In Section 2 we extend the notion of a network configuration, that endowed principals with the structure of actors in [19], into the notion of *box configuration*, that describes how some objects contain other objects. This turns out to be convenient for expressing the basic goals and methods of, e.g., airport security procedures, since the complex configurations of objects controlled by the participating actors are usually enclosed into physical box configurations. In Section 3 we provide an overview of a *box configuration logic* in which conditions and consequences of moves can be reasoned about. In Section 4 we apply the logic to airport security procedures, by specifying two different airport security procedures in terms of the constraints on passenger movements imposed a a result of the various subprocedures a passenger must engage in order to pass into the secured area. We also use the logic to analyze the procedures, and show how, in the case of the shoe procedure, the attempt that prove that the security goals are satisfied fails. In Section 5 we discuss related work. In Section 6 we conclude the paper and discuss future directions, in particular our plans for extending our work to include the analysis of interactions and data flows between principals.

2 Configurations and box configurations

In this section we introduce the basic notion of configurations and box configurations. A configuration is a recursively defined set of sets. A box configuration is a recursively defined set of possibly labeled sets. Both configurations and box configurations may also be thought of as unordered trees.

2.1 Basic definitions

A configuration is a collection of nodes that operate jointly in the execution of a procedure. A *box configuration* is a special case of a slightly expanded notion of the definition of configuration introduced in [19]. There we defined a configuration over a set S to be either a subset of S or a set of configurations over S. For box configurations, we find it convenient to allow sets made up of both elements of S and configurations. Thus we use the following definition:

Definition 1. *A* configuration over a set S *is either an element of* S *or a finite set of configurations over* S*, i.e.* $C ::= s \mid \{\} \mid \{C_0, C_1, \ldots, C_n\}$*, where* $s \in S$ *and* $n \geq 0$*. The empty configuration is* $\{\} = \emptyset$*. The set of* S*-configurations is* $\mathcal{C}(S)$*.*

Configurations are thus the elements of what set theorists would call a *cumulative hierarchy of finite sets*, generated by a set of *atoms* S. It is easy to see

that each S-configuration corresponds to a unique finite tree where the leaves may be labelled by the elements of S or the empty label (i.e., the empty set). The subconfigurations of a configuration are those corresponding to the subtrees of its tree representation.

Now we introduce the notion of a box configuration:

Definition 2. *A* box configuration *over a set S, or an S-box configuration, is either an S-configuration, or an S-box configuration in a* box *b, i.e. $F ::= s \mid \{\} \mid \{F_0, \ldots, F_n\} \mid \{F_0, \ldots, F_n\}_b$ where the boxes b are distinguished elements of S. For simplicity, we assume that all elements of S can be boxes. The set of S-box configurations is written $\mathcal{F}(S)$. The element relation is generated by the clauses $F_0, \ldots, F_n \in \{F_0, \ldots, F_n\}$ and $b, F_0, \ldots, F_n \in \{F_0, \ldots, F_n\}_b$. We say that $\{F_0, \ldots, F_n\}_b$ is* rooted in b. *We use the notation X_b to denote an arbitrary box configuration rooted in b.*

Example 1. A box configuration describing a passenger *pa* with a newspaper *np*, a phone *ph* and sunglasses *sg* in his pocket, and a suitcase *sc* containing a knife *kn* and an explosive device *ex* can be written as $\{np, \{ph, sg\},\ \{kn, ex\}_{sc}\}_{pa}$.

Definition 3. *We say that a box configuration A is* contained *in a box configuration C, or that A is a part of C's* contents*, and write*

$$A \sqsubseteq C \iff A = C \ \vee\ \exists B.\ A \sqsubseteq B \in C \text{ and } A \sqsubset C \iff A \sqsubseteq C \ \wedge\ A \neq C$$

Example 2. In Example 1 above, both $kn \sqsubset \{np, \{ph, sg\},\ \{kn, ex\}_{sc}\}_{pa}$ and $\{kn, ex\}_{sc} \sqsubset \{np, \{ph, sg\},\ \{kn, ex\}_{sc}\}_{pa}$ hold.

Configurations and box configurations as trees

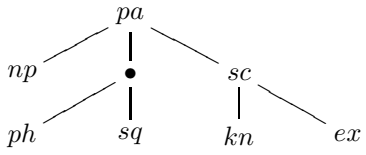
Proposition 1. *The set $\mathcal{C}(S)$ of S-configuration is in bijective correspondence with the set of finite unordered* irredundant *trees with whose leaves labelled by elements of S or the empty label. (A tree is irredundant if no two child trees of any node are identical (see [15], Sec. 5.2 for a discussion).)The set $\mathcal{F}(S)$ of S-box configurations is in bijective correspondence with the set of finite unordered irredundant trees whose nodes are labelled by elements of S or the empty label.*

For reasons of space the proof is omitted. We note in particular that if A and B are two box configurations, then $A \sqsubseteq B$ only if the tree associated with A is a subtree of the tree associated with B.

We will be particularly interested in box configurations whose labels are all different; i. e. each labelled node corresponds to a unique object.

Definition 4. *A box configuration C is* normal *if no two nodes of the corresponding tree are labelled by the same atom from S.*

Example 3. Example 1 can also be depicted as the following labelled tree



2.2 Types and Constraining Set Membership

We put a typing structure on S, where a type is simply a subset of S. Types are partially ordered by the subset relation. The typing is also applied to box configurations of the form X_s; if $s \in S$ is of type $\mathbf{t}$, then so is X_s.

One important application of types is in the expression of restrictions on what can box configurations can be elements of what other box configurations. For example, we may want to specify that an passenger can be inside an airplane, but an airplane can't be inside a passenger. We define a relation $\ltimes$ on box configuration types, where $\mathbf{a} \ltimes \mathbf{b}$ if and only if box configurations of type $\mathbf{a}$ can be elements of box configurations of type $\mathbf{b}$.

Example 4. Let **pa** be the type passenger, **su** be the type suitcase, **kn** be the type knife, and let **ex** be the type explosive. Then $\mathbf{su}, \mathbf{kn}, \mathbf{ex} \ltimes \mathbf{pa}$ and $\mathbf{kn}, \mathbf{ex} \ltimes \mathbf{su}$.

We note that, although the $\ltimes$ relation defined in Example 4 is transitive, this is not always the case. For example, if we have box configurations of the type principal, room, and building, it may make sense to say that a principal may be an element of a room, and a room may be an element of a building, but the principal can't be an element of a building. That is, a principal can't be contained in a building unless he is in a room in the building.

Definition 5. *We define the* multiplicity *of a type* $\mathbf{t}$ *in a box configuration* C*, denoted by* $mult(C, \mathbf{t})$ *as follows: 1)* $mult(\emptyset, \mathbf{t}) = 0$*, 2)* $mult(s, \mathbf{t}) = 1$ *if* $s \in \mathbf{t}$*, else* $mult(s, \mathbf{t}) = 0$*. 3)* $mult(\{F_1, \ldots F_n\}, \mathbf{t}) = \sum_{i=0}^{n} mult(F_i, \mathbf{t})$*, and 4)* $mult(X_s, \mathbf{t}) = mult(X, \mathbf{t}) + mult(s, \mathbf{t})$*.*

We also use $mult(b, \mathbf{t})$ to refer to the multiplicity of $\mathbf{t}$ in the box configuration rooted in b, when we can avoid confusion.

Example 5. Let $PA = \{\{kn_1, kn_2, ex_1\}_{su1}, su_2\}_{pa}$ where $pa \in \mathbf{pa}$, $su_1, su_2 \in \mathbf{su}$, $kn_1, kn_2 \in \mathbf{kn}$, and $ex_1 \in \mathbf{ex}$. Then $mult(PA, \mathbf{kn}) = 2, mult(PA, \mathbf{su}) = 2$, and $mult(PA, \mathbf{ex}) = 1$.

The following property of multiplicities, which follows straightforwardly from the properties of trees, will also be useful to us.

Proposition 2. *Let* X *and* Y *be box configurations, and let* $\mathbf{t}$ *be a type. Then if* $X \sqsubseteq Y$*, then* $mult(X, \mathbf{t}) \leq mult(Y, \mathbf{t})$*.*

Example 6. Let X_{pa} *be a passenger in the secure area sa of an airport, i. e.* $X_{pa} \sqsubset Y_{sa}$*. Suppose that it is known that the secure area contains no explosives, that is* $mult(sa, \mathbf{ex}) = 0$*. Then we can conclude that* $mult(pa, \mathbf{ex}) = 0$*.*

2.3 Displacing Subtrees

A key feature of our logical system is the ability to reason about what occurs when a box subconfiguration moves from one part of a box configuration to another. In order to capture what happens in these circumstances, we define the notion of box subconfiguration displacement. We first introduce some notation.

Definition 6. *Suppose that X, Y, and Z are box configurations such that $X \in Y$ and $X \notin Z$. We denote $Y \setminus \{X\}$ by $Y \ominus X$ and $Z \cup \{X\}$ by $Z \oplus X$. If $Y, Z \sqsubseteq W$, we denote the result of replacing Y in W with $Y \ominus X$ by $W[Y \ominus X]$, and the result of replacing Z in W with $Z \oplus X$ by $W[Z \oplus X]$.*

Definition 7. *Suppose that $X \in Z \sqsubseteq U$ and $Y \sqsubseteq U$ where U is normal. We define a* displacement of X from under Z to under Y in U *, denoted by $U[Y \ominus X, Z \oplus X]$, as follows. If $Z \neq Y$, then replace Z with $Z \ominus X$ and replace Y with $Y \oplus Z$. Else, if $Z = Y$, then $U[Y \ominus X, Z \oplus X] = U$.*

Example 7. Suppose that two passengers pa_1 and pa_2 are in an airport ap. Suppose that one passenger is carrying a suitcase, in which there is a knife, and hands the knife to the other passenger. The original box configuration is $U = \{\{\{kn\}_{sc}\}_{pa_1}, pa_2\}_{ap}$. The new one is is $\{\{sc\}_{pa_1}, \{kn\}_{pa_2}\}_{ap}$.

The next proposition follows directly from the properties of normal trees.

Proposition 3. *Suppose U is normal, and that $X \sqsubseteq U$ and $Y \sqsubseteq U$. Then*

1. *If $X \in Z \sqsubseteq Q \sqsubseteq U \wedge Q \cap Y = \emptyset$ then Q is replaced in $U[Y \ominus X, Z \oplus X]$ with $Q[Z \ominus X]$ and $mult(Q[Z \ominus X], \mathbf{m}) = mult(Q, \mathbf{m}) - mult(X, \mathbf{m})$.*
2. *If $Z \sqsubseteq Q \sqsubseteq U \wedge X \cap Q = \emptyset \wedge Y \sqsubseteq Q$ then Q is replaced in $U[Y \ominus X, Z \oplus X]$ with $Q[Y \oplus X]$ and $mult(Q[Y \oplus X], \mathbf{m}) = mult(Q, \mathbf{m}) + mult(X, \mathbf{m})$*
3. *If $X \in Z \sqsubseteq Q \sqsubseteq U \wedge Y \sqsubseteq Q$ then Q is replaced in $U[Y \ominus X, Z \oplus X]$ with $Q[Y \ominus X, Z \oplus X]$ and $mult(Q[Y \ominus X, Z \oplus X], \mathbf{m}) = mult(Q, \mathbf{m})$.*

3 A Logic of Moves

In this section we present our logic of moves. In Section 3.1 we give the semantic underpinnings of the logic, in Section 3.2, a policy language for moves, and in Section 3.3, a logical system for proving that a policy specified in the policy language implements a policy defined in terms of state invariants on runs.

3.1 States, Runs, and Processes

We consider a process $\mathcal{P}$ made up of nondeterministic concurrent processes P, each associated to a unique principal p, that act upon a single state variable $\mathbf{U}$ whose value is a normal box configuration called the *universe*. When we can avoid confusion, we will refer to a state by the current value of $\mathbf{U}$. P acts on $\mathbf{U}$ by assigning to it a new version of the universe that has been altered by moving some box subconfiguration O of the universe to another box subconfiguration Y, resulting in a displacement of O to Y. We refer to such actions as *move actions*.

We make use of the following notation:

Definition 8. *A* move action of O from under Y to under X executed by principal p *is a state transition in which the preceding state U satisfies $O \in Y \sqsubseteq U$*

and the succeeding state is $U[Y \ominus X, O \oplus X]$. *It is denoted by* $p : X \xrightarrow[\in]{O} Y$. *A* move action of O from X to Y executed by p *is an action* $p : R \xrightarrow[\in]{O} Q$ *in which* U *satisfies* $O \in R \sqsubseteq X \sqsubseteq U$ *and* $Q \sqsubseteq Y \sqsubseteq U$ *for some* R, Q *immediately before the transition. It is denoted by* $p : X \xrightarrow{O} Y$.

We note that $p : X \xrightarrow[\in]{O} Y$ denotes a unique action, while $p : X \xrightarrow{O} Y$ denotes one of a possible set of actions. We define $p : X \xrightarrow{O} Y$ in this way because in many cases we do not care whether an object is an element of another as long as it is a box subconfiguration of the other. For example, if A puts a knife into B's suitcase, she can be considered as having given it to B.

Move actions are assumed to be *atomic*, so that a condition that holds when a move action begins to execute continues to hold until it is finished.

Example 8. The action in Example 7 can be represented as
$pa_1 : \{\{\{kn\}_{sc}\}_{pa_1}, pa_2\}_{ap} \xrightarrow{kn} pa_2$

Moves can describe a number of different actions. Let Z_p and W_q be box configurations of type principal, and let X,Y, O be of some other type. Then,

1. $p : X \xrightarrow{Z_p} Y$ describes a principal p moving from X to Y.
2. $p : Z_p \xrightarrow{O} W_q$ describes one principal p giving O to another, q.
3. $p : W_q \xrightarrow{O} Z_p$ describes one principal taking O from another.
4. $p : Z_p \xrightarrow{O} Y$ describes a principal putting O in Y.
5. $p : X \xrightarrow{O} Z_p$ describes a principal removing O from X.
6. $p : X \xrightarrow{O} Y$ describes a principal p moving O from X to Y.

Move polices are conditions on what moves may be taken and under what conditions. Move polices are defined locally, but the intended consequences are global, defined in terms of runs, as follows.

Definition 9. *A* run *is an alternating sequence of states and moves*

$$U_1, p_1 : X_1 \xrightarrow{O_1} Y_1, U_2, \ldots, U_i, p_i : X_i \xrightarrow{O_i} Y_i, U_{i+1}, \ldots$$

A *policy* $\mathcal{R}$ defines a set of legal runs. For example, the legal runs of the airport policy are those in which, in every state, the multiplicity of knives and explosives in the secure area is zero.

3.2 Syntax of Move Policies

For practical reasons, one enforces a policy, not by controlling the states, but by controlling the actions. We make this more precise below.

Definition 10. *Let $\mathcal{P}$ be a policy, and $\mathcal{R}$ be the set of runs satisfying that policy, let a be an action, and let f be a predicate on states. We say that f* guards *a , written as $f \Rrightarrow A$ if, for every run $R \in \mathcal{R}$, if S immediately precedes a in $\mathcal{R}$, then $f(S)$ holds.*

Definition 11. *A* move policy *is a set of statements describing under what conditions an action may take place. The syntax of move policies is given below, where X is a variable representing a boxconfiguration, and p, $\mathbf{t}$, and n, are each a variable or constant representing respectively a principal, type, and integer.*

$$
\begin{aligned}
C &::= X \mid b \mid p \mid \{C\} \mid C_b \mid C_1[C_2 \ominus C_3] \mid C_1[C_2 \oplus C_3] \mid \{C|R\} \mid \\
&\quad C_1[C_3 \ominus C_2, C_4 \oplus C_2] \mid \{C_1, \ldots, C_k\} \\
E &::= n \mid \mathit{mult}(C, \mathbf{t}) \mid \mathit{mult}(b, \mathbf{t}) \mid E_1 \times E_2 \mid E_1 - E_2 \mid E_1 + E_2 \\
R &::= E_1 = E_2 \mid E_1 \leq E_2 \mid E_1 < E_2 \mid C_1 \sqsubset C_2 \mid C_1 \sqsubseteq C_2 \mid C_1 = C_2 \mid \\
&\quad \mathbf{t_1} \ltimes \mathbf{t_2} \mid b \in \mathbf{t} \mid R_1 \Rightarrow R_2 \mid \mathit{not}(R) \mid R_1 \vee R_2 \mid R_1 \wedge R_2 \mid \exists X.R \mid \bot \mid \top \\
A &::= p : C_1 \xrightarrow{C_3} C_2 \mid p : C_1 \xrightarrow[\in]{C_3} C_2 \\
G &::= R \Rrightarrow A
\end{aligned}
$$

Variables are assumed to be universally quantified unless bound by $\exists$.

Example 9. Given two locations, the unsecured area *ua*, and the secure area *sa*, such that no object or person containing a knife may enter *sa*, we write the move policy as $\mathit{mult}(Z, \mathbf{kn}) = 0 \Rrightarrow p : O_{ua} \xrightarrow{O} Y_{sa}$.

3.3 Move Logic

In this section we give the logic of moves that will be used to show that a move policy enforces a policy on runs that is described in terms of state invariants.

Statements in the move logic are of two forms. They can be Hoare triples, which say that if a state satisfies predicate f_1 before an action, then it satisfies predicate f_2 after that action:

$$\{f_1\}p : X \xrightarrow[\in]{O} Y\{f_2\} \quad \text{or} \quad \{f_1\}p : X \xrightarrow{O} Y\{f_2\}$$

Statements can also be expressed in terms of move policy statements as defined in Section 11. If no policy statement exists for an action a, then $\bot \Rrightarrow a$ holds.

The idea is to specify a safety policy in terms of invariants on states that are preserved in all possible runs. A move policy is then specified as in Definition 11. One then uses the Hoare logic to determine the result of enforcing the policy. This is done by finding, for each move action a, and each invariant g specified by the policy, a guard f such that $\{f \wedge g\}a\{g\}$ holds.

We use the standard inference rules in Hoare logic (e.g. as presented in [11]), with the exception of the while rules and the rule for composition, which are not

sound with respect to our concurrent semantics. We also use the inference rules for first-order logic and the following rule governing guards:

$$\frac{f_1 \Rrightarrow a \qquad f_2 \Rrightarrow a}{f_1 \wedge f_2 \Rrightarrow a}$$

We also give the following axioms describing results of moves, which follow directly from the definitions given in Section 3.1.

$$\{\mathbf{U} = U\}p : X \xrightarrow[\in]{O} Y\{\mathbf{U} := U[X \ominus O, Y \oplus O]\} \tag{1}$$

$$O \in X \sqsubseteq U \Rrightarrow p : X \xrightarrow[\in]{O} Y \tag{2}$$

$$\{\mathbf{U} = U\}p : X \xrightarrow{O} Y\{\exists Q, R.Q \sqsubseteq X \wedge R \sqsubseteq Y \wedge \mathbf{U} := U[Q \ominus O, R \oplus O]\} \tag{3}$$

$$O \sqsubseteq X \sqsubseteq U \Rrightarrow p : X \xrightarrow{O} Y \tag{4}$$

4 Airport Security Procedure

We develop the airport security procedure in an incremental fashion. We begin with the simplest case, with only spatial constraints on passenger movements. We then add a step in which passengers are checked for forbidden items. We next add the step in which passenger carry-ons are checked separately. We finish with an insecure procedure in which the passenger's shoes are checked separately.

4.1 Locations, Passengers and the Items Passengers May Carry

In this section we specify the various types used in the procedure, and the subtype relation between them. First is the passenger type **pa**: a subtype of principal. Next is the type **su** for suitcase. The type **su** has three subtypes, large suitcase **ls**, small suitcase **ss**, and personal item **pi**. We also have type disallowed, or **da**, with subtypes knives **kn** and explosives **ex**. We also have the types **ft** for foot and **sh** for shoes. Finally, we have the type location **lo** and the universe **u**.

We give the relation $\ltimes$ defining what box configurations may contain others:

$\mathbf{lo} \ltimes \mathbf{u}$ $\mathbf{sh} \ltimes \mathbf{ft}$	$\mathbf{ls} \ltimes \mathbf{t.t} \in \{\mathbf{pa}, \mathbf{lo}\}$ $\mathbf{ss} \ltimes \mathbf{t.t} \in \{\mathbf{ls}, \mathbf{pa}, \mathbf{lo}\}$	$\mathbf{pi} \ltimes \mathbf{t.t} \in \{\mathbf{ss}, \mathbf{ls}, \mathbf{pa}, \mathbf{lo}\}$ $\mathbf{da} \ltimes \mathbf{t.t} \in \{\mathbf{sh}, \mathbf{pi}, \mathbf{ss}, \mathbf{ls}, \mathbf{pa}, \mathbf{lo}\}$	$\mathbf{pa} \ltimes \mathbf{lo}$

Our goal is to prove that all actions maintain the invariant $mult(sa, \mathbf{da}) = 0$.

4.2 Procedure With Spatial Constraints Only

We begin with just three locations: outside the airport, or *os*, the unsecured area inside the airport, or *ua*, and the secure area inside the airport, or *sa*.

At this point, the security controls are minimal. We do make a few restrictions however: the only thing a passenger can move from one location to another is himself, and there are restrictions on the areas he can move between. This can be expressed in the move logic syntax as the disjunction of atomic conditions, but for the sake of conciseness and readability we express it using a relation $\mathcal{R}_1 = \{(os, ua), (ua, os), (ua, sa)(sa, ua)\}$. We specify the permitted move actions:

$$(\ell_1, \ell_2) \in \mathcal{R}_1 \Rrightarrow pa : X_{\ell_1} \xrightarrow{Z_{pa}} Y_{\ell_2} \tag{5}$$

$$lo \in \mathbf{lo} \wedge Z \sqsubset X \wedge U_{pa} \sqsubset V_{lo} \wedge X \sqsubset V_{lo} \wedge Y \sqsubset V_{lo} \Rrightarrow pa : X \xrightarrow{Z} Y \tag{6}$$

Axiom 5 tells us that a passenger can move from one location ℓ_1 to another location ℓ_2 only if $(\ell_1, \ell_2) \in \mathcal{R}_1$. Axiom 6 tells us that a passenger can move a box configuration (other than himself) from X to Y only if X, Y, and the passenger are in the same location.

By applying Proposition 3, we see that the only way a disallowed item can pass into the secured area sa is by a passenger carrying it passing from the unsecured area ua into sa. Thus one way to prevent disallowed items from entering sa is to prevent passengers carrying them from moving from ua to sa.

4.3 Adding Passenger Screening

In order to add passenger screening, we add the location it will take place in: the passenger screening area, psa. We replace the passage between ua and sa in $\mathcal{R}_1$ with passages between ua and psa and between psa and sa:

$$\mathcal{R}_2 = (\mathcal{R}_1 / \{(ua, sa)\}) \cup \{(ua, psa), (psa, ua), (psa, sa)\}$$

We now replace Axiom 5 with

$$(lo_1, lo_2) \in \mathcal{R}_2 \Rrightarrow pa : X_{lo_1} \xrightarrow{Z_{pa}} Y_{lo_2} \tag{7}$$

We next need to include a condition saying that the passenger can be carrying no large suitcases, and only one small suitcase and one personal item. However, we note that it is not enough to count multiplicities in the passenger, since a passenger can carry more than one personal item as long as all but one are in the small suitcase. That is, the multiplicity of personal items in the box configuration formed by deleting the small suitcase box configuration (if one exists) from the passenger box configuration should be no more than one. We write this as follows.

$$Z_{pa} \sqsubset X_{sl} \wedge mult(Z_{pa}, \mathbf{ls}) = 0 \wedge mult(Z_{pa}, \mathbf{ss}) \leq 1 \wedge$$
$$\wedge mult(Z_{pa}, \mathbf{pi}) - mult(\{W_{ss} | W_{ss} \sqsubset Z_{pa}\}, \mathbf{pi}) \leq 1 \Rrightarrow pa : X_{sl} \xrightarrow{Z_{pa}} Y_{psa} \tag{8}$$

Axiom 6 remains the same.

We now add an axiom saying that the passenger can move from the passenger screening area to the secure area only if he is carrying no disallowed items.

$$Z_{pa} \sqsubset X_{psa} \wedge mult(pa, \mathbf{da}) = 0 \Rrightarrow pa : X_{psa} \xrightarrow{Z_{pa}} Y_{sa} \tag{9}$$

Proposition 4. *Suppose that Axioms 6 7, and 9 hold. Then permitted action a preserves the invariant.*

Proof. (Sketch) By Proposition 3 the only actions permitted by the policy that we need to consider are those of the form $a = pa : X_{psa} \xrightarrow{Z_{pa}} Y_{sa}$, guarded by $mult(pa, \mathbf{da}) = 0$. Since, according to Axiom 9, a is guarded by $mult(pa, \mathbf{da}) = 0$, it remains to prove that $\{mult(pa, \mathbf{da}) = 0 \wedge mult(sa, \mathbf{da}) = 0\} a \{mult(sa, \mathbf{da}) = 0\}$. But this again follows directly from Proposition 3.

4.4 Removing Carry-on

In the above procedure, we specified a single screening step. But the passenger screener actually consists of two screeners, one for the passenger, and one for the carry-on. We refine the screening step by introducing two new locations: the carry-on screener ca, and the passenger screener ps. The passenger now goes from pa to ps to sa, and before he goes through ps, he is expected to deposit any carry-on in ca. Thus we begin by defining a new relation $\mathcal{R}_4$ on locations.

$$\mathcal{R}_3 = (\mathcal{R}_2 / \{(psa, sa)\}) \cup \{(psa, ps), (ps, psa), (ps, sa)\}$$

Axiom 7 is replaced with:

$$(lo_1, lo_2) \in \mathcal{R}_3 \Rrightarrow Z_{pa} : X_{lo_1} \xrightarrow{Z_{pa}} Y_{lo_2} \tag{10}$$

We also introduce an exception to the rule not allowing passengers to move items from one location to another by allowing passengers in the passenger screening area to place any item they are is carrying in the carry-on screener. Thus Axiom 6 is replaced by the following:

$$\begin{aligned}(Z_{pa} \sqsubset W_{lo} \wedge X \sqsubset W_{lo} \wedge Y \sqsubset W_{lo}) \vee (Z_{pa} \sqsubset U_{psa} \wedge T \sqsubset Z_{pa} \wedge Y \sqsubset V_{ca}) \\ \Rrightarrow Z_{pa} : X \xrightarrow{T} Y\end{aligned} \tag{11}$$

Furthermore, a passenger moving from the passenger screening area to the passenger screener must not be carrying any small suitcases or personal items:

$$mult(pa, \mathbf{ss}) = mult(pa, \mathbf{pi}) = 0 \Rrightarrow pa : X_{psa} \xrightarrow{Z_{pa}} Y_{psa} \tag{12}$$

Axiom 9 is replaced by the following, which applies the same conditions to the passenger's movement from the passenger screener to the secure area.

$$Z_{pa} \sqsubset X_{ps} \wedge mult(Z_{pa}, \mathbf{da}) = 0 \Rrightarrow pa : X_{ps} \xrightarrow{Z_{pa}} Y_{sa} \tag{13}$$

To get the carry-on into the secure area, we introduce another type of principal: the airport authority **aa**, and a subtype of this principal, the carry-on screening authority **csaa**. The responsibility of **csaa** is to remove carry-ons from the carry-on screener to the secure area if they are free of disallowed items. Thus:

$$Z_{ca} \sqsubset X_{cs} \wedge mult(ca, \mathbf{da}) = 0 \Rrightarrow csaa : X_{cs} \xrightarrow{Z_{ca}} Y_{sa} \tag{14}$$

Proposition 5. *Let SA denote the secure area. Suppose that Axioms 11, 13, 10, 8, ,12 and 14 hold. Then any permitted action a preserves the invariant.*

Proof. The proof is similar to that of Proposition 4 and we omit it.

4.5 Screening Shoes

To reproduce the Heathrow solution, we introduce two new locations : the post-passenger screening area $ppsa$, and the shoe screener shs. The passenger now goes from psa to $ppsa$ to the shs, and finally to the secure area sa. Likewise, the carry-on screener authority puts cleared carry-on in $ppsa$, not in the sa. The check done in psa is on all of the passenger except his feet:

$$\mathcal{R}_4 = (\mathcal{R}_3 / \{(ps, sa)\}) \cup \{(ps, ppsa), (ppsa, ps), (ppsa, shs), (shs, ppsa), (shs, sa)\}$$

$$(lo_1, lo_2) \in \mathcal{R}_4 \Rrightarrow ps : X_{lo_1} \xrightarrow{Z_{pa}} Y_{lo_2} \tag{15}$$

$$mult(Z_{pa}, \mathbf{da}) - mult(\{W_{ft} | W_{ft} \sqsubset Z_{pa}\}, \mathbf{da}) = 0 \Rrightarrow pa : X_{ps} \xrightarrow{Z_{pa}} Y_{sa} \tag{16}$$

$$Z_{ca} \sqsubset X_{cs} \wedge mult(Z_{ca}, \mathbf{da}) = 0 \Rrightarrow csaa : X_{cs} \xrightarrow{contZca} Y_{ppsa} \tag{17}$$

We also include an axiom describing conditions in which the passenger passes from the shoe screener to the secure area. If the passenger has one or more feet then each foot must have a shoe, a shoe that is free of disallowed items:[3]

$$(W_{ft} \sqsubset Z_{pa} \Rightarrow V_{sh} \sqsubset W_{ft}) \wedge mult(sh, \mathbf{da}) = 0) \Rrightarrow pa : X_{ssh} \xrightarrow{Z_{pa}} Y_{sa} \tag{18}$$

The relevant axioms are now Axioms 8, 12 , 14, 15, 16, 18 and 17.

[3] We note that the policy does not take into account the existence of socks, and the fact that a passenger, could, e.g., hide a knife in his socks. This reflects the Heathrow policy, which in our experience did not require passengers to remove their socks.

We now try to prove that the procedure still satisfies the security policy. There is no longer a guard requiring passengers moving to *sa* to be free of disallowed items, so we cannot apply Proposition 3 directly as before. However, all box configurations entering *sa* do so through the shoe screener. Thus, if we can show that for any action a, $\{mult(ssh, \mathbf{da}) = 0\}a\{mult(ssh, \mathbf{da}) = 0\}$, we can use Proposition 2 to obtain that any box configuration within *ssh* contains no disallowed items, and thus use Proposition 3(1) to prove that for any action a moving a box configuration from *ssh* to *sa*, $\{mult(SA, \mathbf{da}) = 0\}a\{mult(SA, \mathbf{da}) = 0\}$.

But this is not the case, because shoes are not checked before being moved from *ppsa* to *ssh*. Moreover, passengers and shoes are together in the *ppsa*, and, according to Axiom 6, passengers can move disallowed objects from their shoes to their persons when both passengers and shoes are both in the same location. Thus, we can show that the procedure does not satisfy the security property by constructing a scenario in which a passenger moves a disallowed item from his shoes to his person (or, to give Schneier's example, to his carry-on).

5 Related Work

There has been a substantial amount of work on the logical modeling of location and movement. Much of it is motivated by the desire to give as accurate picture of the relevant details of the world as possible (e.g. for modeling transmission in wireless networks), and is thus beyond the scope of what we are attempting to achieve. Another area that has been extensively studied is the security of mobile processes, in which, although location is relevant and included in the model, is still focused on cyber networks, not human or mixed cyber and human security procedures. However, as we have noted, there has been increasing awareness of the need to extend the concept of network security, and with it its formal analysis beyond cyberspace. We discuss some closely related work below.

Portunes [6] models the physical, social, and digital interactions that take place in order to defend against or execute a cyber attack. Portunes covers a wide range of attacks, including social engineering; however the writer of a Portunes specification needs to specify the individual steps (although not the attack sequence) taken by the intruder. Thus, if an attacker needs to attach a dongle to a computer to load a rootkit, one can write that the attacker can offer a dongle to an employee. In our approach, one would write an axiom governing the deposition of items in the computer box configuration. The application of our logic would uncover what the permissible flows are (that is, whether objects not supplied by the company can be deposited on a computer in a secure room), but not the specific attack. This difference arises from the somewhat different goals of Portunes, which is designed to generate attack scenarios for further analysis, and ours, which is designed for proving (or disproving) security properties.

In [22] Srivatanakul applies different techniques developed for assessing system safety to the analysis of security policies involving authorization and containment, including that used by the Bangkok International Airport's baggage

handling system. The techniques used included methods such as hazard analysis, fault tree analysis, and mutation testing of formal specifications. In the case of the baggage handling study, they were particularly useful in identifying potential consequences of the security assumptions not being satisfied. These results are complementary to ours, and we can see Srivatanakul's and our techniques being applied iteratively in tandem, with the safety analysis techniques used to identify risks, and ours used to evaluate mitigations.

In [21] Scott presents a language for specifying concerning location and movement. His model and ours have some features in common, as they both involve nested box configurations and conditions on moves, but it provides only a method for specifying policies, not reasoning about them. However, he discusses the possibility of analysis as well as specification, and it possible that an approach such as ours to reasoning about moves might be realizable within his framework.

6 Conclusion

We have given a logical framework for reasoning about permitted and required movements, and have shown how it can be applied for reasoning about the security of procedures that govern the movement of human beings, such as airport security procedures. It should be clear though that there is still much relevant material that is left out. This is everything covering the passenger's interactions with the airport authorities. It is not enough, for example, that a passenger's carry-on must not contain disallowed items. Instead an airport security agent, or rather, a configuration consisting of the agent and the screening device, must observe that the bag is free of disallowed items, and that the passenger has appropriate authorization and documents.

We have avoided these issues in this paper in order to have as clean a model as possible of the properties that need to be checked for, as opposed to the process of checking. But in [19] we developed a framework for reasoning about interactions between configurations across channels that is intended to capture just those types of procedures, and the framework set forth in this paper is designed to be complementary to it. In future work we plan to combine these two approaches to obtain a comprehensive methodology for reasoning about procedures that govern movement.

We also see this work as having potential for application in many other areas. There has been a substantial interest in security policies and procedures involving location, including location-based access control, secure verification of location, and location privacy. However, movement is usually only considered as a means of hiding or tracking location. We believe that the including policies about movement and logical methods for reasoning about it, can lead, for example, to richer and more meaningful location-based access control policies. For example, a passenger in a secure area could be granted certain privileges *because* he or she has been verified to be free of knives and explosives. A method of reasoning about the procedure by which that conclusion was reached would help in ascertaining the safety of such a policy.

References

1. M. Anlauff, D. Pavlovic, R. Waldinger, and S. Westfold. Proving authentication properties in the Protocol Derivation Assistant. In *Proc. FCS-ARSPA 2006*. ACM, 2006.
2. M. Blaze. Toward a broader view of security protocols. In *Security Protocols Workshop*, LNCS vol. 3957, pp. 106–120. Springer, 2004.
3. I. Cervesato, C. Meadows, and D. Pavlovic. An encapsulated authentication logic for reasoning about key distribution protocols. In *Proc. CSFW 2005*, IEEE, 2005.
4. A. Datta, A. Derek, J. Mitchell, and A. Roy. Protocol composition logic (PCL). *Electron. Notes Theor. Comput. Sci.*, 172:311–358, 2007.
5. A. Datta, A. Derek, J. Mitchell, and D. Pavlovic. A derivation system and compositional logic for security protocols. *J. of Comp. Security*, 13:423–482, 2005.
6. T. Dimkov, W. Pieters, and P. H. Hartel. Portunes: Representing attack scenarios spanning through the physical, digital and social domain. In *ARSPA-WITS*, LNCS vol. 6186, pp. 112–129. Springer, 2010.
7. D. Dolev, S. Even, and R. M. Karp. On the security of ping-pong protocols. *Information and Control*, 55(1-3):57–68, 1982.
8. D. Dolev and A. C.-C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983.
9. N. Durgin, J. Mitchell, and D. Pavlovic. A compositional logic for proving security properties of protocols. *J. of Comp. Security*, 11(4):677–721, 2004.
10. C. Ellison. Ceremony design and analysis. Cryptology ePrint Archive, Report 2007/399, October 2007.
11. D. Gries. *The Science of Programming*. Springer, 1981.
12. B. Latour. *Reassembling the Social: An Introduction to Actor-Network Theory*. Oxford University Press, 2005.
13. C. Meadows and D. Pavlovic. Deriving, attacking and defending the GDOI protocol. In *Proc. ESORICS 2004*, LNCS vol. 3193, pp. 53–72. Springer Verlag, 2004.
14. C. Meadows, R. Poovendran, D. Pavlovic, L. Chang, and P. Syverson. Distance bounding protocols: authentication logic analysis and collusion attacks. In R. Poovendran, C. Wang, and S. Roy, editors, *Secure Localization and Time Synchronization in Wireless Ad Hoc and Sensor Networks*. Springer Verlag, 2006.
15. D. Pavlovic. Categorical logic of concurrency and interaction I. synchronous processes. In *Theory and Formal Methods of Computing 94*, pp. 105–141. World Scientific, 1995.
16. D. Pavlovic and C. Meadows. Deriving secrecy properties in key establishment protocols. In *Proc. ESORICS 2006*, LNCS vol. 4189, 2006.
17. D. Pavlovic and C. Meadows. Bayesian authentication: Quantifying security of the Hancke-Kuhn protocol. *E. Notes in Theor. Comp. Sci.*, 265:97 – 122, 2010.
18. D. Pavlovic and C. Meadows. Deriving ephemeral authentication using channel axioms. In *Proc. Cambridge Workshop on Security Protocols 2009*. Springer Verlag. to appear.
19. D. Pavlovic and C. Meadows. Actor-network procedures - (extended abstract). In *ICDCIT*, LNCS vol. 7154, pp. 7–26. Springer, 2012.
20. B. Schneier. Defeating the shoe scanning machine at Heathrow Airport. Schneier on Security, December 14 2007.
21. D. J. Scott. *Abstracting application-level security policy for ubiquitous computing*. PhD thesis, University of Cambridge, 2004. UCAM-CL-TR-613 ISSN 1476-2986.
22. T. Srivatanakul. *Security Analysis with Deviational Techniques*. PhD thesis, University of York, 2005. YCST-2005-12.