Information Flow Security, Abstraction, and Composition

Fredrik Seehusen[1,2] and Ketil Stølen[1,2]
[1] SINTEF ICT, Norway
[2] University of Oslo, Norway
{ fse, kst }@sintef.no

December 9, 2008

Abstract

We present a framework that supports an incremental and modular development process of secure software systems. The framework unifies the treatment of secure information flow properties and their relationship to refinement of underspecification, translation from one level of granularity to another, and composition.

1 Introduction

We examine the relationship of *information flow security*, *abstraction*, and *composition*. Information flow security properties are requirements on the flow of information between different security domains (see e.g. [2, 4, 15, 18, 21, 22, 29, 31]). The underlying idea is that an observer residing in one security domain (call it low) shall not, based on its observations, be able to deduce whether behavior associated with another security domain (call it high) has, or has not occurred.

Abstraction and composition are two of the most important notions of software engineering. Abstraction is a means of suppressing irrelevant details. There are two orthogonal kinds of abstraction: abstraction by *translation* and abstraction by *underspecification*. Underspecification arises when a term means more than one thing and all these "things" are valid interpretations of that term. For instance, the sentence "a red car" is more concrete than the sentence "a car" because everything that is meant by the former sentence is also meant by the latter[1]. Translation is a notion that relates specifications described in one vocabulary (call it abstract) to specifications described in another vocabulary (call it concrete) that more closely coincides with the real things one wants to describe. Composition supports the notion *separation of concerns* in which a (composite) system specification is developed, not as a monolithic entity, but by putting together, or composing, other (basic) specifications.

The question we ask is: *how can we preserve secure information flow properties under refinement of underspecification, translation, and composition?*

[1] Underspecification is related to, but different than ambiguity. Ambiguity arises when a term can mean either one thing or another, but not both.

This question is of interest because it is in general desirable to analyze abstract specifications (or basic specifications) w.r.t security as opposed to concrete specifications (or composite specifications). There are three main reasons for this: (1) Analysis is more feasible at the abstract level since the concrete level may include too much detail to make analysis practical. (2) Abstract specifications tend to be more understandable or manageable than concrete specifications, hence it is easier to specify and check security requirements at the abstract levels as opposed to the concrete levels. (3) Abstract specifications are more platform independent than concrete specifications. This means that analysis results are more reusable at the abstract levels.

Items (1) and (2) are applicable for composition as well. This is because composition may result in an explosion of complexity, and because the notion of composition makes system development more manageable.

In answer to the question stated above, we present a framework for specifying secure information flow properties that are preserved under refinement of underspecification. The framework combines the modular structure of Mantel's framework [15] with the simplicity of the generalized unwinding theorem [2]. In addition we define the notions of translation and composition in a manner that is sufficiently general to capture most translations and composition operators that can be described in terms of trace-semantics, and propose theorems in answer to the question stated above.

The only work that we are aware of that unifies the treatment of information flow security and the notions of refinement of underspecification, translation, and composition is the work of Santen et.al. [8, 26, 24, 25]. However, they address probabilistic security (as opposed to possibilistic security as we do) and their notion of data refinement is less general than our notion of translation.

Apart from the work of Santen et. al. the only other work that we are aware of that considers secure information flow in relation to notions similar to translation is [5, 10]. However, Graham-Cumming and Sanders [5] considers a less general notion of security than we do, and Hutter [10] does not address refinement of underspecification.

Information flow security and refinement of underspecification has been extensively studied in the past. In 1989 it was shown by Jacob [11] that secure information flow properties in general are not preserved by the standard notion of refinement. It has later been observed that the problem originates in the inability of most specification languages to distinguish between underspecification and unpredictability[2] [8, 12, 22]. We show how secure information flow properties in general are preserved under a notion of refinement that takes the distinction of underspecification and unpredictability into consideration. A similar approach is taken in [13, 14], but we consider a more general notion of refinement.

Secure information flow and composition has been previously addressed in [2, 9, 19, 20, 30], but of these papers, only Bossi et. al. [2] considers the problem in light of a general framework of security and a general notion of composition. One of the main differences between our work and the paper by Bossi et. al. is that we consider composition in a semantic model in which the distinction of underspecification and unpredictability is made. This distinction is not made by Bossi et. al., and therefore they cannot rely on this distinction in order to

[2] Also termed *probabilistic nondeterminism* [22].

show that security is preserved by refinement of underspecification.

This paper builds on previous work by the authors [28]. However, composition was not considered in [28] and the security framework we used was based on [15]. The framework proposed in this paper is simpler and easier to work with. In summary, the main contributions of this paper are (1) a framework for specifying security information flow properties that are preserved under refinement of underspecification, (2) a characterization of translations that preserve security properties of our framework, and (3) a definition a general notion of composition and conditions under which security properties of our framework are preserved under composition.

This paper is structured as follows: In Sect. 2, we formalize a notion of systems, specifications, and refinement. In Sect. 3, we propose a framework for specifying secure information flow properties for both systems and specifications and show that the framework is sufficiently general to capture many security properties of the literature. Sections 4 and 5 formalize a notion of translation and composition and present conditions under which these notions preserve information flow properties of our framework. In Sect. 6, we discuss related work and in Sect. 7, we provide conclusions and directions of future work. Proofs of all theorems in the paper are given in the appendix.

2 Systems and Specifications

We model the input-output behavior of systems by sequences of *events* called *traces*, where an event represents the *transmission* or the *reception* of a message. Formally, an event is a pair (k, m) consisting of a kind k and a message m. An event whose kind equals ! represents the transmission of a message, whereas an event whose kind equals ? represents the reception of a message. A message is a triple (ll_1, ll_2, si) consisting of a transmitter ll_1, a receiver ll_2, and a signal si representing the message body. Both transmitters and receivers are referred to as *lifelines*, i.e., system entities such as objects or components.

The set of all events is denoted $\mathcal{E}$, and the set of all traces is denoted by $\mathcal{T}$. We require that the traces of $\mathcal{T}$ are causal in the sense that messages must be transmitted before they are received. Throughout this paper, we let e range over $\mathcal{E}$ and s, t, and u range over $\mathcal{T}$. A sequence of *events*, i.e., a trace, is written $\langle e_1, e_2, ..., e_n \rangle$. The empty trace, i.e., the trace with no events is written $\langle\rangle$.

Definition 1 (System) *The semantics of a system, denoted Φ, is a set of traces.*

In the sequel, we will for short write "system" instead "the semantics of a system" when it clear from the context what is meant.

A specification may describe possible traces that are equivalent in the sense that it is sufficient for a system to fulfill only one of them to be in compliance with the specification. We say that such traces provide *potential choices* or *underspecification*. We distinguish such choices from choices that provide *unpredictability* and that are *explicit* in sense that they must be fulfilled by a compliant system.

To distinguish between potential and explicit choices of a specification, we interpret a specification, not a trace set, but as a set of trace sets called *obligations*.

Definition 2 (Specification) *A specification, denoted* Ω*, is a set of trace sets. Each trace set in a specification is called an obligation.*

Intuitively, the traces within the same obligation may provide underspecification, while the obligations provide unpredictability in the sense that a compliant system is required to fulfill at least one trace in each obligation of a specification.

Definition 3 (Compliance) *A system* Φ *is in compliance with specification* Ω*, written* $\Omega \rightarrowtail \Phi$*, iff*

$$(t \in \Phi \implies \exists \phi \in \Omega : t \in \phi) \wedge (\phi \in \Omega \implies \exists t \in \Phi : t \in \phi)$$

A specification Ω' is a refinement of a specification Ω if all systems that are in compliance with Ω' are also in compliance with Ω. In this sense, Ω' describes its compliant systems more accurately than Ω, ergo it is at least as concrete as Ω.

Definition 4 (Refinement) *Specification* Ω' *is a refinement of specification* Ω*, written* $\Omega \rightsquigarrow \Omega'$*, iff*

$$(\forall \phi \in \Omega : \exists \phi' \in \Omega' : \phi' \subseteq \phi) \wedge (\forall \phi' \in \Omega' : \exists \phi \in \Omega : \phi' \subseteq \phi)$$

This corresponds to so-called limited refinement in STAIRS [23]. For an arbitrary obligation ϕ at the abstract level, there must be an obligation ϕ' at the concrete level such that ϕ' is a refinement of ϕ in the sense of the classic notion of refinement (see e.g., [11]). Moreover, each obligation at the concrete level must be a refinement of an obligation at the abstract level. The latter ensures that behavior that was not considered at the abstract level is not introduced at the concrete level.

The following theorem shows how the notions of refinement and compliance are related.

Theorem 1 *If specification* Ω' *is a refinement of specification* Ω *and system* Φ *is in compliance with* Ω'*, then* Φ *is in compliance with* Ω*. Formally,*

$$(\Omega \rightsquigarrow \Omega' \wedge \Omega' \rightarrowtail \Phi) \implies \Omega \rightarrowtail \Phi$$

2.1 Example of a specification

Our semantic model for specifications is based on STAIRS [6]. STAIRS provides a formal semantics for UML sequence diagrams. In this section, we explain how a UML sequence diagram can be interpreted as a set of obligations.

Fig. 1 shows an example of a UML sequence diagram. Sequence diagrams describe communication between system entities which we refer to as *lifelines*. In a diagram, lifelines are represented by vertical dashed lines. An arrow between two lifelines represents a signal being sent from one lifeline to the other in the direction of the arrow. Communication is asynchronous. Therefore, each arrow describes two events: one output event and one input event.

The diagram of Fig. 1 describes two alternative communication scenarios (these are the operands of the `xalt` construct). In the first scenario, $UserL$ stores a document on $Server$ by sending the signal *storeDocument* to it. $Server$ responds by sending an *ok* signal back to $UserL$. In the second scenario, $UserL$

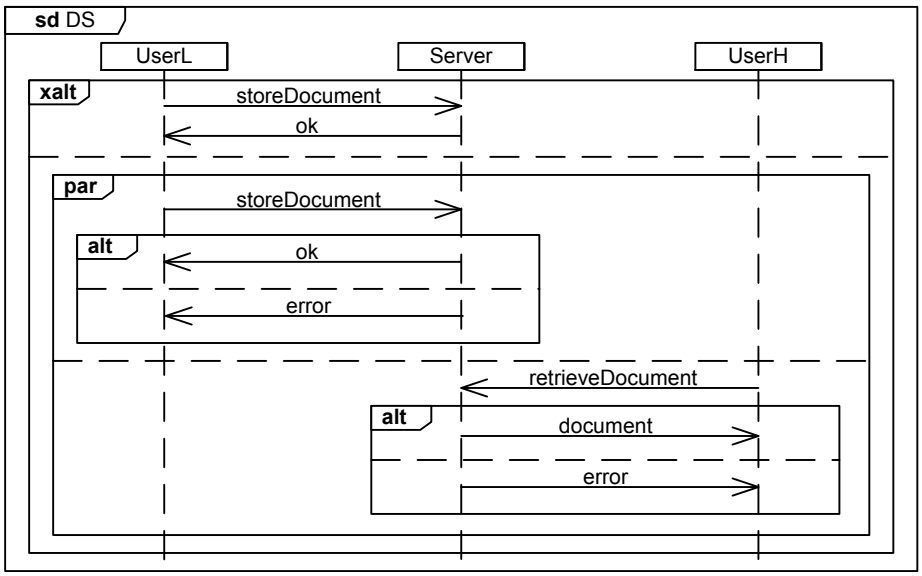

Figure 1: Document storage and retrieval.

and $UserH$ stores and retrieves a document in parallel, respectively. Since the same document cannot be stored on the server while it is being retrieved, it may be the case that a user receives an error signal instead of an ok signal.

In STAIRS, explicit choices are specified by the `xalt`-construct, whereas potential choices are specified by the `alt`-construct. This means that a system that is in compliance with the specification of Fig. 1 is required to fulfill both alternatives of the `xalt`-operator. However, a system that does not produce an error signal is in compliance with the specification, since the error signal is in a potential choice operand.

Semantically, the `xalt`-operator creates new obligations, whereas the `alt` operator collapses obligations. For instance, the semantics of the specification that describes lifeline $UserL$ of Fig. 1, written $[\![UserL]\!]$, is given by

$$[\![UserL]\!] = \{\{\langle !s, ?o \rangle\}, \{\langle !s, ?o \rangle, \langle !s, ?e \rangle\}\}$$

The two obligations correspond to the operands of the `xalt` construct of Fig. 1. Note that the last obligation contains two traces that provide potential alternatives due to the `alt` construct. For short, the signal names of Fig. 1 are referred to by their first letter, and the transmitter and the receiver of messages are suppressed. For instance, $!s$ is short for the event $(!, (UserL, Server, storeDocument))$.

The semantics of the specification DS of Fig. 1 is given by the parallel composition of the specifications that describe each of its three lifelines, i.e.,

$$[\![DS]\!] = [\![UserL]\!] \parallel [\![Server]\!] \parallel [\![UserH]\!]$$

where $\parallel$ denotes the parallel composition operator.

As described previously, traces that provide potential choices may be removed during refinement. For instance, the specification DS' of Fig. 2 is a refinement of specification DS of Fig. 1. In DS', the alternatives in which the server sends out an error message have been removed.

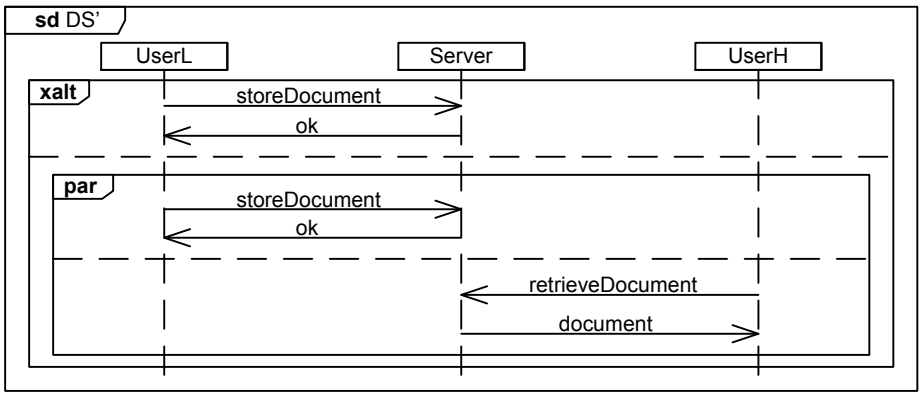

Figure 2: Document storage and retrieval (refined).

It is possible to refine specification DS' further due to the way in which the parallel operator in defined in STAIRS. That is, the scenario in which $UserL$ stores a document on the server and the scenario in which $UserH$ retrieves a document from the server, may come in any order since the two scenarios are encapsulated by the `par`-operator. In STAIRS, the order in which these scenarios occur are interpreted as potential nondeterministic choices.

The semantics of specification DS' of Fig. 2 is given by

$$\begin{gathered}[\![UserL]\!] \parallel [\![Server]\!] \parallel [\![UserH]\!] = \\ \{\langle !s, ?o\rangle\}\} \parallel \{\{\langle ?s, !o\rangle\}, \{\langle ?s, !o, ?r, !d\rangle, \langle ?s, ?r, !o, !d\rangle, \langle ?s, ?r, !d, !o\rangle, \\ \langle ?r, !d, ?s, !o\rangle, \langle ?r, ?s, !d, !o\rangle, \langle ?r, ?s, !p, !d\rangle\}\} \parallel \{\{\langle\rangle\}, \{\langle !r, ?d\rangle\}\} = \\ \{\{\langle !s, ?s, !o, ?o\rangle\}, \{\langle !s, ?s, !o, ?o, !r, ?r, !d, ?d\rangle, \langle !r, ?r, !d, ?d, !s, ?s, !o, ?o\rangle, \\ \langle !s, ?s, !r, ?r, !o, ?o, !d, ?d\rangle, \langle !r, ?r, !s, ?s, !o, ?o, !d, ?d\rangle, \dots\}\}\end{gathered} \tag{1}$$

Note that the second obligation contains the well-formed traces obtained by interleaving the traces of the operands of the `par` operator of Fig. 2.

3 Information Flow Security

Information flow security properties (in the following referred to as security properties for short) are requirements on allowed flow of information between different *security domains*. The underlying requirement is that an observer residing in one security domain (call it low) shall not, based on its observations, be able to deduce whether behavior associated with another security domain (call it high) has, or has not occurred.

In the following, we first (in Sect. 3.1) present a framework for defining security properties for systems. Then, (in Sect. 3.2) we show how standard security properties of the literature can be defined in the framework. Finally, (in Sect. 3.3) we generalize the security framework to specifications and show that all security properties defined in the framework are preserved under refinement.

3.1 A security framework for systems

In this section, we define a security framework for specifying security properties for systems. First, we present an example which illustrates the notions that underlie information flow security properties.

Example Suppose we require that $UserL$ of Fig. 1 must not deduce that $UserH$ has done something. Thus we let $UserL$ be in the low level domain, and $UserH$ be in the high level domain. Suppose further that $UserL$ can observe its own communication with the server. He can make two low level observations: $\langle !s, ?o\rangle$ and $\langle !s, ?e\rangle$. If $UserL$ observes $\langle !s, ?o\rangle$, he can not be sure that user $UserH$ has done something even if $UserL$ has complete knowledge of the specification DS of Fig. 1. This is because $UserL$ cannot exclude the fact that the topmost `xalt`-alternative of Fig. 1 (where $UserH$ is inactive) has occurred when observation $\langle !s, ?o\rangle$ is made. However, if $UserL$ observers $\langle !s, ?e\rangle$, he will know for sure that $UserH$ has done something; only the lowermost `xalt`-alternative of Fig. 1 is compatible with observation $\langle !s, ?e\rangle$, and $UserH$ is never inactive in this alternative. The specification DS is therefore not secure w.r.t. our requirement. However, the refinement DS' (Fig. 2) of DS, *is* secure w.r.t. our requirement. In this specification, only one observation can me made ($\langle\langle !s, ?o\rangle\rangle$), and, as explained above, $UserL$ cannot assert that $UserH$ has done something when observation $\langle !s, ?o\rangle$ is made. $\triangle$

The previous example suggests that we need two ingredients to define a security property:

- a notion of low level compatibility;
- a notion of high level behavior.

The notion of low level compatibility is formalized by an equivalence relation $\mathfrak{l} \subseteq \mathcal{T} \times \mathcal{T}$ on traces, and we write $s \sim_{\mathfrak{l}} t$ (i.e., $(s,t) \in \mathfrak{l}$) if the low level observation of s is compatible with the low level observation of t. The *low level equivalence class* $s^{\mathfrak{l}}$ of trace s is the set of all traces that are low level compatible with s, i.e., $s^{\mathfrak{l}} = \{t \mid s \sim_{\mathfrak{l}} t\}$.

High level behavior is similarly defined in terms of an equivalence relation $\mathfrak{h} \subseteq \mathcal{T} \times \mathcal{T}$ on traces, and we write $s \sim_{\mathfrak{h}} t$ if s and t are high level compatible, i.e., in the same high level equivalence class. Conversely, we write $s \nsim_{\mathfrak{h}} t$ (i.e., $(s,t) \notin \mathfrak{h}$) if the traces s and t are high level incompatible, i.e., not members of the same high level equivalence class.

To prevent a low level user of a system Φ from deducing that high level behavior *has* occurred, there must for each trace t and s of Φ, be a trace u in Φ such that u is both low level compatible with t and high level incompatible with s, i.e.,

$$\forall s, t \in \Phi : \exists u \in \Phi : s \nsim_{\mathfrak{h}} u \sim_{\mathfrak{l}} t \tag{2}$$

The presence of u ensures that the low level user cannot know for certain that the high level behavior class of s has occurred when observing t. Since u is low level compatible with t, the low level user cannot know whether t or u has occurred when observing t. Since, in addition, u is high level incompatible with s, the low level user cannot know for sure if the high level class of s has occurred when observing t. This is illustrated on the left hand side of Fig. 3

Figure 3: Trace u ensures that system Φ is secure w.r.t. $s^{\mathfrak{h}}$ and $t^{\mathfrak{l}}$.

Example In the previous example, we required that $UserL$ must not deduce that $UserH$ has done something. This requirement can be captured by condition (2) for appropriate definitions of $\mathfrak{l}$ and $\mathfrak{h}$.

We define the low level compatibility relation in terms of a set $\mathcal{L}$ of low level events that can be observed by the low level user. In the example, $UserL$ can observe its own communication with the server. Therefore, we define $\mathcal{L}$ to be set of all events that can occur on the lifeline $UserL$, i.e., $\mathcal{L} = \{!s, ?o, ?e\}$. Low level compatibility is then defined such that two trace are low level compatible if they contain the same low level observations:

$$s \sim_{\mathfrak{l}} t \Leftrightarrow s|_{\mathcal{L}} = t|_{\mathcal{L}} \tag{3}$$

where $s|_{\mathcal{L}}$, the projection of s to $\mathcal{L}$, yields the trace obtained from s by removing all events that are not in the set of events $\mathcal{L}$.

Since $UserL$ must not deduce that $UserH$ has done something, the high level equivalence class of interest should characterize all traces that contain an event occurring on the lifeline of $UserH$. We let $\mathcal{H}$ denote all events that can occur on the lifeline $UserH$, i.e., $\mathcal{H} = \{!r, ?d, ?e\}$, and define the high level equivalence relation $\mathfrak{h}$ by

$$s \sim_{\mathfrak{h}} t \Leftrightarrow s|_{\mathcal{H}} \neq \langle\rangle \wedge t|_{\mathcal{H}} \neq \langle\rangle \tag{4}$$

To obtain the high level incompatibility relation, we simply take the complement of $s \sim_{\mathfrak{h}} t$, i.e.,

$$s \nsim_{\mathfrak{h}} t \Leftrightarrow s|_{\mathcal{H}} = \langle\rangle \vee t|_{\mathcal{H}} = \langle\rangle \tag{5}$$

In the previous example, we claimed that specification DS of Fig. 1 was not secure w.r.t. out requirement. We now explain the reason for this by showing that a system Φ_{DS} that is compliant with DS does not satisfy condition (2).

Let Φ_{DS} be defined by

$$\Phi_{DS} \triangleq \{\langle !s, ?s, !o, ?o\rangle, \langle !s, ?s, !o, ?o, !r, ?r, !d, ?d\rangle, \langle !r, ?r, !s, ?s, !e, ?e, !d, ?d\rangle\} \tag{6}$$

Choose traces s and t in Φ_{DS} such that $s = t = \langle !r, ?r, !s, ?s, !e, ?e, !d, ?d\rangle$. By condition (2), there must be trace u in Φ_{DS} such that u is low level compatible with t and high level incompatible with s. However, no such u exists. The only trace in Φ_{DS} that is low level compatible with t is t itself, i.e., $t^{\mathfrak{l}} \cap \Phi_{DS} = \{t\}$, and t is not high level incompatible with s. Therefore system Φ_{DS} is not secure.

In the previous example, we also claimed that the specification DS' of Fig. 2 was secure w.r.t. our requirement. To see why a system $\Phi_{DS'}$ which is compliant with DS' is secure w.r.t. our requirement, define $\Phi_{DS'}$ by

$$\Phi_{DS'} \triangleq \{\langle !s, ?s, !o, ?o\rangle, \langle !s, ?s, !o, ?o, !r, ?r, !d, ?d\rangle\} \tag{7}$$

Regardless of how we chose s and t in $\Phi_{DS'}$, there is always a trace u in $\Phi_{DS'}$ which is low level compatible with t and high level incompatible with s. Therefore system $\Phi_{DS'}$ is secure w.r.t. our requirement. $\triangle$

As indicated in the beginning of this section, information also flows from the high level domain to the low level domain if the low level observer deduces that a high level behavior has *not* occurred. To prevent this kind of information flow for a system Φ, there must for each trace t and s of Φ, be a trace u in Φ such that u is both low level compatible with t and high level compatible with s, i.e.,

$$\forall s, t \in \Phi : \exists u \in \Phi : s \sim_{\mathfrak{h}} u \sim_{\mathfrak{l}} t \tag{8}$$

The presence of u ensures that the low level user cannot deduce that the high level behavior class of s has *not* occurred. This is pictured on the right hand side of Fig. 3.

Example Let system Φ_{DS} be given by (6) of the previous example. Suppose that we want to prevent $UserL$ from deducing that $UserH$ has *not* done something. This requirement is formalized by condition (8) by letting $\mathfrak{l}$ and $\mathfrak{h}$ be the low level and high level equivalence relations defined by (3) and (4) in the previous example, respectively. The system Φ_{DS} *is* secure w.r.t. this requirement; regardless of the choice of s and t in Φ_{DS}, there is always a trace u in Φ_{DS} that is low level equivalent to t and high level compatible with s. $\triangle$

Both condition (2) and condition (8) may be seen as schemas that are parameterized by $\mathfrak{h}$ and $\mathfrak{l}$ for defining security properties. The conditions are related in the sense that a system Φ satisfies condition (8) for some $\mathfrak{h}$ and $\mathfrak{l}$ if and only if Φ satisfies condition (2) for the complement $\overline{\mathfrak{h}}$ (i.e., $\overline{\mathfrak{h}} = \{(s,t) \,|\, (s,t) \notin \mathfrak{h}\}$) and $\mathfrak{l}$. This means that condition (8) can be expressed in terms of condition (2) and vice versa. It is convenient to work with one condition instead of two. Therefore, we henceforth only work with conditions of the form (8). We now let $\mathfrak{h}$ denote either an equivalence relation, or the complement of an equivalence relation and write $s \xrightarrow{\mathfrak{h}} t$ instead of $s \sim_{\mathfrak{h}} t$ or $s \nsim_{\mathfrak{h}} t$ to highlight this.

Many information flow properties of the literature may be expressed as conjunctions of instantiations of condition (8), but not all. The reason for this is that the condition $s \xrightarrow{\mathfrak{h}} u \sim_{\mathfrak{l}} t$ should sometimes only hold for *some* traces s and t, but not for *all*. We therefore generalize the condition by introducing a new relation $\mathfrak{r}$, the *restriction relation*, and call the new condition a *basic security predicate schema*.

Definition 5 (Basic security predicate schema for systems) *The basic security predicate* $\textsc{Bsp}_{\mathfrak{r}\mathfrak{h}\mathfrak{l}}(\Phi)$ *for restriction relation* $\mathfrak{r}$*, high level relation* $\mathfrak{h}$*, and low level equivalence relation* $\mathfrak{l}$ *holds iff*

$$\forall s, t \in \Phi : s \xrightarrow{\mathfrak{r}} t \implies \exists u \in \Phi : s \xrightarrow{\mathfrak{h}} u \sim_{\mathfrak{l}} t$$

The name basic security predicate is taken from [15] where it is shown that many security properties of the literature can be expresses as conjunctions of basic security predicates.

Definition 6 (Security property) *A security property* Sp *is a conjunction of basic security predicates.*

Table 1: Basic security predicates of Mantel's schema

Name	R	Q
RE'	TRUE	$u\|_{\mathcal{H}} = \langle\rangle$
RI'	TRUE	$u\|_{\mathcal{HI}} = \langle\rangle$
IHAI'	$e \in \mathcal{HI} \wedge t = \beta \frown \alpha \wedge \alpha\|_{\mathcal{HI}} = \langle\rangle \wedge \gamma' \frown \langle e\rangle \in \Phi \wedge \gamma'\|_{\mathcal{HI}} = \beta\|_{\mathcal{HI}}$	$u = \beta' \frown \langle e\rangle \frown \alpha' \wedge \beta'\|_{\mathcal{L}\cup\mathcal{HI}} = \beta\|_{\mathcal{L}\cup\mathcal{HI}} \wedge \alpha'\|_{\mathcal{L}\cup\mathcal{HI}} = \alpha\|_{\mathcal{L}\cup\mathcal{HI}}$
IAE'	$e \in \mathcal{H} \wedge t = \beta \frown \alpha \wedge \alpha\|_{\mathcal{H}} = \langle\rangle \wedge \beta \frown \langle e\rangle \in \Phi$	$u = \beta \frown \langle e\rangle \frown \alpha$
* The traces α', β', and γ' are existentially quantified over $\mathcal{T}$		

3.2 Security properties for systems

In this section, we define the security properties *non-inference* NF [21], *generalized non-inference* GNF [20], *generalized non-interference* GNI [18], and *the perfect security property* PSP [31] in our framework.

The idea of defining security properties as a conjunction of basic security predicates was introduced by Mantel in [15]. For this reason, we will first present the above mentioned security properties as they are defined by Mantel's framework. Then we present the corresponding definitions in our framework.

In Mantel's framework, a basic security predicate is given by the following definition.

Definition 7 (Mantel's basic security predicate schema) *System Φ satisfies the basic security predicate* BSP'$_{RQ}(\Phi)$ *for restriction R and closure requirement Q iff*

$$\forall t \in \Phi, \alpha, \beta \in \mathcal{T}, e \in \mathcal{E} :$$
$$(R(\Phi, t, \alpha, \beta, e) \implies \exists u \in LLES(\Phi, t) : Q(t, u, \alpha, \beta, e))$$

where $LLES$, the so-called low level equivalence set, is defined by

$$LLES(\Phi, t) \triangleq t^{\mathsf{I}} \cap \Phi$$

We make some initial assumptions before we present the basic security predicates of Mantel's schema that are needed to define the properties NF, GNF, GNI, and PSP. First, we assume that there is a set of low level events $\mathcal{L}$ that can be observed by the low level user, and a set $\mathcal{H}$ disjoint from $\mathcal{L}$ of high level events that are to be considered as confidential. The subset $\mathcal{HI}$ of $\mathcal{H}$ will denote the set of all high level input events. For simplicity, we shall assume that the set of all events $\mathcal{E}$ equals $\mathcal{L} \cup \mathcal{H}$.

Throughout this section we assume a fixed low level equivalence relation defined

$$s \sim_{\mathsf{I}} t \Leftrightarrow s|_{\mathcal{L}} = t|_{\mathcal{L}} \tag{9}$$

and we assume that all systems Φ are prefix-closed[3] (this assumption is made in [15].)

Table 1 shows the four basic security predicates of Mantel's schema that are needed to define the security properties. These are RE' (Removal of Events), RI' (Removal of Inputs), IHAI' (Insertion of High level Admissible Inputs), and

[3] Φ is prefix closed if $t \in \Phi \wedge s \sqsubseteq t \implies s \in \Phi$.

Table 2: Basic security predicates of our schema

Name	$\mathfrak{r}$	$\mathfrak{h}$
RE	TRUE	$s\|_{\mathcal{H}} = \langle\rangle \vee u\|_{\mathcal{H}} = \langle\rangle$
RI	TRUE	$s\|_{\mathcal{HI}} = \langle\rangle \vee u\|_{\mathcal{HI}} = \langle\rangle$
IHAI	$t = \beta \frown \alpha \wedge \alpha\|_{\mathcal{HI}} = \langle\rangle \wedge s = \gamma' \frown \langle e'\rangle \wedge \gamma'\|_{\mathcal{HI}} = \beta\|_{\mathcal{HI}}$	$s\|_{\mathcal{HI}} = u\|_{\mathcal{HI}}$
IAE	$t = \beta \frown \alpha \wedge \alpha\|_{\mathcal{H}} = \langle\rangle \wedge s = \beta \frown \langle e\rangle$	$h(s) = h(u)$
* The traces α, β and γ' is existentially quantified over $\mathcal{T}$, e' is existentially quantified over $\mathcal{HI}$, e is existentially quantified over $\mathcal{H}$, and α' is existentially quantified over $\mathcal{L}^*$.		

IAE' (Insertion of Admissible Events). See [15] for an explanation of these. In Table 1, *Name* refers to the name of the basic security predicate and R and Q refer to the restriction and the closure requirement that are used to instantiate the schema. Variables that are not quantified in the table refer to the variables of Def. 7. For instance, the definition of the basic security predicate RE' is

$$\text{RE'}(\Phi) \triangleq \forall t \in \Phi, \alpha, \beta \in \mathcal{T}, e \in \mathcal{E} : \text{TRUE} \implies \exists u \in LLES(\Phi, t) : u|_{\mathcal{H}} = \langle\rangle$$

Due to [15], we know that the following Theorem holds.

Theorem 2 ([15]) *The following equivalences hold:*

$$\begin{array}{lcl} \text{GNF}(\Phi) & \Leftrightarrow & \text{RI'}(\Phi) \\ \text{NF}(\Phi) & \Leftrightarrow & \text{RE'}(\Phi) \\ \text{GNI}(\Phi) & \Leftrightarrow & (\text{RI'}(\Phi) \wedge \text{IHAI'}(\Phi)) \\ \text{PSP}(\Phi) & \Leftrightarrow & (\text{RE'}(\Phi) \wedge \text{IAE'}(\Phi)) \end{array}$$

The NF property ensures that the low level user cannot deduce that a high level event has occurred. The GNF property is similar except that it treats high level *input* events as confidential as opposed to all high level events. As explained in [15], GNI demands that any interleaving of the high level input of one trace with the low level behavior of another trace can be made a possible trace by adapting the outputs. The PSP property ensures the low level user cannot deduce that a high level event has occurred (the RE part) or that any high level event which is not influenced by the low level user has not occurred (the IAE part).

Table 2 shows the corresponding basic security predicates of our schema. Variables that are not quantified in the table refer to the variables of Def. 5. For instance, the definition of the basic security predicate RE is

$$\text{RE}(\Phi) \triangleq \forall s, t \in \Phi : \text{TRUE} \implies \exists u \in \Phi : (s|_{\mathcal{H}} = \langle\rangle \vee u|_{\mathcal{H}} = \langle\rangle) \wedge u \sim_{\mathfrak{l}} t$$

In Table 2, the high level relation $\mathfrak{h}$ for RE and RE' may be seen as the complement of the equivalence relation defining the set of all traces that contain a high level event or a high level input event, respectively. Note that the definition of $\mathfrak{h}$ in IAE makes use of the function $h \in \mathcal{T} \to \mathcal{T}$ which yields the high level behavior of a given trace. More precisely, $h(t)$ yields the trace obtained from t by replacing all low level events by the dummy event $\surd$, and removing all dummy-events occurring after the last high level event in t. If t does not

contain any high level events, then $h(t) = \langle\rangle$. For example, for $e_l, e'_l \in \mathcal{L}$, and $e_h, e'_h \in \mathcal{H}$, we have

$$h(\langle e_l, e_h, e'_h, e'_l, e_l, e_h, e_l\rangle) = \langle\surd, e_h, e'_h, \surd, \surd, e_h\rangle$$

Intuitively, $h(s) = h(u)$ holds when s and t contain the same timing sensitive sequences of high level events.

Theorem 3 *The following equivalences hold:*

$$\begin{array}{lcl} \textsc{Re}(\Phi) & \Leftrightarrow & \textsc{Re'}(\Phi) \\ \textsc{Ri}(\Phi) & \Leftrightarrow & \textsc{Ri'}(\Phi) \\ \textsc{Ihai}(\Phi) & \Leftrightarrow & \textsc{Ihai'}(\Phi) \\ \textsc{Iae}(\Phi) & \Leftrightarrow & \textsc{Iae'}(\Phi) \end{array}$$

3.3 A security framework for specifications

In this section, we define a security framework for specifications. This allows us to exploit the distinction of potential and explicit choices to define security properties that are preserved under refinement. Closing security properties under refinement without making this distinction would make the properties too strong to be useful. To see this, consider the the standard notion of refinement by underspecification which has previously been used in investigating the relationship of information flow security and refinement (see e.g., [11]). It states that a set of traces Φ' is a refinement of a trace set Φ iff

$$\Phi' \subseteq \Phi \tag{10}$$

The reason why secure information flow properties are not preserved by this notion refinement of underspecification becomes apparent when one considers again the manner in which these properties are defined (see Def. 5); Φ is secure if for each of its traces s and t satisfying the restriction $\mathfrak{r}$, there is a trace u in Φ whose presence ensures that s and t do not compromise security. However, by (10) there is no guarantee that a refinement of Φ will include those traces that ensure the security of Φ, hence secure information flow properties are in general not preserved by refinement.

Intuitively, the cause of this problem is that security properties depend on unpredictability. For instance, the strength of one's password may be measured in terms of how hard it is for an attacker to guess the password one has chosen. The demand for the presence of the traces u may be seen as the security predicate's requirement of unpredictability, but traces that provide this unpredictability may be removed during refinement.

Clearly, closing security properties under the standard notion of refinement would be make the properties too strong to be useful. This motivates a redefinition of the basic security schema which takes into account the distinction between underspecification and unpredictability.

Definition 8 (Basic security predicate schema for specifications) *The basic security predicate* $\textsc{Bsp}_{\mathfrak{r}\mathfrak{h}\mathfrak{l}}(\Omega)$ *for restriction relation* $\mathfrak{r}$*, high level relation* $\mathfrak{h}$*,*

and low level equivalence relation $\mathfrak{l}$ *holds iff*[4]

$$\forall s, t \in \widehat{\Omega} : s \xrightarrow{\mathfrak{r}} t \implies \exists \phi \in \Omega : \{s\} \xrightarrow{\mathfrak{h}} \phi \sim_{\mathfrak{l}} \{t\}$$

Instead of demanding that there is a trace u that satisfies some criterion, we demand that there is an *obligation* ϕ such that all its traces satisfy that criterion. The intuition of (Def. 8) is that obligations, as opposed to individual traces, are understood as providing the unpredictability required by instances of the schema. It follows from this that the following theorem holds.

Theorem 4 $\textsc{Bsp}_{\mathfrak{r}\mathfrak{h}\mathfrak{l}}$ *is preserved by refinement for arbitrary restriction relation* $\mathfrak{r}$*, high level relation* $\mathfrak{h}$*, and low level equivalence relation* $\mathfrak{l}$*:*

$$\Omega \rightsquigarrow \Omega' \wedge \textsc{Bsp}_{\mathfrak{r}\mathfrak{h}\mathfrak{l}}(\Omega) \implies \textsc{Bsp}_{\mathfrak{r}\mathfrak{h}\mathfrak{l}}(\Omega')$$

The basic security predicate schema for specifications is a generalization of the basic security schema for systems in the sense of the following theorem.

Theorem 5 *Let* Φ *be a system, and let* Ω *be the specification obtained from* Φ *by creating a new obligation for each trace in* Φ*, i.e.,* $\Omega = \{\{t\} \mid t \in \Phi\}$*. Then* Ω *satisfies the basic security predicate* $\textsc{Bsp}_{\mathfrak{r}\mathfrak{h}\mathfrak{l}}$ *(of Def. 8) if and only if* Φ *satisfies* $\textsc{Bsp}_{\mathfrak{r}\mathfrak{h}\mathfrak{l}}$ *(of Def. 5), i.e.,*

$$\textsc{Bsp}_{\mathfrak{r}\mathfrak{h}\mathfrak{l}}(\Phi) \Leftrightarrow \textsc{Bsp}_{\mathfrak{r}\mathfrak{h}\mathfrak{l}}(\Omega)$$

Since a security property is a conjunction of basic security predicates and all security predicates are preserved under refinement, then clearly all security properties are preserved under refinement as well.

Corollary 1 *All security properties are preserved under refinement.*

Example In Sect. 3.2, we formalized the requirement that $UserL$ must not deduce that $UserH$ has done something by instantiating condition (2) by the equivalence relation $\mathfrak{l}$ and $\mathfrak{h}$ defined by (3) and (4), respectively. This condition is in fact the NF property as defined in Sect. 3.2. The specification DS' of Fig. 2 is secure w.r.t. the NF property defined for specifications when the set $\mathcal{L}$ of low level events is defined as all the events that can occur on the lifeline of $UserL$ and the set $\mathcal{H}$ of high level events is defined as all the event that can occur on lifeline $UserH$. To see this, recall the semantics of specification DS' of Fig. 2 as given by (1) in Sect. 2.1. The low level user can make the single observation $\langle !s, ?o\rangle$. Since $[\![DS']\!]$ contains the obligation $\{\langle !s, ?s, !o, ?o\rangle\}$ whose trace is both low level compatible with the observation and contains no high level events, $UserL$ cannot deduce with certainty that $UserH$ has done something when observing $\langle !s, ?o\rangle$. The specification DS' can be refined further, for instance to ensure that server replies to the users immediately after a request has been received. By Theorem. 4, we know that any such refinement is secure w.r.t. the NF-property.

We note that it is also the case that the specifications $[\![UserL]\!]$, $[\![Server]\!]$, and $[\![UserH]\!]$ of Fig. 2 are secure w.r.t. the NF-property. △

[4]The operator $\widehat{\ }$ collapses a set of obligations into a set of traces, i.e. $\widehat{\Omega} = \bigcup_{\phi \in \Omega} \phi$. Also note that for trace sets A and B and relation some relation $\mathfrak{c}$ on traces, we write $A \xrightarrow{\mathfrak{c}} B$ iff $a \xrightarrow{\mathfrak{c}} b$ for all a in A and b in B.

4 Translation

The vocabulary that is used in describing a system is always an abstraction of the real things one wants to describe. The more closely the vocabulary coincides with these things, the more concrete it is. For example, the behavior of a human may be described in terms of a vocabulary consisting sequences of actions such as eat, sleep, walk and so on. The behavior of the same human may also be described by sequences of actions performed by a collection of cells. The notion of *translation* essentially relates specifications described in one vocabulary to specifications described in another vocabulary of a finer granularity.

We model the abstract vocabulary by a set $\mathcal{T}$ of abstract traces, and the concrete vocabulary by a set $\mathcal{T}'$ of concrete traces. Attention is henceforth restricted to translation functions[5] $f \in \mathcal{T} \to (\mathbb{P}(\mathcal{T}') \setminus \emptyset)$ mapping abstract traces to sets of concrete traces and translations indirectly built from such. A translation function f is lifted to trace sets as follows

$$f(\phi) = \{t' \in f(t) \,|\, t \in \phi\}$$

Definition 9 (Translation) *A translation is a set F of translation functions. The (concrete) specification $F(\Omega)$ obtained by applying F to (abstract) specification Ω is defined*

$$F(\Omega) \triangleq \{f(\phi) \,|\, \phi \in \Omega \wedge f \in F\}$$

We use sets of translation functions on traces to model the fact that one abstract obligation may be translated to several concrete obligations. Technically, this is more convenient than using an obligation set valued function on obligations.

The aim of this section is to characterize translations that allow us to exploit the security of the abstract level in establishing the security at the concrete level.

The abstract specification Ω and its translation $F(\Omega)$ may be in different semantic domains; $F(\Omega)$ may include particularities that are not present in Ω. Consequently, the security requirement at the abstract level may be different than the security requirement at the concrete level since the requirement at the concrete level must take these particularities into account. We therefore distinguish between abstract security requirements and concrete security requirements. More formally, we let the abstract security requirement (denoted A) be a basic security predicate that is instantiated by some fixed but arbitrary relations $\mathfrak{r}$, $\mathfrak{h}$, and $\mathfrak{l}$ on $\mathcal{T}$. Similarly, we let the concrete security requirement (denoted C) be a basic security predicate that is instantiated by some fixed but arbitrary relations $\mathfrak{r}'$, $\mathfrak{h}'$, and $\mathfrak{l}'$ on $\mathcal{T}'$. We have

$$\mathrm{A} \triangleq \textsc{Bsp}_{\mathfrak{r}\mathfrak{h}\mathfrak{l}} \qquad \mathrm{C} \triangleq \textsc{Bsp}_{\mathfrak{r}'\mathfrak{h}'\mathfrak{l}'} \tag{11}$$

We are interested in conditions on transformations that allow us to exploit the fact that Ω is secure w.r.t. A in order to show that $F(\Omega)$ is secure w.r.t. to C. What does this mean? Consider the diagram below[6].

$$\begin{array}{ccccc} s & \xrightarrow{\mathfrak{r}} & t & \overset{a}{\Longrightarrow} & \{s\} \xrightarrow{\mathfrak{h}} \phi \sim_{\mathfrak{l}} \{t\} \\ \Big\Uparrow x & & & & \Big\Downarrow y \\ s' & \xrightarrow{\mathfrak{r}'} & t' & \overset{c}{\Longrightarrow} & \{s'\} \xrightarrow{\mathfrak{h}'} \phi' \sim_{\mathfrak{l}'} \{t'\} \end{array}$$

[5] $\mathbb{P}(A)$, the *power-set* of set A is defined $\{B \,|\, B \subseteq A\}$.

[6] Note that the diagram is informal; it is only meant as an illustration.

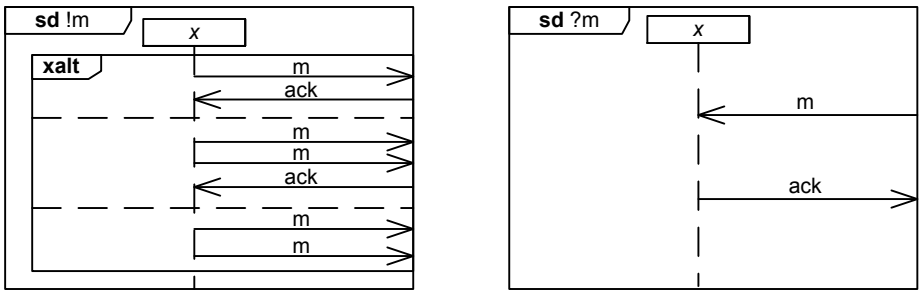

Figure 4: Translation of A level events to the C level.

By Def. 8, the implication labeled a holds for all $s, t \in \widehat{\Omega}$ if we know that Ω is secure w.r.t. A. Showing that $F(\Omega)$ is secure w.r.t. C corresponds to showing that the implication labeled c holds for traces in $F(\Omega)$. Exploiting the fact that security has been established at the abstract level therefore means that one takes advantage of the implication a in order to show c. We can do this by showing that the implications labeled x and y hold, because if x, a, and y hold, then c holds by transitivity. This results in the following theorem.

Theorem 6 *Specification $F(\Omega)$ is secure w.r.t.* $\textsc{Bsp}_{\mathfrak{r}'\mathfrak{h}'\mathfrak{l}'}$ *if Ω is secure w.r.t.* $\textsc{Bsp}_{\mathfrak{r}\mathfrak{h}\mathfrak{l}}$ *and F is a translation that satisfies the following conditions for all $f_1, f_2 \in F$, $s, t \in \widehat{\Omega}$, $\phi \in \Omega$, $s' \in f_1(s)$, and $t' \in f_2(t)$*

$$s' \xrightarrow{\mathfrak{r}'} t' \implies s \xrightarrow{\mathfrak{r}} t$$

$$\{s\} \xrightarrow{\mathfrak{h}} \phi \sim_{\mathfrak{l}} \{t\} \implies \exists f \in F : \{s'\} \xrightarrow{\mathfrak{h}'} f(\phi) \sim_{\mathfrak{l}'} \{t'\}$$

4.1 Example of translation

Preservation of security under translation enables security to be established for an abstract specification without having to reestablish security for its concrete translation. This is desirable since analyzing an abstract specification is usually cheaper than analyzing its more detailed translation. Another reason is that abstract specifications are more platform independent than concrete specifications. This means that analysis results can be reused for abstract specifications that are translated to several different platforms. Since all security properties defined in framework are preserved under refinement, our approach offers the benefit that security is preserved under a series of successive translation and refinement steps.

In this section, we give an example of a translation and we exploit Theorem 6 to show that the translation preserves the NF-property.

Consider again Fig. 2. Imagine that the communication between the users and the server is based on a high level communication protocol. In practice, high level communication protocols are bound to lower level protocols that handle issues such as message disappearance, message overtaking, message fragmentation, and error correction.

The process of binding a high level communication protocol (call it A) to a lower-level protocol (call it C) can be described by a translation that takes a

specification at the A level as input and yields a specification at the C level as output.

We give an example of a lower-level protocol C which can be used to handle message disappearance. The C protocol is governed by the following rules: (1) The reception of each message is explicitly acknowledged by the receiver, i.e., the receiver sends an acknowledgement message (ack) back to the transmitter to indicate that the message has been received. (2) If the transmitter of a message does not receive an acknowledgement, then the message is retransmitted. (3) The transmitter of a message gives up after transmitting the same message twice.

In Fig. 4, we have illustrated what an output event at the A level (on the left hand side of Fig. 4) and an input event at the A level (on the right hand side of Fig. 4) corresponds to at the C level. Note that the lifeline x in the figure represents an arbitrary lifeline. For output events, there are three alternatives. The first (upper most) alternative describes the case in which a message is transmitted whereupon an acknowledgement (ack) is received. In the second alternative, no acknowledgement is received after the message has been transmitted once. The message is therefore retransmitted, whereupon an acknowledgement is received. In the third alternative, the sender transmits the message twice and gives up because no acknowledgement is received. As further illustrated on the right hand side of Fig. 4, input events at the A level correspond to one possible scenario at the C level in which an explicit acknowledgement is sent after a message has been received.

A translation from a specification Ω at the A level to a specification at the C level replaces all events in Ω by their corresponding behavior at the C level. In the following we formalize this translation for specifications describing the behavior of a *single* lifeline. When then show that the translation preserves the NF-property. Later, in Sect. 5.1, we show how to formalize the translation for specifications consisting of more than one lifeline.

First, we formalize the translation of an output event at the A level to its corresponding behavior at the C level. Because output events at the C level correspond to three different alternatives that are specified as *explicit nondeterministic* choices (due to the `xalt`-operator), we need three functions, oe_1, oe_2, and oe_3, one function for each alternative. These functions are defined by

$$\begin{array}{lcl} oe_1(!m) & \triangleq & \{\langle !m, ?ack\rangle\} \\ oe_2(!m) & \triangleq & \{\langle !m, !m, ?ack\rangle\} \\ oe_3(!m) & \triangleq & \{\langle !m, !m\rangle\} \end{array}$$

The translation of input events is defined by the function ie (only one function is needed since an input event only corresponds to one alternative at the C level) as follows

$$ie(?m) \quad \triangleq \quad \{\langle ?m, !ack\rangle\}$$

We let $a2c_\rho$ be the translation function that takes a trace at the A level, and applies ie to all input event of the trace, and oe_1, oe_2, or oe_3 to each output event of the trace in the order given by the infinite sequence ρ over the set $\{1, 2, 3\}$ (representing the three alternative output event scenarios at the C

level). Formally[7],

$$\begin{array}{lcl} a2c_\rho(\langle\rangle) & \triangleq & \{\langle\rangle\} \\ a2c_{i\rho}(\langle !m\rangle \frown s) & \triangleq & oe_i(!m) \frown a2c_\rho(s) \\ a2c_\rho(\langle ?m\rangle \frown s) & \triangleq & ie(?m) \frown a2c_\rho(s) \end{array}$$

The translation of a single lifeline specification at the A level to the C level, is given by the set of translation functions $A2C$ defined as follows

$$A2C \triangleq \{a2c_\rho \,|\, \rho \in \{1,2,3\}^\infty\}$$

where B^∞ yields the set of all infinite sequences of elements in the set B.

Let Ω denote $[\![UserL]\!]$, $[\![Server]\!]$, or $[\![UserH]\!]$ of Fig. 2. To show why the translation $A2C$ preserves the NF-property for a specification Ω, we instantiate the conditions of Theorem 6 according to the definition of NF. For simplicity, we define NF as the basic security predicate instantiated by the restriction relation r, high level relation $\mathfrak{h}$, and low level relation $\mathfrak{l}$ defined by

$$s \xrightarrow{\mathfrak{r}} t \Leftrightarrow \text{TRUE} \qquad s \xrightarrow{\mathfrak{h}} t \Leftrightarrow t|_{\mathcal{H}} = \langle\rangle \qquad s \sim_{\mathfrak{l}} t \Leftrightarrow s|_{\mathcal{L}} = t|_{\mathcal{L}}$$

This definition of NF is equivalent to the definition given in Sect. 3.2, but the definition of $\mathfrak{h}$ given here simplifies the instantiation of the conditions of Theorem 6. To instantiate the conditions, we lift the projection operator to trace sets such that $\Phi|_E$ yields the set obtained from Φ by projecting each trace in Φ to E, i.e., $\Phi|_E = \{t|_E \,|\, t \in \Phi\}$. We get

$$\begin{array}{c} \phi|_{\mathcal{H}} = \{\langle\rangle\} \wedge \phi|_{\mathcal{L}} = \{t\}|_{\mathcal{L}} \implies \\ \exists a2c' \in A2C : a2c'(\phi)|_{\mathcal{H}} = \{\langle\rangle\} \wedge a2c'(\phi)|_{\mathcal{L}} = \{t'\}|_{\mathcal{L}} \end{array} \tag{12}$$

for all t in $\widehat{\Omega}$, ϕ in Ω, $a2c$ in $A2C$, and t' in $a2c(t)$.

Note that by definition of the restriction $\mathfrak{r}$ for the NF-property, the upper most condition of Theorem. 6 is trivally satisfied for all translations. Thus this condition can be ignored.

The first part of condition (12) requires that an obligation at the A level which does not contain any high level events, is not translated to an obligation at the C level that contains a high level event. This condition is satisfied by any $A2C$ because the set $\mathcal{H}$ of high level events is assumed to be *all* events that can occur on a single lifeline. Translation $A2C$ cannot change the lifeline that an event occurs on. It follows from this that condition (12) is reduced to

$$\phi|_{\mathcal{L}} = \{t\}|_{\mathcal{L}} \implies \exists a2c' \in A2C : a2c'(\phi)|_{\mathcal{L}} = a2c(t)|_{\mathcal{L}} \tag{13}$$

for all t in $\widehat{\Omega}$, ϕ in Ω, and $a2c$ in $A2C$.

This condition holds because the set $\mathcal{L}$ of low level events is assumed to be all events that can occur on lifeline $UserL$. Since we only consider translation of single lifeline specifications in this section, this means that we either have $\phi|_{\mathcal{L}} = \{\langle\rangle\}$ and $\{t\}|_{\mathcal{L}} = \{\langle\rangle\}$ in the case where Ω denotes $[\![Server]\!]$ or $[\![UserH]\!]$, or we have $\phi|_{\mathcal{L}} = \{t\}|_{\mathcal{L}} \Leftrightarrow \phi = \{t\}$ in the case where Ω denotes $[\![UserL]\!]$.

[7]The operator $\frown$ yields the concatenation of two sequences. The operator is lifted to sets of sequences such that $A \frown B$ yields the set obtained by concatenating all sequences in A by all sequences in B.

In either case, it is easy to see that condition (13) holds. Thus we know by Theorem 6 and definition of NF-property, that the translation $A2C$ preserves the NF-property.

When the translation $A2C$ is applied to, say the specification $[\![Server]\!]$, all events of the specification are replaced by their corresponding behavior at the C level. It is still possible to refine $A2C([\![Sever]\!])$ by removing underspecification, for instance to ensure that all reception messages are *immediately* followed by an acknowledgement. However, this refinement will not render the specification insecure w.r.t. the NF property since all refinements are security preserving.

5 Composition

In any reasonable specification language, a (composite) specification may be obtained by putting together, or as we shall say, *composing*, (basic) specifications. Standard composition operators include parallel composition, sequential composition, and nondeterministic choice. However, we do not restrict attention to any particular operator. Composition is instead considered in the abstract, and defined in terms of binary operators $\diamond \in \mathcal{T} \times \mathcal{T} \to (\mathbb{P}(\mathcal{T}) \setminus \emptyset)$ on traces. Any trace operator $\diamond$ is lifted to trace sets as follows

$$\phi_1 \diamond \phi_2 \triangleq \{s \in (s_1 \diamond s_2) \mid s_1 \in \phi_1 \wedge s_2 \in \phi_2\}$$

Definition 10 (Composition operator) *A composition operator is a set $\diamond$ of trace operators. The composition of specification Ω_1 and specification Ω_2, written $\Omega_1 \diamond \Omega_2$, is defined*

$$\Omega_1 \diamond \Omega_2 \triangleq \{\phi_1 \diamond \phi_2 \mid \phi_1 \in \Omega_1 \wedge \phi_2 \in \Omega_2 \wedge \diamond \in \diamond\}$$

The definition captures standard binary composition operators.

Example The operators which are used in STAIRS for specifying potential nondeterministic choice (`alt`), explicit nondeterministic choice (`xalt`), and parallel composition (`par`) are defined by

$$\begin{array}{lcl} [\![d_1 \texttt{alt} d_2]\!] & \triangleq & \{\phi_1 \cup \phi_2 \mid \phi_1 \in [\![d_1]\!] \wedge \phi_2 \in [\![d_2]\!]\} \\ [\![d_1 \texttt{xalt} d_2]\!] & \triangleq & [\![d_1]\!] \cup [\![d_2]\!] \\ [\![d_1 \texttt{par} d_2]\!] & \triangleq & \{\phi_1 \parallel \phi_2 \mid \phi_1 \in [\![d_1]\!] \wedge \phi_2 \in [\![d_2]\!]\} \end{array}$$

where $\phi_1 \parallel \phi_2$ yields all well-formed interleavings of all traces in ϕ_1 and ϕ_2 (see [7]).

All these definitions can be captured by our notion of composition operator (Def. 10). We define potential nondeterministic choice by operator $\diamond_{alt} \triangleq \{\diamond_{alt}\}$ where $\diamond_{alt}$ is defined by

$$s \diamond_{alt} t \triangleq \{s, t\}$$

Explicit nondeterministic choice is defined by operator $\diamond_{xalt} \triangleq \{\diamond_{xalt_1}, \diamond_{xalt_2}\}$ where

$$s \diamond_{xalt_1} t \triangleq \{s\} \qquad s \diamond_{xalt_2} t \triangleq \{t\}$$

Finally, parallel composition is defined by $\diamond_{par} \triangleq \{\diamond_{par}\}$ where $s \diamond_{par} t$ yields the set of all well-formed interleavings of its arguments (see [7]). $\triangle$

We characterize compositions that allow us to exploit the security of the basic specifications in establishing the security of the composite specification. We follow the same line of reasoning as in the previous section. This is because the formal definition of a composite operator is of the same form as the definition of a translation (Def. 9). The only essential difference is that a composition operator takes two arguments whereas a translation takes one argument. Hence, we need only to generalize the results of the previous section such that they apply to operators with two arguments instead of one.

In the following we assume three fixed but arbitrary security requirements, one for each basic specification, and one for the composite specification:

$$\mathrm{B}_1 \triangleq \mathrm{BSP}_{\mathfrak{r}_1\mathfrak{h}_1\mathfrak{l}_1} \qquad \mathrm{B}_2 \triangleq \mathrm{BSP}_{\mathfrak{r}_2\mathfrak{h}_2\mathfrak{l}_2} \qquad \mathrm{C} \triangleq \mathrm{BSP}_{\mathfrak{r}\mathfrak{h}\mathfrak{l}} \tag{14}$$

We want to exploit the fact that the basic specification Ω_1 is secure w.r.t. B_1 and that the basic specification Ω_2 is secure w.r.t. B_2 in order to show that the composition $\Omega_1 \diamond \Omega_2$ is secure w.r.t. C. This scenario is illustrated in the diagram below.

$$\begin{array}{ccc}
s_1 \xrightarrow{\mathfrak{r}_1} t_1 & \overset{b_1}{\Longrightarrow} & \{s_1\} \xrightarrow{\mathfrak{h}_1} \phi_1 \sim_{\mathfrak{l}_1} \{t_1\} \\
\Big\Uparrow x_1 & & \Big\Downarrow y_1 \\
s \xrightarrow{\mathfrak{r}} t & \overset{c}{\Longrightarrow} & \{s\} \xrightarrow{\mathfrak{h}} \phi \sim_{\mathfrak{l}} \{t\} \\
\Big\Downarrow x_2 & & \Big\Uparrow y_2 \\
s_2 \xrightarrow{\mathfrak{r}_2} t_2 & \underset{b_2}{\Longrightarrow} & \{s_2\} \xrightarrow{\mathfrak{h}_2} \phi_2 \sim_{\mathfrak{l}_2} \{t_2\}
\end{array}$$

As in the previous section, the implications labeled b_1, b_2, and c are understood to hold whenever $\mathrm{B}_1(\Omega_1)$, $\mathrm{B}_2(\Omega_2)$, and $\mathrm{C}(\Omega_1 \diamond \Omega_2)$ hold, respectively. Thus if the basic specifications Ω_1 and Ω_2 are secure, then the implications labeled b_1 and b_2 hold, and we can take advantage of this in order to establish the implication c. As indicated in the diagram, we can do this by ensuring that the implications labeled x_1, x_2, y_1, and y_2 hold. The conditions under which the implications hold are essentially the same as in the previous section except for the fact that we have to take two arguments into consideration instead of one.

Theorem 7 *$\Omega_1 \diamond \Omega_2$ is secure w.r.t. $\mathrm{BSP}_{\mathfrak{r}\mathfrak{h}\mathfrak{l}}$ if Ω_1 is secure w.r.t. $\mathrm{BSP}_{\mathfrak{r}_1\mathfrak{h}_1\mathfrak{l}_1}$, Ω_2 is secure w.r.t. $\mathrm{BSP}_{\mathfrak{r}_2\mathfrak{h}_2\mathfrak{l}_2}$, and $\diamond$ is a composition operator that satisfies the following conditions for all $\diamond_1, \diamond_2 \in \diamond$, $s_1, t_1 \in \widehat{\Omega_1}$, $\phi_1 \in \Omega_1$, $s_2, t_2 \in \widehat{\Omega_2}$, $\phi_2 \in \Omega_2$, $s \in (s_1 \diamond_1 s_2)$, $t \in (t_1 \diamond_2 t_2)$*

$$s \xrightarrow{\mathfrak{r}} t \implies (s_1 \xrightarrow{\mathfrak{r}_1} t_1 \wedge s_2 \xrightarrow{\mathfrak{r}_2} t_2)$$

$$\{s_1\} \xrightarrow{\mathfrak{h}_1} \phi_1 \sim_{\mathfrak{l}_1} \{t_1\} \wedge \{s_2\} \xrightarrow{\mathfrak{h}_2} \phi_2 \sim_{\mathfrak{l}_2} \{t_2\} \implies \exists \diamond \in \diamond : \{s\} \xrightarrow{\mathfrak{h}} (\phi_1 \diamond \phi_2) \sim_{\mathfrak{l}} \{t\}$$

5.1 Example of Composition

Preservation of security under composition enables the security of the specification as a whole to be established by analyzing each part of the specification in

isolation. This is usually less expensive than establishing security of the specification as a whole. Our approach offers the advantage that composition may be used together with translation and refinement in such a way that security is preserved. Also, any additional nondeterminism that arises as a result of composition will not violate the security of the specification. This is because all security properties defined in our framework are preserved under refinement.

In this section, we continue the example of Sect. 4.1. Recall that we formalized a translation from the high level protocol A to the low level protocol C for specifications consisting of a single lifeline. The translation for specifications with more than one lifeline may be defined by parallel composing the result of applying $A2C$ to each lifeline of the specification. For instance, the translation of specification DS' (Fig. 2) to the C level is defined by

$$DS'' = A2C([\![\, UserL \,]\!]) \diamond_{par} (A2C([\![\, Server \,]\!]) \diamond_{par} A2C([\![\, UserH \,]\!]))$$

In order to show that DS'' is secure w.r.t. the NF-property, we may check if the parallel composition operator $\diamond_{par}$ preserves the NF-property. This is sufficient because we have already shown that the specifications $[\![\, UserL \,]\!]$, $[\![\, Server \,]\!]$, and $[\![\, UserH \,]\!]$ of Fig. 2 are secure w.r.t. the NF-property (see Sect. 3.3) and that the translation $A2C$ preserves the NF-property (see Sect. 4.1).

To show that the NF property is preserved under the $\diamond_{par}$ operator, we instantiate the conditions of Theorem. 7 and get

$$\begin{array}{c} \phi_l|_{\mathcal{H}} = \{\langle\rangle\} \wedge \phi_l|_{\mathcal{L}} = \{t_l\}|_{\mathcal{L}} \wedge \phi_{sh}|_{\mathcal{H}} = \{\langle\rangle\} \wedge \phi_{sh}|_{\mathcal{L}} = \{t_{sh}\}|_{\mathcal{L}} \implies \\ (\phi_l \diamond_{par} \phi_{sh})|_{\mathcal{H}} = \{\langle\rangle\} \wedge (\phi_l \diamond_{par} \phi_{sh})|_{\mathcal{L}} = \{t\}|_{\mathcal{L}} \end{array} \tag{15}$$

for all t_l and ϕ_l in $A2C([\![\, UserL \,]\!])$, t_{sh} and ϕ_{sh} in $(A2C([\![\, Server \,]\!]) \diamond_{par} A2C([\![\, UserH \,]\!]))$, and t in $(t_l \diamond_{par} t_{sh})$.

The first part of condition of (15) is trivially satisfied because the interleaving of traces in two obligations that do not contain any high level events, will not contain any high level events either. The second part of condition (15) is satisfied because the set $\mathcal{L}$ of low level events is assumed to be all events occurring in $A2C([\![\, UserL \,]\!])$. Thus for all obligations ϕ_l and traces t_l in $[\![\, UserL \,]\!]$, we know that $\phi_l|_{\mathcal{L}} = \{t_l\}|_{\mathcal{L}} \Leftrightarrow \phi_l = \{t_l\}$. It follows that condition (15) is reduced to

$$(\{t_l\} \diamond_{par} \phi_{sh})|_{\mathcal{L}} = (\{t_l\} \diamond_{par} \{t_{sh}\})|_{\mathcal{L}} \tag{16}$$

Again, since $\mathcal{L}$ is exactly the set of events that can occur in $[\![\, A2C(UserL) \,]\!]$, we know that $\phi_{sh}|_{\mathcal{L}} = \{\langle\rangle\}$ and $\{t_{sh}\}|_{\mathcal{L}} = \{\langle\rangle\}$. Thus for each trace u' in $(\{t_l\} \diamond_{par} \phi_{sh})$ or $(\{t_l\} \diamond_{par} \{t_{sh}\})$, we know that $u'|_{\mathcal{L}} = t_l$, which again means that condition (16) is satisfied.

The parallel composition of two specification may result in new potential nondeterministic alternatives to be created. This may be useful when the underlying communication medium of a specification is underspecified. For instance, one might not know whether message may disappear during transmission, or whether messages are received in the same order that they are sent. On the one hand, the specification must take all these alternatives into account, but on the other hand, a system that for instance ensures that messages are always received in the same order that they are sent is a perfectly valid implementation of the specification.

As an example, the parallel composition of the two specifications $\{\{\langle !a, !b\rangle\}, \{\langle !c, !d\rangle\}\}$ and $\{\{\langle ?a, ?b\rangle, \langle ?b, ?a\rangle\}, \{\langle ?c, ?d\rangle, \langle ?d, ?c\rangle\}\}$, yields

$$\{\{\langle !a, !b, ?a, ?b\rangle, \langle !a, !b, ?b, ?a\rangle, \langle !a, ?a, !b, ?b\rangle\},\\ \{\langle !c, !d, ?c, ?d\rangle, \langle !c, !d, ?d, ?c\rangle, \langle !c, ?d, !d, ?d\rangle\}\}$$

Notice that messages are not necessarily received in the same order that they are sent, but that the alternative orders in which messages are received are specified as potential choices. This means the following specification in which messages are received in the same order they are sent is a refinement of the above:

$$\{\{\langle !a, !b, ?a, ?b\rangle, \langle !a, ?a, !b, ?b\rangle\}, \{\langle !c, !d, ?c, ?d\rangle, \langle !c, ?d, !d, ?d\rangle\}\}$$

In any case, since the parallel composition operator preserves the NF-property when the set $\mathcal{L}$ of low level events and the set $\mathcal{H}$ of high level events are assumed to be all events that can occur on two distinct lifelines, we know that additional potential nondeterministic alternatives that arise during composition will not violated the NF-property.

6 Related Work

This paper builds on [27] and in particular [28], where the authors showed how to preserve security properties under the notion of refinement (by underspecification) given in Sect. 2. A notion of security preserving transformations (similar to translation) was also defined. However, the security framework in [28] was based on [15], and composition was not considered. The main contributions of this paper are the security framework (Sect. 3), and conditions under which security properties are preserved under translation (Sect. 4) and composition (Sect. 5).

The notion of secure information flow was first introduced by Sutherland [29] as a generalization of the notion of non-interference [4]. Several such generalizations have later been proposed (see e.g. [18, 21, 31]) that again have led to the development of frameworks in which secure information flow properties can be represented in a uniform way and compared [1, 2, 15, 20, 22, 31]. Of these, the framework proposed by Mantel [15], is most similar to ours. Indeed, the modular way of constructing security properties from basic security predicates was introduced by Mantel [15], and the schema we propose for specifying such predicates is similar to Mantel's schema [15]. The main difference between our schema and the one proposed by Mantel is that ours is based on relations, whereas the one proposed by Mantel is not. The use of relations makes it easier to propose conditions w.r.t. translation and composition.

Our framework is, at first glance, similar to the so-called *generalized unwinding condition* [2]. The difference is that the unwinding condition is specified in terms of a relation on *processes* in the setting of process algebras. Our framework is defined in terms of relations on traces in a trace based model. In general, unwinding conditions demand properties on events (actions) rather than traces. This results in proof obligations that are easier to handle, but yields a less abstract definition of security than what we may define in our framework. Our framework exploits the distinction of underspecification and unpredictability, but we are not aware of any unwinding conditions in which this is done.

The work that is most similar to ours is the work of Santen et.al. [8, 26, 24, 25]. This is because the notions of information flow security, underspecification, data refinement (a notion similar to translation), and composition are all considered. Their notion of refinement is a based on CSP refinement notions together with a relation from concrete events to abstract events. This notion of refinement is similar to our notion of refinement by underspecification modulo translation. However, our notion of translation is more general than the data refinement considered by Santen et. al. because we map (abstract) traces to (concrete) trace sets as opposed to (abstract) events to (concrete) event sets. Without this generality, we would not have been able to characterize the transformation of the example in Sect. 4.1 where abstract events are mapped to several concrete traces. Another difference is that the work of Santen et. al. considers *probabilistic* security properties as opposed to *possibilistic* security properties as we do. Consequently the semantic model of specifications and the security framework considered by Santen et. al. are substantially different from the semantics and security framework of this paper.

Secure information flow properties in relation to data refinement also previously considered in [10, 5]. However, Graham-Cumming and Sanders [5] consider a less general notion of security than we do, and they only consider refinements of internal states of a system but not of the input and output data. The notion of data refinement presented by Hutter [10] is quite general, but refinement of underspecification is not considered. Another difference between Hutter's paper [10] and our work is that Hutter's specification model is based on so-called configuration structures and his security framework is different from ours.

There are a number of papers addressing the compositionality of secure information flow properties [2, 9, 19, 17, 20, 30]. [19, 30] focus on particular properties without considering a general framework of security properties. Hinton [9] proposes a notion of so-called emergent behavior and emergent properties as a way of analyzing composability of security properties. However, the treatment is rather informal in that no formal definition of a notion of composition nor formal conditions under which security properties are composable is given. [2, 17, 20] address compositionality in the light of a general framework for security properties. However, Mantel [17] and McLean [20] restricts attention to particular notions of composition. This is not the case for Bossi et. al. [2], which handles compositionality of security properties in a manner that is similar to our approach. The main difference is that (1) the security framework of Bossi et. al. [2] is based on the generalized unwinding condition in a process algebra, which as noted above, differs from the framework we propose in this paper, and (2) the distinction of unpredictability and underspecification is not made by Bossi et. al.

Jacob is the first person that we are aware of to show that secure information flow properties are not preserved by the standard notion of refinement [11]. Since then, a large number of papers have investigated the relationship of information flow security and refinement of underspecification. These can be classified into two categories: those that strengthen the notion of refinement, and those that strengthen the notion of security, e.g. by closing the security properties under refinement. Our work is in line with the latter approach. Other works that follows this approach are [13, 14]. In both papers, the security properties are explicitly closed under refinement. The authors of the papers are able to do this without making the security properties unreasonably strong be-

cause the notion of refinement considered prohibits the refinement of *external nondeterminism* (which is determined by the environment); only refinement of *internal nondeterminism* (which is determined by the system) is allowed. The notion of refinement considered in this paper is more general because it both allows external nondeterminism to be refined and it allows refinement of internal nondeterministic alternatives to constrained by specifying these as explicit alternatives. This generality is useful, particularly when input totality is not assumed. For instance, the example of Sect. 5.1 in which we refine the communication medium would not have been possible without being able to refine external nondeterminism.

Related work that ensures security preservation under refinement by strengthening the notion of refinement (instead of the security properties) are [1, 3, 16]. Bossi et. al. [3] present sufficient conditions with which to check that a given refinement preserves information flow properties. Apart from the fact that they do not strengthen their notion of security, their work differs from ours in that they consider specifications expressed in a process calculus, and that they focus attention to bisimulation-based properties. Mantel [16] presents refinement operators that can be used to check or modify refinements such that security is preserved. However, according to Heisel et. al. [8], the refinement operators may lead to concrete specifications that are practically hard to implement, because the changes in the refinement they induce are hard to predict and may not be easy to realize in an implementation. Alur et. al. [1] presents a notion of secrecy preserving refinement. The underlying idea is that an implementation leaks a secret only when the specification also leaks it. Apart from the fact that they do not strengthen their notion of security, their approach differs from ours in that they consider a different security framework and their specification notion is based on labeled transition systems.

7 Conclusions and Future Work

We have presented a framework that supports an incremental and modular development process for secure software systems. The framework supports the preservation of information flow properties under a series of successive refinement, transformation, and composition steps.

In [28], the authors showed how to preserve security properties under the notion of refinement (by underspecification) given in Sect. 2. A notion of security preserving transformations (similar to translation) was also defined. However, the security framework in [28] was based on [15], and composition was not considered. The main contributions of this paper are the security framework (Sect. 3), and conditions under which security properties are preserved under translation (Sect. 4) and composition (Sect. 5).

The emphasis of this paper has been on simplicity and generality in a semantic setting. In future work, we will address syntactic notions of transformation. Eventually, we would like to develop a computerized tool that checks whether transformations violate security.

References

[1] R. Alur, P. Černý, and S. Zdancewic. Preserving Secrecy Under Refinement. In *Proc. of the 33rd International Colloquium on Automata, Languages and Programming (ICALP'06)*, volume 4052 of *Lecture Notes in Computer Science*, pages 107–118. Springer, 2006.

[2] A. Bossi, R. Focardi, D. Macedonio, C. Piazza, and S. Rossi. Unwinding in information flow security. *Electronic Notes in Theoretical Computer Science*, 99:127–154, 2004.

[3] A. Bossi, R. Focardi, C. Piazza, and S. Rossi. Refinement operators and information flow security. In *Proc. of the 1st International Conference on Software Engineering and Formal Methods (SEFM'03)*, pages 44–53. IEEE Computer Society, 2003.

[4] J. A. Goguen and J. Meseguer. Security Policies and Security Models. In *Proc. of the 1982 IEEE Symposium on Security and Privacy (S&P'82)*, pages 11–20. IEEE Computer Society, 1982.

[5] J. Graham-Cumming and J. W. Sanders. On the refinement of non-interference. In *Proc. of the IEEE Computer Security Foundations Workshop*, pages 35–42. IEEE Computer Society Press, 1991.

[6] Ø. Haugen, K. E. Husa, R. K. Runde, and K. Stølen. Why timed sequence diagrams require three-event semantics. Research Report 309, Department of Informatics, University of Oslo, 2004.

[7] Ø. Haugen, K. E. Husa, R. K. Runde, and K. Stølen. STAIRS towards formal design with sequence diagrams. *Journal of Software and Systems Modeling*, 4(4):355–367, 2005.

[8] M. Heisel, A. Pfitzmann, and T. Santen. Confidentiality-preserving refinement. In *14th IEEE Computer Security Foundations Workshop (CSFW-14 2001)*, pages 295–306. IEEE Computer Society Press, 2001.

[9] H. M. Hinton. Under-specification, composition and emergent properties. In *1997 New Security Paradigms Workshop*, pages 83–93. ACM Press, 1997.

[10] D. Hutter. Possibilistic Information Flow Control in MAKS and Action Refinement. In *Proc. of Emerging Trends in Information and Communication Security, International Conference (ETRICS'06)*, volume 3995 of *Lecture Notes in Computer Science*, pages 268–281. Springer, 2006.

[11] J. Jacob. On the derivation of secure components. In *Proc. of the IEEE Symposium on Security and Privacy (S&P'89)*, pages 242–247. IEEE Computer Society, 1989.

[12] J. Jürjens. Secrecy-preserving refinement. In *Proc. of the International Symposium of Formal Methods Europe on Formal Methods for Increasing Software Productivity (FFME'01)*, volume 2021 of *Lecture Notes in Computer Science*, pages 135–152. Springer, 2001.

[13] J. Jürjens. *Secure systems development with UML*. Springer, 2005.

[14] G. Lowe. On information flow and refinement-closure. In *Proc. of the Workshop on Issues in the Theory of Security (WITS '07)*, 2007.

[15] H. Mantel. Possibilistic definitions of security - an assembly kit. In *IEEE Compuer Security Foundations Workshop (CSFW'00)*, pages 185–199. IEEE Computer Society Press, 2000.

[16] H. Mantel. Preserving information flow properties under refinement. In *IEEE Symposium on Security and Privacy*, pages 78–91. IEEE Computer Society Press, 2001.

[17] H. Mantel. On the composition of secure systems. In *IEEE Symposium on Security and Privacy*, pages 88–101. IEEE Computer Society Press, 2002.

[18] D. McCullough. Specifications for Multi-Level Security and a Hook-Up Property. In *Proc. of the IEEE Symposium on Security and Privacy (S&P'87)*, pages 161–166. IEEE Computer Society, 1987.

[19] D. McCullough. Noninterference and the composability of security properties. In *Proc. of the IEEE Symposium on Security and Privacy (S&P'88)*, pages 177–186. IEEE Computer Society, 1988.

[20] J. McLean. A General Theory of Composition for Trace Sets Closed under Selective Interleaving Functions. In *Proc. of the IEEE Symposium on Research in Security and Privacy (S&P'94)*, pages 79–93. IEEE Computer Society, 1994.

[21] C. O'Halloran. A calculus for information flow. In *Proc. of the 1st European Symposium on Research in Computer Security (ESORICS'90)*, pages 147–159. AFCET, 1990.

[22] A. W. Roscoe. CSP and determinism in security modelling. In *Proc. of the IEEE Symposium on Security and Privacy (S&P'95)*, pages 114–127. IEEE Computer Society Press, 1995.

[23] R. K. Runde, Ø. Haugen, and K. Stølen. Refining UML Interactions with Underspecification and Nondeterminism. *Nordic Journal of Computing*, 12(2):157–188, 2005.

[24] T. Santen. A Formal Framework for Confidentiality-Preserving Refinement. In *Proc. of the 11th European Symposium on Research in Computer Security (ESORICS'06)*, volume 4189 of *Lecture Notes in Computer Science*, pages 225–242. Springer, 2006.

[25] T. Santen. Preservation of probabilistic information flow under refinement. *Information and Computation*, 206(2-4):213–249, 2008.

[26] T. Santen, M. Heisel, and A. Pfitzmann. Confidentiality-preserving refinement is compositional - sometimes. In *Proc. of the 7th European Symposium on Research in Computer Security (ESORICS'02)*, volume 2502 of *Lecture Notes in Computer Science*, pages 194–211. Springer, 2002.

[27] F. Seehusen and K. Stølen. Information flow property preserving transformation of UML interaction diagrams. In *Proc. of the 11th ACM Symposium on Access Control Models and Technologies (SACMAT'06)*, pages 150–159. ACM, 2006.

[28] F. Seehusen and K. Stølen. Maintaining information flow security under refinement and transformation. In *Proc. of the 4th International Workshop on Formal Aspects in Security and Trust (FAST'06)*, Lecture Notes in Computer Science, pages 143–157. Springer, 2007.

[29] D. Sutherland. A model of information. In *Proc. of the 9th National Computer Security Conference*, pages 175–183, 1986.

[30] A. Zakinthinos and E. S. Lee. The composability of non-interference. In *Proc. of the 8th IEEE Computer Security Foundations Workshop (CSFW '95)*, pages 2–8. IEEE Computer Society, 1995.

[31] A. Zakinthinos and E. S. Lee. A general theory of security properties. In *Proc. of the IEEE Computer Society Symposium on Research in Security and Privacy (S&P'97)*, pages 94–102. IEEE Computer Society, 1997.

A Proofs

A.1 System specifications

Theorem 1 If specification Ω' is a refinement of specification Ω and system Φ is in compliance with Ω', then Φ is in compliance with Ω. Formally,

$$(\Omega \rightsquigarrow \Omega' \wedge \Omega' \rightarrowtail \Phi) \implies \Omega \rightarrowtail \Phi$$

Proof of Theorem 1
PROVE: $(\Omega \rightsquigarrow \Omega' \wedge \Omega' \rightarrowtail \Phi) \implies \Omega \rightarrowtail \Phi$

$\langle 1 \rangle 1$. ASSUME: 1.1. $\Omega \rightsquigarrow \Omega'$
1.2. $\Omega' \rightarrowtail \Phi$
PROVE: $\Omega \rightarrowtail \Phi$
$\langle 2 \rangle 1$. $(t \in \Phi \implies \exists \phi \in \Omega : t \in \phi) \wedge (\phi \in \Omega \implies \exists t \in \Phi : t \in \phi)$
$\langle 3 \rangle 1$. ASSUME: 3.1. $t \in \Phi$
PROVE: $\exists \phi \in \Omega : t \in \phi$
$\langle 4 \rangle 1$. Choose $\phi' \in \Omega'$ such that $t \in \phi'$
PROOF: By assumption 1.2, assumption 3.1, and definition of compliance (Def. 3).
$\langle 4 \rangle 2$. Choose $\phi \in \Omega$ such that $\phi' \subseteq \phi$
PROOF: By $\langle 4 \rangle 1$, assumption 1.1, and definition of refinement (Def. 4).
$\langle 4 \rangle 3$. Q.E.D.
PROOF: By $\langle 4 \rangle 1$ and $\langle 4 \rangle 2$.
$\langle 3 \rangle 2$. ASSUME: 3.1. $\phi \in \Omega$
PROVE: $\exists t \in \Phi : t \in \phi$
$\langle 4 \rangle 1$. Choose $\phi' \in \Omega'$ such that $\phi' \subseteq \phi$
PROOF: By assumption 3.1, assumption 1.1, and definition of refinement (Def. 4).

⟨4⟩2. Choose $t \in \Phi$ such that $t \in \phi'$
PROOF: By ⟨4⟩1, assumption 1.2, and definition of compliance (Def. 3).
⟨4⟩3. Q.E.D.
PROOF: By ⟨4⟩1 and ⟨4⟩2.
⟨3⟩3. Q.E.D.
PROOF: By ⟨3⟩1 and ⟨3⟩2.
⟨2⟩2. Q.E.D.
PROOF: By ⟨2⟩1 and definition of compliance (Def. 3).
⟨1⟩2. Q.E.D.
PROOF: By ⟨1⟩1.

A.2 Security framework for systems

Theorem 3 The following equivalences hold:

$$\begin{array}{lcl} \textsc{Re}(\Phi) & \Leftrightarrow & \textsc{Re'}(\Phi) \\ \textsc{Ri}(\Phi) & \Leftrightarrow & \textsc{Ri'}(\Phi) \\ \textsc{Ihai}(\Phi) & \Leftrightarrow & \textsc{Ihai'}(\Phi) \\ \textsc{Iae}(\Phi) & \Leftrightarrow & \textsc{Iae'}(\Phi) \end{array}$$

Proof of Theorem 3 By Lemma 3.1, Lemma 3.2, Lemma 3.3, and Lemma 3.4.

Lemma 3.1 The equivalence $\textsc{Re}(\Phi) \Leftrightarrow \textsc{Re'}(\Phi)$ holds.

Proof of Lemma 3.1
PROVE: $\textsc{Re}(\Phi) \Leftrightarrow \textsc{Re'}(\Phi)$

⟨1⟩1. ASSUME: 1.1 $\textsc{Re'}(\Phi)$
PROVE: $\textsc{Re}(\Phi)$
⟨2⟩1. ASSUME: 2.1 $s, t \in \Phi$
PROVE: $\exists u \in \Phi : (s|_{\mathcal{H}} = \langle\rangle \vee u|_{\mathcal{H}} = \langle\rangle) \wedge u \sim_{\mathfrak{l}} t$
⟨3⟩1. Choose $u \in \Phi$ such that $u|_{\mathcal{H}} = \langle\rangle$ and $u \sim_{\mathfrak{l}} t$
PROOF: By assumption 1.1, assumption 2.1, and definition of Re' (Sect. 3.2).
⟨3⟩2. Q.E.D.
PROOF: By ⟨3⟩1.
⟨2⟩2. Q.E.D.
PROOF: By ⟨2⟩1 and definition of Re (Sect. 3.2).
⟨1⟩2. ASSUME: 1.1 $\textsc{Re}(\Phi)$
PROVE: $\textsc{Re'}(\Phi)$
⟨2⟩1. ASSUME: 2.1 $t \in \Phi$
PROVE: $\exists u \in \Phi : u \sim_{\mathfrak{l}} t \wedge u|_{\mathcal{H}} = \langle\rangle$
⟨3⟩1. CASE: 3.1 $s|_{\mathcal{H}} \neq \langle\rangle$ for some $s \in \Phi$
⟨4⟩1. Choose $u \in \Phi$ such that $u \sim_{\mathfrak{l}} t$ and $u|_{\mathcal{H}} = \langle\rangle$
PROOF: By assumption 1.1, assumption 2.1, assumption 3.1, and definition of Re (Sect. 3.2).
⟨4⟩2. Q.E.D.
PROOF: By ⟨4⟩1.
⟨3⟩2. CASE: 3.1 $s|_{\mathcal{H}} = \langle\rangle$ for all $s \in \Phi$
PROOF: Choose $u \in \Phi$ such that $u = t$.
⟨3⟩3. Q.E.D.

PROOF: By ⟨3⟩1 and ⟨3⟩2.
⟨2⟩2. Q.E.D.
PROOF: By ⟨2⟩1 and definition of RE' (Sect. 3.2).
⟨1⟩3. Q.E.D.
PROOF: By ⟨1⟩1 and ⟨1⟩2.

Lemma 3.2 The equivalence RI(Φ) $\Leftrightarrow$ RI'(Φ) holds.

Proof of Lemma 3.2 The proof is the same as for Lemma 3.1, except that all occurrences of $\mathcal{H}$ in the proof must be replaced by $\mathcal{HI}$.

Lemma 3.3 The equivalence IHAI(Φ) $\Leftrightarrow$ IHAI'(Φ) holds.

Proof of Lemma 3.3
PROVE: IHAI(Φ) $\Leftrightarrow$ IHAI'(Φ)

⟨1⟩1. ASSUME: 1.1 IHAI'(Φ)
PROVE: IHAI(Φ)
⟨2⟩1. ASSUME: 2.1 $s, t \in \Phi$
2.2 $e \in \mathcal{HI}$
2.3 $\alpha|_{\mathcal{HI}} = \langle\rangle$ for some $\alpha \in \mathcal{T}$
2.4 $t = \beta \frown \alpha$ for some $\beta \in \mathcal{T}$
2.5 $\gamma'|_{\mathcal{HI}} = \beta|_{\mathcal{HI}}$ for some $\gamma' \in \mathcal{T}$
2.6 $s = \gamma' \frown \langle e \rangle$
PROVE: $\exists u \in \Phi : u \sim_{\mathrm{I}} t \wedge s|_{\mathcal{HI}} = u|_{\mathcal{HI}}$
⟨3⟩1. $R_{\mathrm{IHAI'}}(\Phi, t, \alpha, \beta, e)$
PROOF: By assumptions 2.1 - 2.6 and definition of $R_{\mathrm{IHAI'}}$ (Table 1 in Sect. 3.2).
⟨3⟩2. Choose $u \in \Phi$ and $\alpha', \beta' \in \mathcal{T}$ such that $u \sim_{\mathrm{I}} t$, $u = \beta' \frown \langle e \rangle \frown \alpha'$, $\beta'|_{\mathcal{L}\cup\mathcal{HI}} = \beta|_{\mathcal{L}\cup\mathcal{HI}}$, and $\alpha'|_{\mathcal{L}\cup\mathcal{HI}} = \alpha|_{\mathcal{L}\cup\mathcal{HI}}$
PROOF: By ⟨3⟩1, assumption 1.1, definition of IHAI' (Sect. 3.2).
⟨3⟩3. $s|_{\mathcal{HI}} = u|_{\mathcal{HI}}$
PROOF:

$\gamma'\|_{\mathcal{HI}}$	$=$	$\beta\|_{\mathcal{HI}}$	By assumption 2.5
$\gamma'\|_{\mathcal{HI}}$	$=$	$\beta'\|_{\mathcal{HI}}$	By ⟨3⟩2
$(\gamma' \frown \langle e \rangle)\|_{\mathcal{HI}}$	$=$	$(\beta' \frown \langle e \rangle)\|_{\mathcal{HI}}$	By definition of |
$(\gamma' \frown \langle e \rangle)\|_{\mathcal{HI}}$	$=$	$(\beta' \frown \langle e \rangle \frown \alpha')\|_{\mathcal{HI}}$	By assumption 2.3 and ⟨3⟩2
$s\|_{\mathcal{HI}}$	$=$	$u\|_{\mathcal{HI}}$	By assumption 2.6 and ⟨3⟩2

⟨3⟩4. Q.E.D.
PROOF: By ⟨3⟩2, ⟨3⟩3.
⟨2⟩2. Q.E.D.
PROOF: By definition of IHAI (Sect. 3.2).
⟨1⟩2. ASSUME: 1.1 IHAI(Φ)
PROVE: IHAI'(Φ)
⟨2⟩1. ASSUME: 2.1 $t \in \Phi$, $\alpha, \beta \in \mathcal{T}$, and $e \in \mathcal{E}$
2.2 $e \in \mathcal{HI}$
2.3 $\alpha|_{\mathcal{HI}} = \langle\rangle$
2.4 $t = \beta \frown \alpha$
2.5 $\gamma'|_{\mathcal{HI}} = \beta|_{\mathcal{HI}}$ for some $\gamma' \in \mathcal{T}$
2.6 $\gamma' \frown \langle e \rangle \in \Phi$

PROVE: $\exists u \in \Phi, \alpha', \beta' \in \mathcal{T} : u \sim_{\mathfrak{l}} t \wedge u = \beta' \frown \langle e \rangle \frown \alpha' \wedge \beta'|_{\mathcal{L}\cup\mathcal{HI}} = \beta|_{\mathcal{L}\cup\mathcal{HI}} \wedge \alpha'|_{\mathcal{L}\cup\mathcal{HI}} = \alpha|_{\mathcal{L}\cup\mathcal{HI}}$

⟨3⟩1. Choose $s \in \Phi$ such that $s = \gamma' \frown \langle e \rangle$
PROOF: By assumption 2.6.

⟨3⟩2. $s \xrightarrow{\mathfrak{r}_{\text{IHAI}}} t$
PROOF: By assumptions 2.1 - 2.6, ⟨3⟩1 and definition of $\mathfrak{r}_{\text{IHAI}}$ (Sect. 3.2).

⟨3⟩3. Choose $u \in \Phi$ such that $u \sim_{\mathfrak{l}} t$ and $s|_{\mathcal{HI}} = u|_{\mathcal{HI}}$
PROOF: By assumption 1.2, ⟨3⟩2 and definition of IHAI (Sect. 3.2).

⟨3⟩4. $u|_{\mathcal{HI}\cup\mathcal{L}} = (\beta \frown \langle e \rangle \frown \alpha)|_{\mathcal{HI}\cup\mathcal{L}}$
PROOF:

$u\|_{\mathcal{HI}}$	$=$	$(\gamma' \frown \langle e \rangle)\|_{\mathcal{HI}}$	By ⟨3⟩1 and ⟨3⟩3
$u\|_{\mathcal{HI}}$	$=$	$(\gamma' \frown \langle e \rangle \frown \alpha)\|_{\mathcal{HI}}$	By assumption 2.3
$u\|_{\mathcal{HI}}$	$=$	$(\beta \frown \langle e \rangle \frown \alpha)\|_{\mathcal{HI}}$	By assumption 2.5
$u\|_{\mathcal{HI}\cup\mathcal{L}}$	$=$	$(\beta \frown \langle e \rangle \frown \alpha)\|_{\mathcal{HI}\cup\mathcal{L}}$	By ⟨3⟩3, assumption 2.4 and def. of $\mathfrak{l}$

⟨3⟩5. Q.E.D.
PROOF: By ⟨3⟩4.

⟨2⟩2. Q.E.D.
PROOF: By definition of IHAI' (Sect. 3.2).

⟨1⟩3. Q.E.D.
PROOF: By ⟨1⟩1 and ⟨1⟩2.

Lemma 3.4 The equivalence IAE(Φ) $\Leftrightarrow$ IAE'(Φ) holds.

Proof of Lemma 3.4
PROVE: IAE(Φ) $\Leftrightarrow$ IAE'(Φ)

⟨1⟩1. ASSUME: 1.1 IAE'(Φ)
PROVE: IAE(Φ)

⟨2⟩1. ASSUME: 2.1 $s, t \in \Phi$
2.2 $e \in \mathcal{H}$
2.3 $\alpha|_{\mathcal{H}} = \langle\rangle$ for some $\alpha \in \mathcal{T}$
2.4 $t = \beta \frown \alpha$ for some $\beta \in \mathcal{T}$
2.5 $s = \beta \frown \langle e \rangle$
PROVE: $\exists u \in \Phi : u \sim_{\mathfrak{l}} t \wedge h(s) = h(u)$

⟨3⟩1. $R_{\text{IAE'}}(\Phi, t, \alpha, \beta, e)$
PROOF: By assumptions 2.1 - 2.5 and definition of $R_{\text{IAE'}}$ (Sect. 3.2).

⟨3⟩2. Choose $u \in \Phi$ such that $t \sim_{\mathfrak{l}} u$ and $u = \beta \frown \langle e \rangle \frown \alpha$
PROOF: By ⟨3⟩1, assumption 1.1, and definition of IAE' (Sect. 3.2).

⟨3⟩3. $h(s) = h(u)$
PROOF: By ⟨3⟩2, assumption 2.5, assumption 2.3, and definition of h (Sect. 3.2).

⟨3⟩4. Q.E.D.
PROOF: By ⟨3⟩2 and ⟨3⟩3.

⟨2⟩2. Q.E.D.
PROOF: By definition of IAE (Sect. 3.2).

⟨1⟩2. ASSUME: 1.1 IAE(Φ)
PROVE: IAE'(Φ)

⟨2⟩1. ASSUME: 2.1 $t \in \Phi$, $\alpha, \beta \in \mathcal{T}$, and $e \in \mathcal{E}$
2.2 $e \in \mathcal{H}$
2.4 $\alpha|_{\mathcal{H}} = \langle\rangle$

2.5 $t = \beta \frown \alpha$
2.6 $\beta \frown \langle e \rangle \in \Phi$
PROVE: $\exists u \in \Phi : u \sim_\mathfrak{l} t \wedge u = \beta \frown \langle e \rangle \frown \alpha$
⟨3⟩1. Choose $s \in \Phi$ such that $s = \beta \frown \langle e \rangle$
PROOF: By assumption 2.6.
⟨3⟩2. $s \xrightarrow{\mathfrak{r}_{\text{IAE}}} t$
PROOF: By assumptions 2.1 - 2.6, ⟨3⟩1 and definition of $\mathfrak{r}_{\text{IAE}}$ (Sect. 3.2).
⟨3⟩3. Choose $u \in \Phi$ such that $u \sim_\mathfrak{l} t$ and $h(s) = h(u)$
PROOF: By ⟨3⟩2, assumption 1.1, and definition of IAE (Sect. 3.2).
⟨3⟩4. $u = \beta \frown \langle e \rangle \frown \alpha$
⟨4⟩1. $h(\beta \frown \langle e \rangle) = h(u)$
PROOF: By ⟨3⟩1 and ⟨3⟩3.
⟨4⟩2. $(\beta \frown \alpha)|_\mathcal{L} = u|_\mathcal{L}$
PROOF: By assumption 2.5, ⟨3⟩3, and definition of $\mathfrak{l}$.
⟨4⟩3. $\beta \sqsubseteq u$
PROOF: By ⟨4⟩1 , assumption 2.2, and definition of h, there must be a prefix of u that is identical to β when we replace all low-level events in the two traces by the dummy event $\surd$. By ⟨4⟩2, that prefix must also contain exactly the same low-level events that are in β. Hence, the prefix is equal to β.
⟨4⟩4. $\beta \frown \langle e \rangle \sqsubseteq u$
PROOF: By assumption 2.2, ⟨4⟩1, ⟨4⟩3, and definition of h.
⟨4⟩5. Choose $\alpha' \in \mathcal{T}$ such that $u = \beta \frown \langle e \rangle \frown \alpha'$
PROOF: By ⟨4⟩4 and definition of $\sqsubseteq$.
⟨4⟩6. $\alpha' = \alpha$
PROOF: If $\alpha' \neq \alpha$, then (a) α' must contain a high-level event (by assumption 2.4) or (b) α' must differ from α on the low-level events. (a) cannot hold due to ⟨4⟩1 and definition of h, and (b) cannot hold due to ⟨4⟩2 and definition of $|$. Therefore $\alpha' = \alpha$.
⟨4⟩7. Q.E.D.
PROOF: By ⟨4⟩5 and ⟨4⟩6.
⟨3⟩5. Q.E.D.
PROOF: By ⟨3⟩1, ⟨3⟩3, and ⟨3⟩4.
⟨2⟩2. Q.E.D.
PROOF: By definition IAE' (Sect. 3.2).
⟨1⟩3. Q.E.D.
PROOF: By ⟨1⟩1 and ⟨1⟩2.

A.3 Security framework for specifications

Theorem 4 $\text{BSP}_{\mathfrak{r}\mathfrak{h}\mathfrak{l}}$ is preserved by refinement for arbitrary restriction relation $\mathfrak{r}$, high-level relation $\mathfrak{h}$, and low-level equivalence relation $\mathfrak{l}$:

$$\Omega \rightsquigarrow \Omega' \wedge \text{BSP}_{\mathfrak{r}\mathfrak{h}\mathfrak{l}}(\Omega) \implies \text{BSP}_{\mathfrak{r}\mathfrak{h}\mathfrak{l}}(\Omega')$$

Proof of Theorem 4
ASSUME: 1. $\Omega \rightsquigarrow \Omega'$
2. $\text{BSP}_{\mathfrak{r}\mathfrak{h}\mathfrak{l}}(\Omega)$
PROVE: $\text{BSP}_{\mathfrak{r}\mathfrak{h}\mathfrak{l}}(\Omega')$

⟨1⟩1. ASSUME: 1.1 $s \xrightarrow{\mathfrak{r}} t$ for some $s, t \in \widehat{\Omega'}$
PROVE: $\exists \phi' \in \Omega' : \{s\} \xrightarrow{\mathfrak{h}} \phi' \sim_{\mathfrak{l}} \{t\}$
⟨2⟩1. $s, t \in \widehat{\Omega}$
PROOF: By assumption 1.1, assumption 1, and definition of $\rightsquigarrow$ (Def. 4).
⟨2⟩2. Choose $\phi \in \Omega$ such that $\{s\} \xrightarrow{\mathfrak{h}} \phi \sim_{\mathfrak{l}} \{t\}$
PROOF: By assumption 1.1, ⟨2⟩1, assumption 2, and definition of BSP's (Def. 8).
⟨2⟩3. Choose $\phi' \in \Omega'$ such that $\phi' \subseteq \phi$
PROOF: By ⟨2⟩2, assumption 1, and definition of $\rightsquigarrow$ (Def. 4).
⟨2⟩4. $\{s\} \xrightarrow{\mathfrak{h}} \phi' \sim_{\mathfrak{l}} \{t\}$
PROOF: By ⟨2⟩2, ⟨2⟩3 and definition of $\rightarrow$ on trace sets (see Sect. 3.3).
⟨2⟩5. Q.E.D.
PROOF: By ⟨2⟩4.
⟨1⟩2. Q.E.D.
PROOF: By ⟨1⟩1 and definition of BSP's (Def. 8).

Theorem 5 Let Φ be a system, and let Ω be the specification obtained from Φ by creating a new obligation for each trace in Φ, i.e. $\Omega = \{\{t\} \mid t \in \Phi\}$. Then Ω satisfies the basic security predicate $\text{BSP}_{\mathfrak{rhl}}$ (of Def. 8) if and only if Φ satisfies $\text{BSP}_{\mathfrak{rhl}}$ (of Def. 5), i.e.,

$$\text{BSP}_{\mathfrak{rhl}}(\Phi) \Leftrightarrow \text{BSP}_{\mathfrak{rhl}}(\{\{t\} \mid t \in \Phi\})$$

Proof of Theorem 5
ASSUME: 1. $\Omega = \{\{t\} \mid t \in \Phi\}$
PROVE: $\text{BSP}_{\mathfrak{rhl}}(\Phi) \Leftrightarrow \text{BSP}_{\mathfrak{rhl}}(\Omega)$
⟨1⟩1. ASSUME: 1.1. $\text{BSP}_{\mathfrak{rhl}}(\Phi)$
PROVE: $\text{BSP}_{\mathfrak{rhl}}(\Omega)$
⟨2⟩1. ASSUME: 2.1 $s \xrightarrow{\mathfrak{r}} t$ for some $s, t \in \widehat{\Omega}$
PROVE: $\exists \phi \in \Omega : \{s\} \xrightarrow{\mathfrak{h}} \phi \sim_{\mathfrak{l}} \{t\}$
⟨3⟩1. Choose $u \in \Phi$ such that $s \xrightarrow{\mathfrak{h}} u \sim_{\mathfrak{l}} t$
PROOF: By assumption 1, assumption 1.1, assumption 2.1 and definition of BSP (Def. 5).
⟨3⟩2. Choose $\phi \in \Omega$ such that $\phi = \{u\}$
PROOF: By assumption 1.
⟨3⟩3. Q.E.D.
PROOF: By ⟨3⟩1 and ⟨3⟩2 and definition of $\rightarrow$ on trace sets (Sect.3.3).
⟨2⟩2. Q.E.D.
PROOF: By definition of BSP (Def. 8).
⟨1⟩2. ASSUME: 1.1. $\text{BSP}_{\mathfrak{rhl}}(\Omega)$
PROVE: $\text{BSP}_{\mathfrak{rhl}}(\Phi)$
⟨2⟩1. ASSUME: 2.1 $s \xrightarrow{\mathfrak{r}} t$ for some $s, t \in \Phi$
PROVE: $u \in \Phi : s \xrightarrow{\mathfrak{h}} u \sim_{\mathfrak{l}} t$
⟨3⟩1. Choose $\{u\} \in \Omega$ such that $\{s\} \xrightarrow{\mathfrak{h}} \{u\} \sim_{\mathfrak{l}} \{t\}$
PROOF: By assumption 1, assumption 1.1, assumption 2.1 and definition of BSP (Def. 8).
⟨3⟩2. Q.E.D.

PROOF: By ⟨3⟩1 and definition of → on trace sets (Sect.3.3).

⟨2⟩2. Q.E.D.

PROOF: By definition of BSP (Def. 8).

⟨1⟩3. Q.E.D.

PROOF: By ⟨1⟩1 and ⟨1⟩2.

A.4 Translation

Theorem 6 Specification $F(\Omega)$ is secure w.r.t. $\text{BSP}_{\mathfrak{r}'\mathfrak{h}'\mathfrak{l}'}$ if Ω is secure w.r.t. $\text{BSP}_{\mathfrak{r}\mathfrak{h}\mathfrak{l}}$ and F is a translation that satisfies the following conditions for all $f_1, f_2 \in F$, $s, t \in \widehat{\Omega}$, $\phi \in \Omega$, $s' \in f_1(s)$, and $t' \in f_2(t)$

$$s' \xrightarrow{\mathfrak{r}'} t' \implies s \xrightarrow{\mathfrak{r}} t$$

$$\{s\} \xrightarrow{\mathfrak{h}} \phi \sim_{\mathfrak{l}} \{t\} \implies \exists f \in F : \{s'\} \xrightarrow{\mathfrak{h}'} f(\phi) \sim_{\mathfrak{l}'} \{t'\}$$

Proof of Theorem 6

ASSUME: 1. $F \subseteq \mathcal{T} \to (\mathbb{P}(\mathcal{T}') \setminus \emptyset)$

2. $\text{BSP}_{\mathfrak{r}\mathfrak{h}\mathfrak{l}}(\Omega)$ for arbitrary $\Omega \in \mathbb{P}(\mathbb{P}(\mathcal{T}))$

3. $\forall f_1, f_2 \in F, s, t \in \widehat{\Omega}, s' \in f_1(s), t' \in f_2(t) : s' \xrightarrow{\mathfrak{r}'} t' \implies s \xrightarrow{\mathfrak{r}} t$

4. $\forall f_1, f_2 \in F, s, t \in \widehat{\Omega}, \phi \in \Omega, s' \in f_1(s), t' \in f_2(t) : \{s\} \xrightarrow{\mathfrak{h}} \phi \sim_{\mathfrak{l}} \{t\} \implies \exists f \in F : \{s'\} \xrightarrow{\mathfrak{h}'} f(\phi) \sim_{\mathfrak{l}'} \{t'\}$

PROVE: $\text{BSP}_{\mathfrak{r}'\mathfrak{h}'\mathfrak{l}'}(F(\Omega))$

⟨1⟩1. ASSUME: 1.1 $s' \xrightarrow{\mathfrak{r}'} t'$ for some $s', t' \in \widehat{F(\Omega)}$

PROVE: $\exists \phi' \in F(\Omega) : \{s'\} \xrightarrow{\mathfrak{h}'} \phi' \sim_{\mathfrak{l}'} \{t'\}$

⟨2⟩1. Choose $f_1, f_2 \in F$ and $s, t \in \widehat{\Omega}$ such that $s' \in f_1(s)$ and $t' \in f_2(t)$

PROOF: By assumptions 1 and 1.1, and definition of translation (Def. 9).

⟨2⟩2. $s \xrightarrow{\mathfrak{r}} t$

PROOF: By ⟨2⟩1, assumption 1.1, and assumption 3.

⟨2⟩3. Choose $\phi \in \Omega$ such that $\{s\} \xrightarrow{\mathfrak{h}} \phi \sim_{\mathfrak{l}} \{t\}$

PROOF: By assumption 2, ⟨2⟩2, and definition of BSP's (Def. 8).

⟨2⟩4. Choose $f \in F$ and $\phi' \in F(\Omega)$ such that $\phi' = f(\phi)$ and $\{s'\} \xrightarrow{\mathfrak{h}'} \phi' \sim_{\mathfrak{l}'} \{t'\}$

PROOF: By ⟨2⟩3, assumption 1, definition of translation (Def. 9), and assumption 4.

⟨2⟩5. Q.E.D.

PROOF: By ⟨2⟩4.

⟨1⟩2. Q.E.D.

PROOF: By definition of BSP's (Def. 8).

A.5 Composition

Theorem 7 $\Omega_1 \diamond \Omega_2$ is secure w.r.t. $\text{BSP}_{\mathfrak{r}\mathfrak{h}\mathfrak{l}}$ if Ω_1 is secure w.r.t. $\text{BSP}_{\mathfrak{r}_1\mathfrak{h}_1\mathfrak{l}_1}$, Ω_2 is secure w.r.t. $\text{BSP}_{\mathfrak{r}_2\mathfrak{h}_2\mathfrak{l}_2}$, and $\diamond$ is a composition operator that satisfies

the following conditions for all $\diamond_1, \diamond_2 \in \diamond$, $s_1, t_1 \in \widehat{\Omega_1}$, $\phi_1 \in \Omega_1$, $s_2, t_2 \in \widehat{\Omega_2}$, $\phi_2 \in \Omega_2$, $s \in (s_1 \diamond_1 s_2)$, $t \in (t_1 \diamond_2 t_2)$

$$s \xrightarrow{\mathfrak{r}} t \implies (s_1 \xrightarrow{\mathfrak{r}_1} t_1 \wedge s_2 \xrightarrow{\mathfrak{r}_2} t_2)$$

$$\{s_1\} \xrightarrow{\mathfrak{h}_1} \phi_1 \sim_{\mathfrak{l}_1} \{t_1\} \wedge \{s_2\} \xrightarrow{\mathfrak{h}_2} \phi_2 \sim_{\mathfrak{l}_2} \{t_2\} \implies \exists \diamond \in \diamond : \{s\} \xrightarrow{\mathfrak{h}} (\phi_1 \diamond \phi_2) \sim_{\mathfrak{l}} \{t\}$$

Proof of Theorem 7

ASSUME: 1. $\diamond \subseteq \mathcal{T} \times \mathcal{T} \to (\mathbb{P}(\mathcal{T}) \setminus \emptyset)$

2. $\text{BSP}_{\mathfrak{r}_1 \mathfrak{h}_1 \mathfrak{l}_1}(\Omega_1)$ for some $\Omega_1 \in \mathbb{P}(\mathbb{P}(\mathcal{T}))$

3. $\text{BSP}_{\mathfrak{r}_2 \mathfrak{h}_2 \mathfrak{l}_2}(\Omega_2)$ for some $\Omega_2 \in \mathbb{P}(\mathbb{P}(\mathcal{T}))$

4. $\forall \diamond_1, \diamond_2 \in \diamond, s_1, t_1 \in \widehat{\Omega_1}, s_2, t_2 \in \widehat{\Omega_2}, s \in (s_1 \diamond_1 s_2), t \in (t_1 \diamond_2 t_2) : s \xrightarrow{\mathfrak{r}} t \implies (s_1 \xrightarrow{\mathfrak{r}_1} t_1 \wedge s_2 \xrightarrow{\mathfrak{r}_2} t_2)$

5. $\forall \diamond_1, \diamond_2 \in \diamond, s_1, t_1 \in \widehat{\Omega_1}, \phi_1 \in \Omega_1, s_2, t_2 \in \widehat{\Omega_2}, \phi_2 \in \Omega, s \in (s_1 \diamond_1 s_2), t \in (t_1 \diamond_2 t_2) : \{s_1\} \xrightarrow{\mathfrak{h}_1} \phi_1 \sim_{\mathfrak{l}_1} \{t_1\} \wedge \{s_2\} \xrightarrow{\mathfrak{h}_2} \phi_2 \sim_{\mathfrak{l}_2} \{t_2\} \implies \exists \diamond \in \diamond : \{s\} \xrightarrow{\mathfrak{h}} (\phi_1 \diamond \phi_2) \sim_{\mathfrak{l}} \{t\}$

PROVE: $\text{BSP}_{\mathfrak{r}\mathfrak{h}\mathfrak{l}}(\Omega_1 \diamond \Omega_2)$

⟨1⟩1. ASSUME: 1.1 $s \xrightarrow{\mathfrak{r}} t$ for some $s, t \in \widehat{\Omega_1 \diamond \Omega_2}$

PROVE: $\exists \phi \in (\Omega_1 \diamond \Omega_2) : \{s\} \xrightarrow{\mathfrak{h}} \phi \sim_{\mathfrak{l}} \{t\}$

⟨2⟩1. Choose $\diamond_1, \diamond_2 \in \diamond$, $s_1, t_1 \in \widehat{\Omega_1}$, and $s_2, t_2 \in \widehat{\Omega_2}$ such that $s \in (s_1 \diamond_1 s_2)$ and $t \in (t_1 \diamond_2 t_2)$

PROOF: By assumption 1.1, assumption 1, and definition of composition (Def. 10).

⟨2⟩2. $s_1 \xrightarrow{\mathfrak{r}_1} t_1$ and $s_2 \xrightarrow{\mathfrak{r}_2} t_2$

PROOF: By ⟨2⟩1, assumption 1.1, and assumption 4.

⟨2⟩3. Choose $\phi_1 \in \Omega_1$ such that $\{s_1\} \xrightarrow{\mathfrak{h}_1} \phi_1 \sim_{\mathfrak{l}_1} \{t_1\}$

PROOF: By assumption 2, ⟨2⟩2, and definition of BSP's (Def. 8).

⟨2⟩4. Choose $\phi_2 \in \Omega_2$ such that $\{s_2\} \xrightarrow{\mathfrak{h}_2} \phi_2 \sim_{\mathfrak{l}_2} \{t_2\}$

PROOF: By assumption 3, ⟨2⟩2, and definition of BSP's (Def. 8).

⟨2⟩5. Choose $\diamond \in \diamond$ and $\phi \in (\Omega_1 \diamond \Omega_2)$ such that $\phi = \phi_1 \diamond \phi_2$ and $\{s\} \xrightarrow{\mathfrak{h}} \phi \sim_{\mathfrak{l}} \{t\}$

PROOF: By ⟨2⟩3, ⟨2⟩4, definition of composition (Def. 10), and assumption 5.

⟨2⟩6. Q.E.D.

PROOF: By ⟨2⟩5.

⟨1⟩2. Q.E.D.

PROOF: By definition of BSP's (Def. 8).