On the Composition of Secure Systems

Heiko Mantel
German Research Center for Artificial Intelligence (DFKI),
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
E-mail: mantel@dfki.de

Abstract

When complex systems are constructed from simpler components it is important to know how properties of the components behave under composition. In this article, we present various compositionality results for security properties. In particular, we introduce a novel security property and show that this property is, in general, composable although it is weaker than forward correctability. Moreover, we demonstrate that certain nontrivial security properties emerge under composition and illustrate how this fact can be exploited. All compositionality results that we present are verified with the help of a single, quite powerful lemma. Basing on this lemma, we also re-prove several already known compositionality results with the objective to unify these results. As a side effect, we obtain a classification of known compositionality results for security properties.

1 Introduction

It is absolutely crucial, for making the development of large and complex systems feasible, to apply some kind of divide-and-conquer approach. Requirements for the overall system are divided into subtasks, these subtasks are assigned to system components, and, after a specification of component interfaces, components can be developed independently. Upon composing the system, the satisfaction of the overall system requirements should follow from the fact that all components fulfill their specifications. Any need to inspect implementation details of system components in this process would not only violate the idea of the divide-and-conquer approach but also lead to an undesired increase of complexity. Moreover, since vendors are often unwilling to reveal details about their products, implementation details about some components might simply not be available.

For safety and liveness properties, general theories of compositionality exist [15, 32, 3] that provide a basis for the divide-and-conquer approach to system development. These theories are applicable for properties that can be specified as combinations of safety and liveness properties [4] like, e.g., most functional system properties. However, it is well known that many security properties are outside the domain of safety and liveness properties. For example, information flow properties are closure properties of sets of traces rather than properties of single traces [19]. Therefore, the above theories of composition cannot be applied for these security properties.

Since McCullough's proposal of a compositional information flow property [16] much progress has been made concerning the composition of secure systems. There is a collection of information flow properties that are preserved under arbitrary composition like, e.g., restrictiveness [16, 17], forward correctability [9], or separability [19]. For certain security properties that are, in general, not preserved under composition it is known how to restrict composition in order to preserve these properties [19, 20]. Nevertheless, a couple of important problems have remained unsolved. In particular, a uniform theory of composition for secure systems is still missing. To date, the various compositionality results are only loosely connected. Deeper insights, on how these results are related, would be highly desirable. Moreover, deriving compositionality results is often nontrivial and quite different techniques have been used to verify, e.g., the above results. More uniform verification techniques could be helpful to reveal similarities between different compositionality results and would also be helpful to simplify their proofs. McLean's theory of selective interleaving functions [19, 20] is certainly a step towards a uniform theory of composition for secure systems. However, this theory is not expressive enough to explain several known compositionality results. In particular, inductively defined security properties like, e.g., forward correctability, are outside the scope of selective interleaving functions.

The main objective of this article is to provide a uniform basis for compositionality results in the context of secure systems. For this purpose, we present a powerful lemma that is helpful for deriving compositionality results. Technically, our lemma can be regarded as a generalization of Johnson and Thayer's zipping lemma [9] that was intro-

In Proceedings of the 2002 IEEE Symposium on Security and Privacy, Oakland, CA, USA, pp. 88–101, May 12-15, 2002. IEEE Computer Society.

duced to prove the compositionality of forward correctability. However, we found that the underlying idea of this lemma is not restricted to forward correctability but rather is much more powerful. Our generalized zipping lemma can be used to prove known compositionality results of quite different flavor, as we will show at various examples. As a side effect of re-proving already known results with our lemma, we obtain a classification of compositionality results that makes close relations between some previously unrelated results explicit. Moreover, in contrast to a statement in [35], we show that there exists a security property that is weaker than forward correctability but, nevertheless, composable. Interestingly, our generalized zipping lemma is not only helpful to prove that security properties are *preserved* under composition (after they have been explicitely proved for the system components) but also for proving that security properties *emerge* under composition if certain conditions are fulfilled (without proving them explicitely). For example, if high- and low-level components are not physically connected to each other within a system then this system satisfies most information flow properties. However, we will show that information flow properties also emerge under more subtle conditions.

For the uniform representation of information flow properties we employ Mantel's modular assembly kit for security properties [11] (abbreviated by $MAKS$ in the sequel). In $MAKS$, information flow properties are composed from simple building blocks with the effect that reasoning about complex information flow properties can be reduced to reasoning about simpler building blocks. As a consequence, two orthogonal notions of composition occur in this article: firstly, the composition of system components and, secondly, the composition of security properties. Note that the focus of this article is on the preservation of security properties under the composition of system components. That security properties are composed from simpler building blocks is very helpful in our investigations, however, this is not the main novelty of the current article.

This article is structured as follows: in Section 2, we introduce a system model and the corresponding notion of composition that we use in this article. We also recall some basics about $MAKS$. In Section 3, we present a generalized zipping lemma that is the main technical contribution of this article. All compositionality results that are presented subsequently have been derived with the help of this lemma. A novel classification of known compositionality results is proposed in Section 4. In Section 5, we weaken Johnson and Thayer's forward correctability and demonstrate that the resulting security property is preserved under arbitrary compositions. In Section 6, we show that certain nontrivial information flow properties emerge under restricted forms of composition. Before we conclude in Section 8, we discuss the large body of related work in Section 7. Proof sketches of our main results are contained in the appendix.

2 Preliminaries

2.1 System Specifications

The behavior of a system can often be adequately specified by the set of its possible execution sequences. We follow this trace-based approach throughout this article. A *trace* is a sequence of events that models one possible execution sequence. An *event* is an atomic action like, e.g., sending or receiving a message. For a given system, we distinguish between input, output, and internal events. The underlying intuition is that input events are controlled by the environment while output and internal events are controlled by the system. When a system is capable to prevent occurrences of input events, then this can be regarded as a signal to the environment. To avoid this kind of communication, input totality is often assumed, i.e. that a system cannot prevent occurrences of input events. Since input totality is quite restrictive, we refrain from making this assumption.

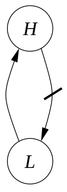
The system model that we assume is that of event systems. An *event system ES* is a tuple (E, I, O, Tr) where E is a *set of events*, $I, O \subseteq E$, respectively, are the *sets of input and output events*, and $Tr \subseteq E^*$ is the *set of traces*, i.e. a set of finite sequences over E. Each trace $\tau \in Tr$ models a possible behavior of *ES*. *Tr* must be closed under prefixes, i.e. any prefix of a trace in *Tr* must also be in *Tr*. Event systems allow for the specification of nondeterministic systems where nondeterminism is reflected by the choice between different events. Note, however, that event systems are a possibilistic system model that abstracts from probabilities.

Rather than specifying complex systems directly, they can be specified as the composition of simpler system components. Synchronization on the occurrence of shared events is used to model communication between different components of a system or between a system and its environment. We define the composition between event systems as usual with the restriction that output events of a component may only be connected to input events of other components. Moreover, communication events between components become internal events for the composed system.

Definition 1 *Assume* $E_1 \cap E_2 \subseteq (I_1 \cap O_2) \cup (I_2 \cap O_1)$, $I_1 \cap O_1 = \emptyset$, $I_2 \cap O_2 = \emptyset$. *The* composition *of* ES_1 *and* ES_2 *is the event system* $ES = ES_1 \parallel ES_2$ *where* E, I, O, *and* Tr *are defined by* $E = E_1 \cup E_2$, $I = (I_1 \setminus O_2) \cup (I_2 \setminus O_1)$, $O = (O_1 \setminus I_2) \cup (O_2 \setminus I_1)$, *and* $Tr = \{\tau \in (E_1 \cup E_2)^* \mid \tau|_{E_1} \in Tr_1 \wedge \tau|_{E_2} \in Tr_2\}$.[1]

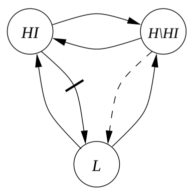
[1]In the remainder of this article, we assume that $ES = (E, I, O, Tr)$, $ES_1 = (E_1, I_1, O_1, Tr_1)$, and $ES_2 = (E_2, I_2, O_2, Tr_2)$ are event sys-

FP_{LH}



D	$\mathcal{V}_D^{LH}$
H	$(E,\emptyset,\emptyset)$
L	$(L,\emptyset,H) = \mathcal{V}_L^{LH}$

FP_{LHI}



D	$\mathcal{V}_D^{LHI}$
HI	$(E,\emptyset,\emptyset)$
$H\backslash HI$	$(E,\emptyset,\emptyset)$
L	$(L,H\backslash HI,HI) = \mathcal{V}_L^{LHI}$

Figure 1. The flow policies FP_{LH} and FP_{LHI} and the views of all domains

For the compositionality of security properties, two restricted forms of composition are of special interest: product and cascade. The composition of ES_1 and ES_2 is a *product* if the event systems do not synchronize on any events (i.e. $E_1 \cap E_2 = \emptyset$). The composition is a *cascade* if ES_1 receives no inputs from ES_2 ($I_1 \cap O_2 = \emptyset$), all outputs of ES_1 are inputs of ES_2, and ES_2 has no other inputs ($O_1 = I_2$). For a *relaxed cascade*, only $I_1 \cap O_2 = \emptyset$ is required.

2.2 Security Properties

Often, security requirements can be expressed nicely as restrictions on the allowed flow of information within a system. To express confidentiality or integrity by such restrictions is the key idea of information flow control. For the specification of information flow properties, we employ *MAKS*, Mantel's modular assembly kit for security properties [11, 12, 13, 14]. In *MAKS*, an *information flow property* consists of two elements: a flow policy and a security predicate.

A *flow policy FP* is a tuple $(\mathcal{D}, \rightsquigarrow_V, \rightsquigarrow_N, \not\rightsquigarrow)$ that specifies restrictions on the allowed flow of information within a system. $\mathcal{D}$ specifies a set of security domains. Typical domains are, e.g., groups of users, collections of files, or memory sections. The relations $\rightsquigarrow_V, \rightsquigarrow_N, \not\rightsquigarrow \subseteq \mathcal{D}\times\mathcal{D}$ must form a disjoint partition of $\mathcal{D}\times\mathcal{D}$ and $\rightsquigarrow_V$ must be reflexive. The *noninterference relation* $\not\rightsquigarrow$ specifies where information flow between domains is forbidden. E.g., $D_1 \not\rightsquigarrow D_2$ expresses that *information must not flow* from D_1 to D_2. The *interference relation* $\rightsquigarrow_V$ specifies that activities of certain domains are directly visible for others. $D_1 \rightsquigarrow_V D_2$ expresses that activities of D_1 *are visible* for D_2. Finally, the relation $\rightsquigarrow_N$ specifies between which domains information flow is *not* restricted. $D_1 \rightsquigarrow_N D_2$ expresses that activities of D_1 are *not* directly visible for D_2 (in contrast to $\rightsquigarrow_V$) and that we do *not* care if information about activities of D_1 is deducible for D_2 (in contrast to $\not\rightsquigarrow$). Note that for any two domains either $D_1 \not\rightsquigarrow D_2$, $D_1 \rightsquigarrow_V D_2$, or $D_1 \rightsquigarrow_N D_2$ holds. If $\rightsquigarrow_V$ is a transitive relation then *FP* is called *transitive* and, otherwise, *intransitive*. In this article, we will only consider transitive flow policies.

tems for which $E_1 \cap E_2 \subseteq (I_1 \cap O_2) \cup (I_2 \cap O_1)$, $I_1 \cap O_1 = \emptyset$, and $I_2 \cap O_2 = \emptyset$ hold. *ES* shall be defined as the composition of ES_1 and ES_2, i.e. $ES = ES_1 \parallel ES_2$. The *projection* $\alpha|_{E'}$ of a sequence $\alpha \in E^*$ to a set $E' \subseteq E$ results from α by removing all events that are *not* in E'.

A *domain assignment* is a function $dom : E \to \mathcal{D}$ that assigns security domains to events. Thereby, a domain assignment links an information flow property to a system specification. We often leave *dom* implicit and denote the set of all events that are associated with a given domain D also by D, the name of the security domain. We also use that name in lower case, possibly with indices or primes, e.g., $d, d_1, \ldots$, to denote events with that domain.

Flow policies can be depicted as graphs where each node corresponds to a security domain. The relations $\rightsquigarrow_V$, $\rightsquigarrow_N$, and $\not\rightsquigarrow$ are depicted as solid, dashed, and crossed arrows, respectively. For the sake of readability, the reflexive subrelation of $\rightsquigarrow_V$ is usually omitted. This graphical representation is illustrated in Figure 1 for two flow policies FP_{LH} and FP_{LHI}. Flow policy FP_{LH} consists of two domains H (high-level events) and L (low-level events). According to FP_{LH}, occurrences of low-level events are visible for the high-level domain ($L \rightsquigarrow_V H$). Occurrences of high-level events must not be visible to L and, moreover, no information about such occurrences must be deducible for L ($H \not\rightsquigarrow L$). The flow policy FP_{LHI} results from FP_{LH} by splitting the high-level domain into two domains HI (high-level input events) and $H\backslash HI$ (high-level internal and output events). The main difference to FP_{LH} is the dashed arrow from $H\backslash HI$ to L. While occurrences of high-level input events must not be deducible for the low-level ($HI \not\rightsquigarrow L$), we do not care if information about occurrences of other high-level events is deducible for the low-level domain ($H\backslash HI \rightsquigarrow_N L$).

The *view* $\mathcal{V}_D = (V, N, C)$ *for a domain* $D \in \mathcal{D}$ in *FP* under *dom* is a disjoint partition of E that is defined by $V = \{e \in E \mid dom(e) \rightsquigarrow_V D\}$, $N = \{e \in E \mid dom(e) \rightsquigarrow_N D\}$, and $C = \{e \in E \mid dom(e) \not\rightsquigarrow D\}$.[2] Consequently, V contains all events that are *v̲isible* for D, C contains all events *c̲onfidential* for D, and N contains all events *n̲either* visible nor confidential for D. The views of all domains in FP_{LH} and FP_{LHI} are depicted in Figure 1. Among these, only the views of the low-level domain L are interesting because they give rise to nontrivial proof obligations. We abbreviate the view of L in FP_{LH} and FP_{LHI} by, respectively, $\mathcal{V}_L^{LH}$ (L visible, H confidential) and $\mathcal{V}_L^{LHI}$ (L visible, $H \cap I$ confidential). The precise proof obligations for these views depend on the security predicate.

[2]In the remainder of this article, we assume that $\mathcal{V} = (V, N, C)$, $\mathcal{V}_1 = (V_1, N_1, C_1)$, $\mathcal{V}_2 = (V_2, N_2, C_2)$ are views in, respectively, E, E_1, E_2.

BSP	$\mathrm{BSP}_{\mathcal{V}}(Tr)$
R	$\forall \tau \in E^*.\, (\tau \in Tr \Longrightarrow \exists \tau' \in E^*.\, (\tau' \in Tr \wedge \tau'\|_V = \tau\|_V \wedge \tau'\|_C = \langle\rangle))$
BSD	$\forall \alpha, \beta \in E^*.\forall c \in C.\, ((\beta.c.\alpha \in Tr \wedge \alpha\|_C = \langle\rangle)$ $\Longrightarrow \exists \alpha' \in E^*.\, (\beta.\alpha' \in Tr \wedge \alpha'\|_V = \alpha\|_V \wedge \alpha'\|_C = \langle\rangle))$
BSI	$\forall \alpha, \beta \in E^*.\forall c \in C.\, ((\beta.\alpha \in Tr \wedge \alpha\|_C = \langle\rangle)$ $\Longrightarrow \exists \alpha' \in E^*.\, (\beta.c.\alpha' \in Tr \wedge \alpha'\|_V = \alpha\|_V \wedge \alpha'\|_C = \langle\rangle))$
$BSIA^{\rho}$	$\forall \alpha, \beta \in E^*.\forall c \in C.\, (\beta.\alpha \in Tr \wedge \alpha\|_C = \langle\rangle \wedge Adm^{\rho}_{\mathcal{V}}(Tr, \beta, c))$ $\Longrightarrow \exists \alpha' \in E^*.\, (\beta.c.\alpha' \in Tr \wedge \alpha'\|_V = \alpha\|_V \wedge \alpha'\|_C = \langle\rangle))$
$FCD^{\nabla,\Delta,\Upsilon}$	$\forall \alpha, \beta \in E^*.\forall c \in C \cap \Upsilon.\forall v \in V \cap \nabla.\, ((\beta.c.v.\alpha \in Tr \wedge \alpha\|_C = \langle\rangle)$ $\Longrightarrow \exists \alpha' \in E^*.\exists \delta' \in (N \cap \Delta)^*.\, (\beta.\delta'.v.\alpha' \in Tr \wedge \alpha'\|_V = \alpha\|_V \wedge \alpha'\|_C = \langle\rangle))$
$FCI^{\nabla,\Delta,\Upsilon}$	$\forall \alpha, \beta \in E^*.\forall c \in C \cap \Upsilon.\forall v \in V \cap \nabla.\, ((\beta.v.\alpha \in Tr \wedge \alpha\|_C = \langle\rangle)$ $\Longrightarrow \exists \alpha' \in E^*.\exists \delta' \in (N \cap \Delta)^*.\, (\beta.c.\delta'.v.\alpha' \in Tr \wedge \alpha'\|_V = \alpha\|_V \wedge \alpha'\|_C = \langle\rangle))$

Figure 2. Definition of basic security predicates

2.3 Basic Security Predicates

Recall that an information flow property is defined by a flow policy *FP* and a security predicate *SP*. We say that an event system *satisfies an information flow property* (FP, SP) wrt. a domain assignment *dom* if *SP* holds for the view $\mathcal{V}$ (denoted by $SP_{\mathcal{V}}(Tr)$) of every domain in *FP* under *dom*. In *MAKS*, security predicates are composed by conjunction from one or more basic security predicates (abbreviated by BSP). $SP_{\mathcal{V}}(Tr)$ holds if $\mathrm{BSP}_{\mathcal{V}}(Tr)$ holds for every BSP from which *SP* is composed.

BSPs are closure properties on sets of possible traces. Intuitively, a BSP expresses that there are *sufficiently many possible traces* such that adversaries cannot deduce information of a particular kind (depending on the respective BSP). In Figure 2, a collection of concrete BSPs is presented.[3] These BSPs are abbreviated by *R* (for Removal), *BSD* (Backwards Strict Deletion), *BSI* (Backwards Strict Insertion), *BSIA* (Backwards Strict Insertion of Admissible events), *FCD* (Forward Correctable Deletion), and *FCI* (Forward Correctable Insertion).[4] Technically, each of these BSPs demands that a particular perturbation of a trace can be corrected such that the resulting sequence, again, is a possible trace.

For example, *BSI* perturbs a trace $\beta.\alpha$ by *inserting* a confidential event $c \in C$ such that it is not followed by any other confidential events ($\alpha|_C = \langle\rangle$). The requirement imposed by $BSI_{\mathcal{V}}(Tr)$ is that the resulting sequence $\beta.c.\alpha$ can be corrected to a possible trace. Corrections must be *causal*, i.e. they may only occur after the inserted event, and they are limited to events in N ($\alpha'|_C = \langle\rangle = \alpha|_C$ and $\alpha'|_V = \alpha|_V$). The requirement $BSI_{\mathcal{V}}(Tr)$ can be read as: the occurrence of a confidential event $c \in C$ must *not disable* possible V-observations. Hence, the security guarantee provided by *BSI* is: adversaries cannot deduce from any V-observation that a confidential event c *has not* occurred.

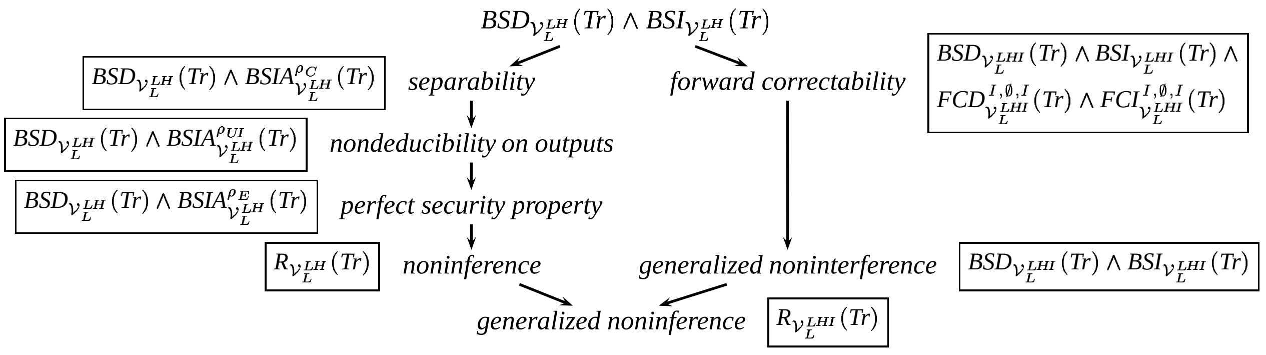
To guarantee that adversaries cannot deduce that a confidential event *has* occurred is the purpose of *BSD*, another BSP of *MAKS*. $BSD_{\mathcal{V}}(Tr)$ requires that the occurrence of an event from C must *not enable* additional V-observations. Considering the system after a trace $\beta.c$ has occurred, any observation $\alpha|_V$ that is possible must be possible also if $c \in C$ is *deleted* from the trace. Consequently, some sequence $\alpha' \in (V \cup N)^*$ must be enabled after β where α' may differ from α only in events from N.

The requirements imposed by the four remaining BSPs from Figure 2 can be informally described as follows. *R* is similar to *BSD* because it demands that the removal of events in C from a trace yields, again a possible trace. Since *R* only demands that *all* events in C can be *removed* at once it is a slightly weaker requirement than *BSD*. *FCD* is another BSP that is concerned with the *deletion* of events. However, it requires that the deletion of an event $c \in C \cap \Upsilon$ that occurs immediately before an event $v \in V \cap \nabla$, yields a trace. The specialty of *FCD* is that adaptations are not only restricted to after the occurrence of c but, moreover, adaptations in between c and v are restricted to events in $N \cap \Delta$. *BSIA* results from *BSI* by adding the assumption $Adm^{\rho}_{\mathcal{V}}(Tr, \beta, c)$.[5] Since only events that are ρ-admissible need to be *insertable*, *BSIA* is slightly weaker than *BSI*. *FCI* is concerned only with the *insertion* of events from $C \cap \Upsilon$

[3] In Figure 2 and in the remainder of this article, we assume that ρ, ρ_1, and ρ_2 are functions from views in, respectively, E, E_1, and E_2 to subsets of, respectively, E, E_1, and E_2. Moreover, $\nabla, \Delta, \Upsilon \subseteq E$; $\nabla_1, \Delta_1, \Upsilon_1 \subseteq E_1$; and $\nabla_2, \Delta_2, \Upsilon_2 \subseteq E_2$ shall be sets of events.

[4] The definitions of *R, BSD, BSI, BSIA* are generalizations of respectively *RI, BSDI, BSII, BSIHAI* in [11] that result from using the concept of views [13]. *FCD* and *FCI* are novel BSPs.

[5] If $Adm^{\rho}_{\mathcal{V}}(Tr, \beta, c)$ holds then we say that c is *ρ-admissible* after the trace $\beta \in Tr$ for the view $\mathcal{V} = (V, N, C)$. *Adm* is defined by $Adm^{\rho}_{\mathcal{V}}(Tr, \beta, e) := (\exists \gamma \in E^*.\, \gamma.e \in Tr \wedge \gamma|_{\rho(\mathcal{V})} = \beta|_{\rho(\mathcal{V})})$.



$BSD_{\mathcal{V}_L^{LH}}(Tr) \wedge BSI_{\mathcal{V}_L^{LH}}(Tr)$
$BSD_{\mathcal{V}_L^{LH}}(Tr) \wedge BSIA^{\rho_C}_{\mathcal{V}_L^{LH}}(Tr)$
separability
forward correctability
$BSD_{\mathcal{V}_L^{LHI}}(Tr) \wedge BSI_{\mathcal{V}_L^{LHI}}(Tr) \wedge FCD^{I,\emptyset,I}_{\mathcal{V}_L^{LHI}}(Tr) \wedge FCI^{I,\emptyset,I}_{\mathcal{V}_L^{LHI}}(Tr)$
$BSD_{\mathcal{V}_L^{LH}}(Tr) \wedge BSIA^{\rho_{UI}}_{\mathcal{V}_L^{LH}}(Tr)$
nondeducibility on outputs
$BSD_{\mathcal{V}_L^{LH}}(Tr) \wedge BSIA^{\rho_E}_{\mathcal{V}_L^{LH}}(Tr)$
perfect security property
$R_{\mathcal{V}_L^{LH}}(Tr)$
noninference
generalized noninterference
$BSD_{\mathcal{V}_L^{LHI}}(Tr) \wedge BSI_{\mathcal{V}_L^{LHI}}(Tr)$
generalized noninference
$R_{\mathcal{V}_L^{LHI}}(Tr)$

Figure 3. Ordering of known security properties by implication

immediately before the occurrence of visible events from $V \cap \nabla$. In this respect, *FCI* is related to *FCD*.

Let us summarize a few useful facts about these BSPs.

Theorem 1 *Let ρ, ρ' be functions from views in E to subsets of E, $\mathcal{V} = (V, N, C)$ and $\mathcal{V}' = (V', N', C')$ be views in E, and $\nabla, \Delta, \Upsilon, \nabla', \Delta', \Upsilon' \subseteq E$ be sets of events.*

1. *$BSD_{\mathcal{V}}(Tr)$ implies $R_{\mathcal{V}}(Tr)$.*
2. *$BSI_{\mathcal{V}}(Tr)$ is equivalent to $BSIA^{\rho}_{\mathcal{V}}(Tr)$ and the condition that ES is total in C.*[6]
3. *If $N \subseteq \Delta$ then $BSI_{\mathcal{V}}(Tr)$ implies $FCI^{\nabla,\Delta,\Upsilon}_{\mathcal{V}}(Tr)$.*
4. *Assume $V \supseteq V'$ and $C \supseteq C'$.*
   (a) *$R_{\mathcal{V}}(Tr)$ and $BSD_{\mathcal{V}}(Tr)$, imply that $R_{\mathcal{V}'}(Tr)$ and $BSD_{\mathcal{V}'}(Tr)$ hold, respectively.*
   (b) *If $BSD_{\mathcal{V}}(Tr)$ then $BSI_{\mathcal{V}}(Tr)$ implies $BSI_{\mathcal{V}'}(Tr)$.*
   (c) *If $BSD_{\mathcal{V}}(Tr)$ and $\rho(\mathcal{V}) \subseteq \rho'(\mathcal{V}')$ hold then $BSIA^{\rho}_{\mathcal{V}}(Tr)$ implies $BSIA^{\rho'}_{\mathcal{V}'}(Tr)$.*
   (d) *If $BSD_{\mathcal{V}}(Tr)$, $\nabla' \subseteq \nabla$, $\Delta' \supseteq \Delta$, and $\Upsilon' \subseteq \Upsilon$ hold then $FCD^{\nabla,\Delta,\Upsilon}_{\mathcal{V}}(Tr)$ and $FCI^{\nabla,\Delta,\Upsilon}_{\mathcal{V}}(Tr)$ imply $FCD^{\nabla',\Delta',\Upsilon'}_{\mathcal{V}'}(Tr)$ and $FCI^{\nabla',\Delta',\Upsilon'}_{\mathcal{V}'}(Tr)$, respectively.*

2.4 Representing Known Security Properties

Let us demonstrate how some known security properties can be represented in *MAKS*. Basically, two popular classes of known security properties can be distinguished, depending on whether FP_{LH} or FP_{LHI} is enforced (cf. Figure 1). *Generalized noninterference* [16], *forward correctability* [9], and *generalized noninference* [19] all enforce the flow policy FP_{LHI}, i.e. only deductions about occurrences of high-level input events are prevented. *Nondeducibility on outputs* [8], *noninference* [24, 19], separability [19], and the *perfect security property* [36] all enforce the flow policy FP_{LH}. Note that in FP_{LH} and FP_{LHI}, proof obligations arise only, respectively, for the views $\mathcal{V}_L^{LH}$ and $\mathcal{V}_L^{LHI}$.

Theorem 2 *Let ρ_C, ρ_E, and ρ_{UI} be defined by $\rho_C(\mathcal{V}) = C$, $\rho_E(\mathcal{V}) = C \cup N \cup V$, and $\rho_{UI}(\mathcal{V}) = C \cup N \cup (V \cap UI)$ where $UI \subseteq I$ is a set of user input events that constitute the interface to the user.*

- Generalized noninference *as defined in [36]*[7] *is equivalent to $R_{\mathcal{V}_L^{LHI}}(Tr)$.*
- Generalized noninterference *is equivalent to $BSD_{\mathcal{V}_L^{LHI}}(Tr) \wedge BSI_{\mathcal{V}_L^{LHI}}(Tr)$.*
- Forward correctability *is equivalent to $BSD_{\mathcal{V}_L^{LHI}}(Tr) \wedge BSI_{\mathcal{V}_L^{LHI}}(Tr) \wedge FCD^{I,\emptyset,I}_{\mathcal{V}_L^{LHI}}(Tr) \wedge FCI^{I,\emptyset,I}_{\mathcal{V}_L^{LHI}}(Tr)$.*
- Noninference *is equivalent to $R_{\mathcal{V}_L^{LH}}(Tr)$.*
- Separability *is equivalent to $BSD_{\mathcal{V}_L^{LH}}(Tr) \wedge BSIA^{\rho_C}_{\mathcal{V}_L^{LH}}(Tr)$.*
- Nondeducibility on outputs *is equivalent to $BSD_{\mathcal{V}_L^{LH}}(Tr) \wedge BSIA^{\rho_{UI}}_{\mathcal{V}_L^{LH}}(Tr)$.*
- *The* perfect security property *is equivalent to $BSD_{\mathcal{V}_L^{LH}}(Tr) \wedge BSIA^{\rho_E}_{\mathcal{V}_L^{LH}}(Tr)$.*

The following corollary shows how these security properties can be ordered. It follows from Theorems 1 and 2.

Corollary 1 *Generalized noninference, generalized noninterference, forward correctability, noninference, nondeducibility on outputs, separability, and the perfect security property are ordered by implication as depicted in Figure 3.*

[6]*ES* is *total* in C if for all $c \in C$ and all $\tau \in Tr$ holds $\tau.c \in Tr$.

[7]The definitions of generalized noninference, noninference, and separability in [19] are based on sequences of states rather than sequences of events. Therefore, we use the event-based formalizations from [36].

3 Generalized Zipping Lemma

A zipping lemma allows one to construct a trace of a composed system by merging traces of the system components. This merging can be envisioned as closing a zipper where the component traces correspond to the two sides of a zipper and the resulting system trace to the closed zipper.

The technique to prove compositionality of a security property with the help of a zipping lemma was pioneered by Johnson and Thayer [9]. Their zipping lemma is only applicable to forward correctability. However, we found that the underlying idea of this proof technique is far more powerful. Below, we propose a generalization of Johnson and Thayer's zipping lemma that provides the basis for all compositionality results in this article.

Lemma 1 (Generalized Zipping Lemma) *For $i \in \{1,2\}$, assume that $V \cap E_i = V_i$ and $C \cap E_i \subseteq C_i$ hold. Moreover, assume that $N_1 \cap N_2 = \emptyset$ and one of the following four conditions hold:*

1. $N_1 \cap E_2 = \emptyset$ *and* $N_2 \cap E_1 = \emptyset$
2. $N_1 \cap E_2 = \emptyset$ *and* $BSI_{\mathcal{V}_1}(Tr_1)$
3. $N_2 \cap E_1 = \emptyset$ *and* $BSI_{\mathcal{V}_2}(Tr_2)$
4. *There are* $\nabla_1, \Delta_1, \Upsilon_1 \subseteq E_1$, $\nabla_2, \Delta_2, \Upsilon_2 \subseteq E_2$ *such that*
   - $BSI_{\mathcal{V}_1}(Tr_1)$, $BSI_{\mathcal{V}_2}(Tr_2)$
   - $FCI_{\mathcal{V}_1}^{\nabla_1,\Delta_1,\Upsilon_1}(Tr_1)$, $FCI_{\mathcal{V}_2}^{\nabla_2,\Delta_2,\Upsilon_2}(Tr_2)$;
   - $V_1 \cap V_2 \subseteq \nabla_1 \cup \nabla_2$;
   - $C_1 \cap N_2 \subseteq \Upsilon_1$, $C_2 \cap N_1 \subseteq \Upsilon_2$; *and*
   - $N_1 \cap \Delta_1 \cap E_2 = \emptyset$, $N_2 \cap \Delta_2 \cap E_1 = \emptyset$ *hold.*

If $\tau|_{E_1}.t_1 \in Tr_1$, $\tau|_{E_2}.t_2 \in Tr_2$, $\lambda|_{E_1} = t_1|_V$, $\lambda|_{E_2} = t_2|_V$, $t_1|_{C_1} = \langle\rangle$, and $t_2|_{C_2} = \langle\rangle$ hold for $\tau \in E^*$, $\lambda \in V^*$, $t_1 \in E_1^*$, $t_2 \in E_2^*$ then there is a sequence $t \in E^*$ with $\tau.t \in Tr$, $t|_V = \lambda$, and $t|_C = \langle\rangle$.

In the lemma, τ can be envisioned as the part of the zipper that has been closed already because $\tau \in Tr$ (follows from $\tau|_{E_1}.t_1 \in Tr_1$ and $\tau|_{E_2}.t_2 \in Tr_2$). t_1 and t_2 can be envisioned as parts of the zipper that are not yet closed. That the zipper can be closed completely is expressed by $\tau.t \in Tr$. The remaining conditions, i.e. $t|_V = \lambda$, $\lambda|_{E_1} = t_1|_V$, $\lambda|_{E_2} = t_2|_V$, $t|_C = \langle\rangle$, $t_1|_{C_1} = \langle\rangle$, and $t_2|_{C_2} = \langle\rangle$, ensure that the zipper is not modified in any essential way.

For a compositional security analysis it appears to be necessary that events that are visible or confidential wrt. the composed system have also been assumed to be, respectively, visible or confidential in the analysis of the components ($V \cap E_i = V_i$, $C \cap E_i \subseteq C_i$). Moreover, events used for corrections in the one component cannot be used for corrections by the other component ($N_1 \cap N_2 = \emptyset$). The four remaining conditions differ in whether corrections of one component have effects on the other one (1: no effects, 2 and 3: only in one direction, 4: in both directions).

Our generalized zipping lemma makes use of the building blocks of *MAKS*. Interestingly, only BSPs are demanded that are concerned with the insertion of events, i.e. *BSI* and *FCI*. However, upon applying the lemma, it might be sensible to require *R* or *BSD* additionally in order to satisfy the preconditions $t_1|_{C_1} = \langle\rangle$ and $t_2|_{C_2} = \langle\rangle$. For example, this will be the case in Theorem 3.

4 Preservation of Security Properties

In this section, we derive various compositionality results based on our generalized zipping lemma. In particular, we present compositionality results for all BSPs from Section 2.3. These results are used to re-justify several already known compositionality results. This approach leads to very simple justifications and, moreover, gives rise to a classification of known compositionality results.

Theorem 3 *If the preconditions of Lemma 1 are satisfied then the following propositions are valid:*

1. $R_{\mathcal{V}_1}(Tr_1) \wedge R_{\mathcal{V}_2}(Tr_2) \implies R_{\mathcal{V}}(Tr)$ *holds.*
2. $BSD_{\mathcal{V}_1}(Tr_1) \wedge BSD_{\mathcal{V}_2}(Tr_2) \implies BSD_{\mathcal{V}}(Tr)$ *holds.*
3. $BSI_{\mathcal{V}_1}(Tr_1) \wedge BSI_{\mathcal{V}_2}(Tr_2) \implies BSI_{\mathcal{V}}(Tr)$ *holds if, for $i \in \{1,2\}$, $BSD_{\mathcal{V}_i}(Tr_i)$.*
4. $BSIA_{\mathcal{V}_1}^{\rho_1}(Tr_1) \wedge BSIA_{\mathcal{V}_2}^{\rho_2}(Tr_2) \implies BSIA_{\mathcal{V}}^{\rho}(Tr)$ *holds if, for $i \in \{1,2\}$, $BSD_{\mathcal{V}_i}(Tr_i)$ and $\rho_i(\mathcal{V}_i) \subseteq \rho(\mathcal{V}) \cap E_i$.*
5. $FCD_{\mathcal{V}_1}^{\nabla_1,\Delta_1,\Upsilon_1}(Tr_1) \wedge FCD_{\mathcal{V}_2}^{\nabla_2,\Delta_2,\Upsilon_2}(Tr_2) \Rightarrow FCD_{\mathcal{V}}^{\nabla,\Delta,\Upsilon}(Tr)$ *holds if, for $i \in \{1,2\}$, $BSD_{\mathcal{V}_i}(Tr_i)$, $\Upsilon \cap E_i \subseteq \Upsilon_i$, $\nabla \cap E_i \subseteq \nabla_i$, $\Delta \supseteq ((\Delta_1 \cap N_1) \cup (\Delta_2 \cap N_2))$, $N_1 \cap \Delta_1 \cap E_2 = \emptyset$, and $N_2 \cap \Delta_2 \cap E_1 = \emptyset$.*
6. $FCI_{\mathcal{V}_1}^{\nabla_1,\Delta_1,\Upsilon_1}(Tr_1) \wedge FCI_{\mathcal{V}_2}^{\nabla_2,\Delta_2,\Upsilon_2}(Tr_2) \Rightarrow FCI_{\mathcal{V}}^{\nabla,\Delta,\Upsilon}(Tr)$ *holds if, for $i \in \{1,2\}$, ES_i is total in $C_i \cap \Upsilon_i$, $BSD_{\mathcal{V}_i}(Tr_i)$, $BSIA_{\mathcal{V}_i}^{\rho_i}(Tr_i)$, $\Upsilon \cap E_i \subseteq \Upsilon_i$, $\nabla \cap E_i \subseteq \nabla_i$, $\Delta \supseteq ((\Delta_1 \cap N_1) \cup (\Delta_2 \cap N_2))$, and*
   - $N_1 \cap \Delta_1 \cap E_2 = \emptyset$ *and* $N_2 \cap \Delta_2 \cap E_1 \subseteq \Upsilon_1$; *or*
   - $N_2 \cap \Delta_2 \cap E_1 = \emptyset$ *and* $N_1 \cap \Delta_1 \cap E_2 \subseteq \Upsilon_2$.

For the preservation of certain BSPs, Theorem 3 demands that other BSPs hold in addition. E.g., for preserving *BSI*, also *BSD* needs to hold. Technically, this additional BSP is required in order to satisfy the preconditions $t_1|_{C_1} = \emptyset$ and $t_2|_{C_2} = \emptyset$ of the generalized zipping lemma. E.g., for preserving *FCI*, also *BSD* and *BSIA* need to hold and, moreover, the components need to be total in certain events (recall from Theorem 1 that, alternatively, *BSI* can be required instead of *BSIA* and totality). In the remainder of this section, we will demonstrate by deriving various compositionality results that the additional BSPs are not a major obstacle when applying Theorem 3.

Note that in Proposition 6 of Theorem 3 only one of $N_1 \cap \Delta_1 \cap E_2 = \emptyset$ or $N_2 \cap \Delta_2 \cap E_1 = \emptyset$ needs to hold. This differs from Proposition 5 and also from Condition 4 of Lemma 1 in which both equations need to hold. This fact will be important for some results of Section 6.

4.1 A New Classification of Known Compositionality Results

Before deriving novel compositionality results with the help of our generalized zipping lemma in Sections 5 and 6, let us re-investigate some known compositionality results. Although these results have already been verified in [9, 8, 19, 20, 36], we find it appealing how easily these results can been justified in our setting. Moreover, the four conditions of our generalized zipping lemma provide a natural classification of known compositionality results. Depending on which of these conditions is satisfied, a compositionality result falls into one of three classes. There are three, rather than four, classes because the second and third condition of the generalized zipping lemma are symmetric. For the following discussion, recall the modular representation of security properties from Theorem 2.

First Class of Compositionality Results. Three main approaches to satisfy the first condition of Lemma 1, i.e. $N_1 \cap E_2 = \emptyset$ and $N_2 \cap E_1 = \emptyset$, can be distinguished: firstly, by ensuring $E_1 \cap E_2 = \emptyset$, which corresponds to restricting composition to product; secondly, by ensuring $N_1 = \emptyset = N_2$, which can be achieved, e.g., in the context of two security levels by preventing deductions not only about occurrences of high-level input but also of high-level output events; and, thirdly, by ensuring $N_1 = \emptyset$ by the security property and then restricting composition such that $N_2 \cap E_1 = \emptyset$ holds.

Following the first approach (restricted composition ensures $E_1 \cap E_2 = \emptyset$), we obtain from Theorem 3 that *all security properties that can be assembled from R, BSD, BSI, BSIA, FCD, and FCI are preserved under product* (for *BSI, BSIA, FCD, FCI* a few side conditions need to hold). A simple consequence of the modular representation of security properties (cf. Theorem 2) is that, e.g., *generalized noninference* (represented by $R_{\mathcal{V}_L^{LHI}}(Tr)$) *and generalized noninterference* ($BSD_{\mathcal{V}_L^{LHI}}(Tr) \wedge BSI_{\mathcal{V}_L^{LHI}}(Tr)$) *are preserved under product*, results first presented in [19].

Following the second approach (security property ensures $N_1 = \emptyset = N_2$), we obtain that *noninference* ($R_{\mathcal{V}_L^{LH}}(Tr)$), *separability* ($BSD_{\mathcal{V}_L^{LH}}(Tr) \wedge BSIA^{\rho_C}_{\mathcal{V}_L^{LH}}(Tr)$), *and the perfect security property* ($BSD_{\mathcal{V}_L^{LH}}(Tr) \wedge BSIA^{\rho_E}_{\mathcal{V}_L^{LH}}(Tr)$) *are preserved under arbitrary compositions*, results also presented in [19, 36]. For these results recall that $\mathcal{V}_L^{LH} = (L, \emptyset, H)$ and, hence, $N_1 = \emptyset = N_2$. The side condition $\rho_i(\mathcal{V}_i) \subseteq \rho(\mathcal{V}) \cap E_i$ of Proposition 4 in Theorem 3 holds because of the following equations:

$$\begin{aligned} \rho_C(\mathcal{V}_{L_i}^{LH}) &= H_i = H \cap E_i = \rho_C(\mathcal{V}_L^{LH}) \cap E_i \\ \rho_E(\mathcal{V}_{L_i}^{LH}) &= L_i \cup H_i = (L \cup H) \cap E_i = \rho_E(\mathcal{V}_L^{LH}) \cap E_i \end{aligned}$$

Nondeducibility on outputs ($BSD_{\mathcal{V}_L^{LH}}(Tr) \wedge BSIA^{\rho_{UI}}_{\mathcal{V}_L^{LH}}(Tr)$) *is preserved under composition if low-level user inputs are not connected*, a result from [8], because for $i \in \{1, 2\}$:

$$\begin{aligned} \rho_{UI}(\mathcal{V}_{L_i}^{LH}) &= H_i \cup (L_i \cap UI_i) \\ &= (H \cup (L \cap UI)) \cap E_i \\ &= \rho_{UI}(\mathcal{V}_L^{LH}) \cap E_i \end{aligned}$$

holds. Low-level user inputs must not be connected because, otherwise, $UI_i = UI \cap E_i$ would not hold.

Following the third approach (security property ensures $N_1 = \emptyset$, restricted composition ensures $N_2 \cap E_1 = \emptyset$), we obtain various conditional compositionality results. For example, *if an event system ES_1 that satisfies noninference* ($R_{\mathcal{V}_{L_1}^{LH}}(Tr_1)$) *is composed by a relaxed cascade* (ensures $E_1 \cap E_2 \subseteq O_1 \cap I_2$) *with an event system ES_2 that satisfies generalized noninference* ($R_{\mathcal{V}_{L_2}^{LHI}}(Tr_2)$) *then* $N_2 \cap E_1 = \emptyset$ is satisfied because $\mathcal{V}_{L_2}^{LHI} = (L_2, H_2 \setminus I_2, H_2 \cap I_2)$. According to Theorem 3, *the resulting event system* satisfies $R_{V,N,C}(Tr)$, where $V = L_1 \cup L_2$, $N = H_2 \setminus I_2$, $C = H_1 \cup (H_2 \cap I_2)$. This implies that *ES satisfies generalized noninference* ($R_{\mathcal{V}_L^{LHI}}(Tr)$), a result also presented in [19].

Second Class of Compositionality Results. Compositionality results that fall into this class, satisfy either the second or the third condition of the generalized zipping lemma. Since these conditions are symmetric, we focus only on the third condition ($N_2 \cap E_1 = \emptyset$, $BSI_{\mathcal{V}_2}(Tr_2)$) below.

The condition $N_2 \cap E_1 = \emptyset$ can be satisfied by first restricting composition to relaxed cascade (ensures $E_1 \cap E_2 \subseteq O_1 \cap I_2$) and then choosing a security property for the second component that ensures that no input events are contained in N_2, i.e. $I_2 \cap N_2 = \emptyset$. Results along these lines include that *generalized noninterference* ($BSD_{\mathcal{V}_L^{LHI}}(Tr) \wedge BSI_{\mathcal{V}_L^{LHI}}(Tr)$) *is preserved under relaxed cascade*, a result also presented in [19, 34]. Moreover, *if a system ES_1 that satisfies generalized noninterference is composed in a relaxed cascade with an input total system ES_2 that satisfies separability then the resulting system satisfies generalized noninterference*. For this result, recall from Theorem 1 that $BSIA^{\rho_2}_{\mathcal{V}_{L_2}^{LHI}}(Tr_2)$ (follows from $BSD_{\mathcal{V}_{L_2}^{LH}}(Tr_2)$ and

$BSIA^{\rho_2}_{\mathcal{V}^{LH}_{L_2}}(Tr_2))$ and totality in I_2 imply $BSI_{\mathcal{V}^{LHI}_{L_2}}(Tr_2)$. *If a system that satisfies generalized noninference* $(R_{\mathcal{V}^{LHI}_{L_1}}(Tr_1))$ *is composed in a relaxed cascade with a system that satisfies generalized noninterference or is input total and satisfies separability then the resulting system satisfies generalized noninference*, results first presented in [19].

Third Class of Compositionality Results. That forward correctability is composable, in general, can be explained using the fourth condition of the generalized zipping lemma. Recall from Theorem 2 that forward correctability is equivalent to
$BSD_{\mathcal{V}^{LHI}_L}(Tr) \wedge BSI_{\mathcal{V}^{LHI}_L}(Tr) \wedge FCD^{I,\emptyset,I}_{\mathcal{V}^{LHI}_L}(Tr) \wedge FCI^{I,\emptyset,I}_{\mathcal{V}^{LHI}_L}(Tr)$. Theorem 3 is applicable to forward correctability because all side conditions are satisfied for $V_i = L_i$, $\nabla_i = I_i$, $C_i = H_i \cap I_i$, $\Delta_i = \emptyset$, and $\Upsilon_i = I_i$ $(i \in \{1,2\})$. Therefore, *forward correctability is preserved under arbitrary composition*, a result, first presented in [9].

Our classification above reveals similarities between previously unrelated compositionality results. E.g., generalized noninference is preserved under product for the same reasons that ensure the preservation of noninference under arbitrary compositions. Unlike in previous classifications like, e.g., [19], we do not classify merely depending on restrictions that are imposed on composition like, e.g., product or cascade. Rather, we classify depending on the effect of such restrictions.

5 Weakened Forward Correctability

In this section, we show that Johnson and Thayer's forward correctability [9] is not the weakest information flow property possible that is preserved under arbitrary compositions and that does not impose any restrictions on the occurrence of high-level output events.[8]

According to Theorem 2, forward correctability is equivalent to $BSD_{\mathcal{V}^{LHI}_L}(Tr) \wedge BSI_{\mathcal{V}^{LHI}_L}(Tr) \wedge FCD^{I,\emptyset,I}_{\mathcal{V}^{LHI}_L}(Tr) \wedge FCI^{I,\emptyset,I}_{\mathcal{V}^{LHI}_L}(Tr)$. This modular representation is now used to analyze how the restrictions of forward correctability can be relaxed while retaining compositionality. Firstly, observe that in order to satisfy Condition 1, 2, or 3 of the generalized zipping lemma, it is necessary to either restrict composition (e.g. to product or cascade) or to prevent deductions about high-level output events (e.g, by using flow policy FP_{LH} rather than FP_{LHI}). Consequently, these conditions cannot be satisfied by a compositional security property that does not prevent deductions about high-level outputs. However, such a security property may satisfy Condition 4 of the generalized zipping lemma. According to this condition, *BSI* and *FCI* must be satisfied. Moreover, certain side conditions must be satisfied for the parameters ∇, Δ, Υ. Recall from Theorem 1 that for a weak security property, the sets ∇ and Υ should be chosen as small as possible while Δ should be chosen as large as possible. An appropriate choice of ∇, Υ, and Δ that satisfies all side conditions from Proposition 4 of Lemma 1 is $\nabla = I$, $\Delta = E \setminus (I \cup O)$, $\Upsilon = I$. Note that the side conditions, indeed, are fulfilled: $(L_1 \cap L_2) \subseteq (I_1 \cup I_2)$, $(HI_1 \cap (H_2 \setminus HI_2)) \subseteq I_1$, $(HI_2 \cap (H_1 \setminus HI_1)) \subseteq I_2$, $N_1 \cap (E_1 \setminus (I_1 \cup O_1)) \cap E_2 = \emptyset$, and $N_2 \cap (E_2 \setminus (I_2 \cup O_2)) \cap E_1 = \emptyset$. Hence, it suffices to require $FCI^{I,E\setminus(I\cup O),I}_{\mathcal{V}^{LHI}_L}(Tr)$ instead of the stronger $FCI^{I,\emptyset,I}_{\mathcal{V}^{LHI}_L}(Tr)$. If *BSD* is required in addition to *BSI* and *FCI* then all preconditions of Propositions 2, 3, and 6 in Theorem 3 are fulfilled because $I \cap E_i \subseteq I_i$ holds for $i \in \{1,2\}$ and $E \setminus (I \cup O) \supseteq \bigcup_{i\in\{1,2\}}((E_i \setminus (I_i \cup O_i)) \cap (H_i \setminus HI_i))$. Apparently, there is no need to require *FCD*. This results in the following definition of a novel security property that is preserved under arbitrary compositions.

Definition 2 *We define* weakened forward correctability *by*

$$BSD_{\mathcal{V}^{LHI}_L}(Tr) \wedge BSI_{\mathcal{V}^{LHI}_L}(Tr) \wedge FCI^{I,E\setminus(I\cup O),I}_{\mathcal{V}^{LHI}_L}(Tr) \ .$$

Theorem 4 (Composability) *If ES_1 and ES_2 satisfy weakened forward correctability then $ES = ES_1 \parallel ES_2$ satisfies weakened forward correctability.*

In order to show that weakened forward correctability is, indeed, weaker than forward correctability, it remains to prove that $FCI^{I,\emptyset,I}_{\mathcal{V}^{LHI}_L}(Tr)$ and $FCD^{I,\emptyset,I}_{\mathcal{V}^{LHI}_L}(Tr)$ are not implied by weakened forward correctability. For $FCI^{I,\emptyset,I}_{\mathcal{V}^{LHI}_L}(Tr)$ this is obvious from the definition of *FCI*. For $FCD^{I,\emptyset,I}_{\mathcal{V}^{LHI}_L}(Tr)$, consider a system with the following possible traces: $Tr = \{hi^*,\ hi.hi^*.li.hi^*,\ ho.hi^*,\ ho.hi^*.li.hi^*\}$ where hi^* denotes a (possibly empty) sequence of events *hi* of arbitrary length and *hi*, *ho*, *li* shall be, respectively, high-level input, high-level output, and low-level input events. This system satisfies $BSD_{\mathcal{V}^{LHI}_L}(Tr)$, $BSI_{\mathcal{V}^{LHI}_L}(Tr)$ and $FCI^{I,\emptyset,I}_{\mathcal{V}^{LHI}_L}(Tr)$. However, it does not satisfy $FCD^{I,\emptyset,I}_{\mathcal{V}^{LHI}_L}(Tr)$ because if *hi* is deleted in the trace *hi.li* then the only possible correction, i.e. *ho.li*, requires the insertion of *ho* before *li*, however, this is not allowed by $FCD^{I,\emptyset,I}_{\mathcal{V}^{LHI}_L}(Tr)$.

Hence, weakened forward correctability is a composable security property that is weaker than forward correctability. Our results above are in contrast to the following statement from [35, page 100] "*forward correctability is the weakest*

[8]Recall the ordering of security properties from Corollary 1 and that generalized noninterference is, in general, not preserved under arbitrary compositions. Also note that the perfect security property, nondeducibility on outputs, and separability all impose restrictions on deductions about occurrences of output events (cf. Theorem 2).

condition of any [composable] property that solely eliminates the possible of there being a condition on a low-level input event". Note that in Definition 2, $\nabla \cap V$ contains only low-level input events because $\nabla \cap V = I \cap L$ holds.

Since weaker properties are easier to verify, weakened forward correctability seems to be, in general, preferable to forward correctability. Verification techniques for weakened forward correctability are outside the scope of the current article. However, the derivation of unwinding results appears to be straightforward by combining the ideas from [21] and [12]. We expect that our weakening of the security property also results in weaker unwinding conditions.

6 Properties Emerging During Composition

Verifying that a given system specification satisfies an information flow property means to prove that the set of possible traces is closed under some closure condition. Usually, so called unwinding results are used during verification (e.g. [7, 27, 21, 11]). However, information flow properties can also be verified directly (e.g. [18]). The elaboration of these verification techniques has simplified the verification of information flow properties substantially but, nevertheless, the effort to verify information flow properties can be considerable. Therefore, it is very appealing that some information flow properties hold trivially and, more interestingly, that certain nontrivial information flow properties emerge during composition.

6.1 Trivially Satisfied Security Properties

Theorem 5 *Let* $\nabla, \Delta, \Upsilon \subseteq E$ *be arbitrary.*

1. *If* $C = \emptyset$ *then* $R_{\mathcal{V}}(Tr)$, $BSD_{\mathcal{V}}(Tr)$, $BSI_{\mathcal{V}}(Tr)$, $BSIA^{\rho}_{\mathcal{V}}(Tr)$, $FCD^{\nabla,\Delta,\Upsilon}_{\mathcal{V}}(Tr)$, $FCI^{\nabla,\Delta,\Upsilon}_{\mathcal{V}}(Tr)$ *hold.*
2. (a) *If* $V = \emptyset$ *then* $R_{\mathcal{V}}(Tr)$, $BSD_{\mathcal{V}}(Tr)$, $FCD^{\nabla,\Delta,\Upsilon}_{\mathcal{V}}(Tr)$, *and* $FCI^{\nabla,\Delta,\Upsilon}_{\mathcal{V}}(Tr)$ *hold.*

   (b) *If* $V = \emptyset$ *and* $\rho(\mathcal{V}) \supseteq C \cup N$ *then* $BSIA^{\rho}_{\mathcal{V}}(Tr)$ *holds.*

   (c) *If* $V = \emptyset$ *and ES is total in* C *then* $BSI_{\mathcal{V}}(Tr)$ *holds.*
3. *If* $\nabla = \emptyset$ *or* $\Upsilon = \emptyset$ *then* $FCD^{\nabla,\Delta,\Upsilon}_{\mathcal{V}}(Tr)$ *and* $FCI^{\nabla,\Delta,\Upsilon}_{\mathcal{V}}(Tr)$ *hold.*

For systems that either do not operate on confidential data or whose behavior does not result in any visible output, one might already expect that most security properties are satisfied. The first two propositions in the above theorem show that this is, indeed, the case. Proposition 1 says that systems that do not engage in any confidential events satisfy all BSPs from Section 2.3. According to Propositions 2a–2c, these BSPs are also satisfied by systems that do not engage in any visible events (for *BSIA* and *BSI* under certain restrictions). Although these facts are intuitively not too astonishing and also technically quite trivial, they can be helpful when building secure systems by composition.

In particular, untrusted programs (potentially Trojan horses) may be connected to the high-level interface of a security critical system without verification under the condition that they cannot directly engage in events that are visible to low-level adversaries. Without explicit verification, most BSPs are trivially satisfied by such components. Note that the security properties that are trivially satisfied include separability, nondeducibility on outputs, the perfect security property, noninference, and generalized noninference (cf. Figure 3). Moreover, if an (untrusted) program is input total then forward correctability and generalized noninterference are also satisfied.

To restrict unverified components to only either operate on confidential data or to provide output to potential adversaries, seems to be a necessity for secure systems.[9] Components that engage in confidential as well as visible events need to be verified to be secure. Minimizing the number of such critical components reduces the verification effort and also appears to be good design practice for secure systems anyway. Upon composition, the security of the overall system should be derived from security properties satisfied by the components (trivially satisfied ones as well as ones that have been proved explicitely) with the help of compositionality results (cf. Section 3–5).

6.2 Nontrivial Security Properties Emerging from Trivial Ones

Security properties that are trivially satisfied by system components can be exploited to demonstrate that a composed system satisfies nontrivial security properties. Technically, the emergent properties follow from the trivially satisfied properties (cf. Theorems 1 and 5) together with compositionality results like the ones in Theorem 3. It is quite appealing that some of the emergent properties are nontrivial in the sense that they would not be known to be satisfied by the composed system if that system is considered as a black box (as for the trivially satisfied BSPs).

Let us illustrate how this phenomenon can be exploited at the example of *FCI*. For this purpose assume a system that is composed of several components that each satisfy *BSD* as well as *BSI*. Moreover, for any two components, the second or third condition of the generalized zip-

[9] A computationally inexpensive method to verify untrusted programs is the use of security type systems (e.g. [1, 2, 30, 29]). The connection between language-based security and trace-based security properties has been explored, e.g., in [22].

ping lemma shall be satisfied. Consequently, we can conclude from Theorem 3 that for the composition of any *two* components, *BSD* and *BSI* are also satisfied. However, if the system consists of more than two components, this does not explain why the overall system also satisfies *BSD* and *BSI*. The problem is that neither the second nor the third condition of the generalized zipping lemma need to be satisfied after composing two components. For example, consider a system that consists of three components ES_1, ES_2, and ES_3 with $E_1 = \{h_1, h_{12}, h_{13}, l_1, l_{12}, l_{13}\}$, $E_2 = \{h_2, h_{12}, h_{23}, l_2, l_{12}, l_{23}\}$, $E_3 = \{h_3, h_{13}, h_{23}, l_3, l_{13}, l_{23}\}$ and the views $\mathcal{V}_1 = (\{l_1, l_{12}, l_{13}\}, \{h_{12}\}, \{h_1, h_{13}\}))$, $\mathcal{V}_2 = (\{l_2, l_{12}, l_{23}\}, \{h_{23}\}, \{h_2, h_{12}\}))$, $\mathcal{V}_3 = (\{l_3, l_{13}, l_{23}\}, \{h_{13}\}, \{h_3, h_{23}\}))$. Note that $N_2 \cap E_1 = \{h_{23}\} \cap E_1 = \emptyset$, $N_3 \cap E_2 = \{h_{13}\} \cap E_2 = \emptyset$, and $N_1 \cap E_3 = \{h_{12}\}$ hold for the components. After composing, e.g., ES_1 and ES_2, we have $N_{12} \cap E_3 \subseteq (N_1 \cup N_2) \cap E_3 = \{h_{23}\} \neq \emptyset$ and $N_3 \cap E_{12} = N_3 \cap (E_1 \cup E_2) = \{h_{13}\} \neq \emptyset$. I.e. neither the second nor the third condition of the generalized zipping lemma are fulfilled for ES_3 and the composition of ES_1 and ES_2. Nevertheless, our compositionality results can be applied to show that the overall system satisfies *BSD* and *BSI*. The reason is that the fourth condition of the generalized zipping lemma holds after composing any number of components. This is demonstrated by the following theorem.

Theorem 6 *Let J be a nonempty index set. For every $j \in J$, let $ES_j = (E_j, I_j, O_j, Tr_j)$ be an event system such that $I_j \cap O_j = \emptyset$ and, for all $j, k \in J$ with $j \neq k$, $E_j \cap E_k \subseteq ((O_j \cap I_k) \cup (O_k \cap I_j))$ hold. Let $ES = (E, I, O, Tr)$ be the composition of these event systems, i.e. $ES = \|_{j \in J} ES_j$.*

Let $\mathcal{V} = (V, N, C)$ be a view in E. For every $j \in J$, let $\mathcal{V}_j = (V_j, N_j, C_j)$ be a view in E_j such that $V \cap E_j = V_j$ and $C \cap E_j \subseteq C_j$ hold.

If, for every $j \in J$, $BSD_{\mathcal{V}_j}(Tr_j)$, $BSI_{\mathcal{V}_j}(Tr_j)$, and, for every $k \in J$ with $k \neq j$, $N_j \cap E_k = \emptyset$ or $N_k \cap E_j = \emptyset$ holds then $BSD_{\mathcal{V}}(Tr)$, $BSI_{\mathcal{V}}(Tr)$, and, for every $j \in J$, holds:

1. $FCI_{\mathcal{V}}^{E_j \setminus (\bigcup_{k \in J, k \neq j} E_k),\, E_j,\, E_j \setminus (\bigcup_{k \in J, k \neq j} E_k)}(Tr)$;
2. $FCI_{\mathcal{V}}^{(\bigcup_{k \in J, k \neq j} E_k) \setminus E_j,\, \emptyset,\, E_j \setminus (\bigcup_{k \in J, k \neq j} E_k)}(Tr)$; *and*
3. $FCI_{\mathcal{V}}^{E_j \setminus (\bigcup_{k \in J, k \neq j} E_k),\, \emptyset,\, (\bigcup_{k \in J, k \neq j} E_k) \setminus E_j}(Tr)$.

Theorem 6 shows that *BSD*, *BSI*, and *FCI* hold for the composition of any number of components that satisfy the above restrictions. Note that if J contains more than one element then *FCI* does not hold trivially for the choices of ∇, Δ, Υ in the Theorem. Nevertheless, it is neither necessary to verify *FCI* explicitely for the overall system nor for any of the components. The satisfaction of *FCI* follows merely from our compositionality results and the trivial satisfaction of *FCI* for each component if $\nabla = \emptyset$, $\Upsilon = \emptyset$, or $\Delta \supseteq N_i$ (cf. Proposition 3 in Theorem 1 and Proposition 3 in Theorem 5).

Example 1 *For the example components ES_1, ES_2, and ES_3 that we introduced before, Theorem 6 ensures that, e.g.,*

$$FCI_{\mathcal{V}_{12}}^{\{l_1, l_{13}, h_1, h_{13}\},\, E_1,\, \{l_1, l_{13}, h_1, h_{13}\}}(Tr_{12}) \text{ and } \quad (1)$$

$$FCI_{\mathcal{V}_{12}}^{\{l_2, l_{23}, h_2, h_{23}\},\, \emptyset,\, \{l_1, l_{13}, h_1, h_{13}\}}(Tr_{12}) \quad (2)$$

hold after composing ES_1 and ES_2.

$$FCI_{\mathcal{V}_{12}}^{\{l_1, l_{13}, l_2, l_{23}, h_1, h_{13}, h_2, h_{23}\},\, E_1,\, \{l_1, l_{13}, h_1, h_{13}\}}(Tr_{12})$$

(follows from Formulas 1 and 2) and $FCI_{\mathcal{V}_3}^{\emptyset,\, \emptyset,\, \{h_{23}\}}(Tr_3)$ (holds trivially) ensure that Condition 4 of the generalized zipping lemma is fulfilled for $ES_1 \parallel ES_2$ and ES_3 (leaving the detailed check as a straightforward exercise). Consequently, $ES_1 \parallel ES_2 \parallel ES_3$ satisfies BSD, BSI, and FCI (for choices of ∇, Δ, Υ that are in accordance with Theorem 6).

7 Related Work

That possibilistic information flow properties need not be preserved under composition has been first demonstrated by McCullough with his well-known example that generalized noninterference is not preserved under arbitrary compositions [16]. The same example can also be used to prove that Sutherland's nondeducibility [31], i.e. the first generalization of Goguen and Meseguer's noninterference [6] for nondeterministic systems, is not compositional. The first information flow property that is preserved under arbitrary compositions was McCullough's restrictiveness [16, 17]. Johnson and Thayer found that restrictiveness is stronger than necessary in order to be composable. As a solution, they proposed forward correctability, an information flow property that is preserved under arbitrary compositions although it is weaker than restrictiveness and, hence, easier to verify [9]. Since forward correctability is, in general, superior to restrictiveness the latter property has been made obsolete. Guttman and Nadel proposed nondeducibility on outputs, another information flow property that is preserved under composition (with minor restrictions, cf. Section 4.1) [8]. The reason for the compositionality of this property is that deductions about high-level outputs are prevented (in addition to high-level inputs). This differs considerably from the requirements imposed by forward correctability or restrictiveness, which only prevent deductions about high-level inputs. Interestingly, this difference also becomes apparent in our classification of known compositionality results in Section 4.1. While the compositionality results for nondeducibility on outputs belongs to the first class, the compositionality of forward correctability belongs to the third class. Other compositionality results in the first class are the compositionality of noninference [24, 19], of separability [19], and of the perfect security property [36].

All results mentioned so far are concerned with information flow properties that are preserved under *arbitrary* compositions. McLean pioneered another line of research that is driven by the question of *how to restrict composition* in order to preserve a given information flow property. In Section 4.1 we have already restated McLean's results along these lines [19, 20] in our setting. Compositionality results based on restricted forms of composition have also been developed by Zakinthinos and Lee in a series of publications [34, 35, 37]. In [34], it has been demonstrated that McCullough's version of generalized noninterference is preserved if feedback loops are avoided (for the compositionality of a slightly different version cf. [19]). More interestingly, generalized noninterference is preserved if a delay component is inserted into all feedback loops that involve high-level events. These components need to delay any feedback to after the next low-level event. According to [35], generalized noninterference is also preserved if every feedback path contains at least three components. If there are no 2-cycles then the amount of delay is immaterial and, hence, a delay of feedback to after the next low-level event is not necessary.[10] A second compositionality theorem is somewhat ambiguous because several notions are used without being formalized.[11] This theorem led to the statement that forward correctability would be the "weakest condition of any [composable] property that solely eliminates the possible of there being a condition on a low-level input event" [35, page 100], which is in contrast to our results in Section 5.

The intuition underlying nondeducibility on strategies [33] is that a secure system should make it impossible for a low-level user to distinguish different strategies employed by a high-level user. Consequently, any system that fulfills nondeducibility on strategies prevents an unverified high-level program (potential Trojan horse) from transmitting confidential information to a low-level adversary. This setting is similar to the one that we have investigated in Section 6.1. The reformulation of nondeducibility on strategies into other system models (the system model used in [33] is somewhat nonstandard) has shown that, in a trace-based model, nondeducibility on strategies is equivalent to noninference [5]. Interestingly, Schneider has derived an information flow property, "may-NI", that is also equivalent to noninference although he has started from may testing, which is a quite different perspective [28]. In that article, compositionality results are presented for various forms of composition, including ones that permit the synchronization of high-level events of one component with low-level events of other components.[12]

Emergent properties have already been investigated in [37]. Technically, the results presented can be regarded as combinations of compositionality results with an ordering of security properties by implication. E.g., if two systems satisfy, respectively, security properties P_1 and P_2 then the product of these systems satisfies any security property P that is product-composable and for which $P_1 \implies P$ as well as $P_2 \implies P$ hold. Similar results, that are, however, technically more involved have been presented for cascade and unrestricted composition. Results of a similar flavor can also be found in [19], however, the term emergent properties is not used in that article.

The observation that a uniform framework can be helpful in the derivation of compositionality results has also been made by McLean. In [19, 20], various compositionality results have been derived based on the elegant framework of selective interleaving functions. That all external compositionality results from these articles can be restated in our setting has been demonstrated in Section 4.1. However, our compositionality results go beyond the ones derived in the framework of selective interleaving functions. One reason for this might be that selective interleaving functions are not well suited for expressing inductively defined properties like, e.g., forward correctability, restrictiveness, or McCullough's original formalization of generalized noninterference (the formalization in [19] differs in that corrections are allowed before the first perturbation).

For comparisons of known information flow properties and further frameworks for such properties, we refer to [5, 36, 25]. Although, compositionality issues have also been investigated in these articles, no benefit has been taken from a uniform representation to simplify the derivation of compositionality results (unlike in the work of McLean or in the current article).

In this article, we have focused on information flow properties in a trace-based system model that abstracts from probabilities. For compositionality results based on other possibilistic system models, we refer to [5, 26, 25]. For compositionality results for a probabilistic system model that, however, is only suitable for synchronous systems, we refer to [10].

8 Conclusion

The behavior of information flow properties under composition has been an important topic of security research

[10]This astonishing no-2-cycle result can also be obtained as a special case of our Theorem 6.

[11]Theorem 2 in [35] is based on the following notions: "high-level state", "a condition to be true at a given point of time", "a condition to *become* false", and "a condition to be *made* true". However, these notions are neither formally defined in terms of the computational model nor does their meaning become entirely clear from the context.

[12]The side conditions of Lemma 1 and Theorem 3 rule out synchronization of confidential events in C with visible events in V. However, according to Proposition 4 in Theorem 1, all BSPs remain valid when the view is modified by moving events from V to N. After all critical events have been moved, Lemma 1 and Theorem 3 become applicable.

for the last fifteen years and numerous interesting and useful insights have been achieved. In this article, we have provided a uniform perspective on compositionality results for information flow properties. In our presentation, uniformity occurs at two levels: firstly, *MAKS* [11, 12, 13] has been used for the uniform representation of information flow properties and, secondly, we have proposed a powerful lemma that can be used to justify various compositionality results in a uniform way. This generalized zipping lemma is the main technical contribution of this article. Based on this lemma, we have derived a classification of compositionality results for information flow properties that, unlike previous classifications, classifies depending on the *effects* of restrictions on the composition rather than only depending on the restriction itself. Several known compositionality results have been re-justified in order to classify them. We have also proposed a novel security property, "weakened forward correctability", that is weaker than Johnson and Thayer's forward correctability [9] but still is composable. Hence, weakened forward correctability makes forward correctability obsolete for the same reason why forward correctability made restrictiveness [16] obsolete. Moreover, we have demonstrated how to reduce the verification effort when building secure systems by choosing a compositional design that ensures that many components trivially fulfill the security properties of interest. Finally, we have illustrated that nontrivial security properties emerge under composition.

Future and ongoing work is focused on the application of the results presented in this article, in particular in the context of programming languages [22, 23].

Acknowledgments. The author would like to thank Dieter Hutter, Serge Autexier, Alexandra Heidger, and Axel Schairer for their valuable comments on an earlier version of this article. This work has been partly supported by the German Research Foundation (DFG).

References

[1] M. Abadi, A. Banerjee, N. Heintze, and J. Riecke. A Core Calculus of Dependency. In *ACM Symposium on Principles of Programming Languages*, 1999.

[2] J. Agat. Transforming out Timing Leaks. In *ACM Symposium on Principles of Programming Languages*, pages 40–53, 2000.

[3] M. Abadi and L. Lamport. Conjoining Specifications. *ACM Transactions on Programming Languages and Systems*, 17(3):507–534, 1995.

[4] B. Alpern and F. B. Schneider. Defining Liveness. Information Processing Letters, 21:181–185, 1985.

[5] R. Focardi and R. Gorrieri. A Classification of Security Properties for Process Algebras. *Journal of Computer Security*, 3(1):5–33, 1995.

[6] J. A. Goguen and J. Meseguer. Security Policies and Security Models. In *IEEE Symposium on Security and Privacy*, pages 11–20, 1982.

[7] J. A. Goguen and J. Meseguer. Inference Control and Unwinding. In *IEEE Symposium on Security and Privacy*, pages 75–86, 1984.

[8] J. D. Guttman and M. E. Nadel. "What Needs Securing?". In *IEEE Computer Security Foundations Workshop*, pages 34–57, 1988.

[9] D. M. Johnson and F. J. Thayer. Security and the Composition of Machines. In *IEEE Computer Security Foundations Workshop*, pages 72–89, 1988.

[10] J. Jürjens. Secure Information Flow for Concurrent Processes. In *International Conference on Concurrency Theory, Concur 2000*, LNCS 1877, pages 395–409, 2000.

[11] H. Mantel. Possibilistic Definitions of Security – An Assembly Kit. In *IEEE Computer Security Foundations Workshop*, pages 185–199, 2000.

[12] H. Mantel. Unwinding Possibilistic Security Properties. In *European Symposium on Research in Computer Security (ESORICS)*, LNCS 1895, pages 238–254, 2000.

[13] H. Mantel. Information Flow Control and Applications – Bridging a Gap. In *FME 2001: Formal Methods for Increasing Software Productivity*, LNCS 2021, pages 153–172, 2001.

[14] H. Mantel. Preserving Information Flow Properties under Refinement. In *IEEE Symposium on Security and Privacy*, pages 78–91, 2001.

[15] J. Misra and K. M. Chandy. Proofs of Networks of Processes. *IEEE Transactions on Software Engineering*, SE-7(4):417–426, 1981.

[16] D. McCullough. Specifications for Multi-Level Security and a Hook-Up Property. In *IEEE Symposium on Security and Privacy*, pages 161–166, 1987.

[17] D. McCullough. A Hookup Theorem for Multilevel Security. *IEEE Transactions on Software Engineering*, 16(6), 1990.

[18] J. McLean. Proving Noninterference and Functional Correctness using Traces. *Journal of Computer Security*, 1(1):37–57, 1992.

[19] J. McLean. A General Theory of Composition for Trace Sets Closed under Selective Interleaving Functions. In *IEEE Symposium on Research in Security and Privacy*, pages 79–93, 1994.

[20] J. McLean. A General Theory of Composition for a Class of "Possibilistic" Security Properties. *IEEE Transaction on Software Engineering*, 22(1):53–67, 1996.

[21] J. K. Millen. Unwinding Forward Correctability. In *Computer Security Foundations Workshop*, pages 2–10, 1994.

[22] H. Mantel and A. Sabelfeld. A Generic Approach to the Security of Multi-threaded Programs. In *IEEE Computer Security Foundations Workshop*, pages 126–142, 2001.

[23] H. Mantel and A. Sabelfeld. A Unifying Approach to the Security of Distributed and Multi-threaded Programs. Submitted, 2001.

[24] C. O'Halloran. A Calculus of Information Flow. In *European Symposium on Research in Computer Security (ESORICS)*, pages 147–159, 1990.

[25] P. Y. A. Ryan and S. A. Schneider. Process Algebra and Non-interference. In *IEEE Computer Security Foundations Workshop*, pages 214–227, 1999.

[26] A. W. Roscoe and L. Wulf. Composing and Decomposing Systems under Security Properties. In *IEEE Computer Security Foundations Workshop*, pages 9–15, 1995.

[27] P. Y. A. Ryan. A CSP Formulation of Non-Interference and Unwinding. *Cipher*, pages 19–30, Winter 1991.

[28] S. Schneider. May Testing, Non-interference, and Compositionality. Technical Report CSD-TR-00-02, Royal Holloway, University of London, January 2001.

[29] G. Smith. A New Type System for Secure Information Flow. In *IEEE Computer Security Foundations Workshop*, pages 115–125, 2001.

[30] A. Sabelfeld and D. Sands. Probabilistic Noninterference for Multi-threaded Programs. In *IEEE Computer Security Foundations Workshop*, pages 200–215, 2000.

[31] D. Sutherland. A Model of Information. In *National Computer Security Conference*, 1986.

[32] J. Widom, D. Gries, and F. B. Schneider. Trace-Based Network Proof Systems: Expressiveness and Completeness. *ACM Transactions on Programming Languages and Systems*, 14(3):396–416, 1992.

[33] J. T. Wittbold and D. M. Johnson. Information Flow in Nondeterministic Systems. In *IEEE Symposium on Research in Security and Privacy*, pages 144–161, 1990.

[34] A. Zakinthinos and E. S. Lee. The Composability of Non-Interference. In *IEEE Computer Security Foundations Workshop*, pages 2–8, 1995.

[35] A. Zakinthinos and E. S. Lee. How and Why Feedback Composition Fails. In *IEEE Computer Security Foundations Workshop*, pages 95–101, 1996.

[36] A. Zakinthinos and E. S. Lee. A General Theory of Security Properties. In *IEEE Symposium on Security and Privacy*, pages 94–102, 1997.

[37] A. Zakinthinos and E. S. Lee. Composing Secure Systems that have Emergent Properties. In *IEEE Computer Security Foundations Workshop*, pages 117–122, 1998.

Appendix

This appendix contains proof sketches of the main results presented in the article. Detailed proofs of these results can be obtained from the author upon request.

Proof. [Sketch of Lemma 1] We make a case distinction on the four conditions in the lemma. In the proofs of the first three cases, the following facts are used: $N_1 \cap V_2 = \emptyset$, $N_2 \cap V_1 = \emptyset$, $N_1 \cap E_2 = \emptyset$ implies $t_1|_{E_2} = \lambda|_{E_1 \cap E_2}$, and $N_2 \cap E_1 = \emptyset$ implies $t_2|_{E_1} = \lambda|_{E_1 \cap E_2}$. The first case follows directly from these conditions. In the proof of the second case, $BSI_{\mathcal{V}_1}(Tr_1)$ is applied in order to insert the sequence $t_2|_{N_2 \cap C_1}$ into $\tau|_{E_1}.t_1$. In the proof of the third case, $BSI_{\mathcal{V}_2}(Tr_2)$ is applied in order to insert the sequence $t_1|_{N_1 \cap C_2}$ into $\tau|_{E_2}.t_2$. The proof of the fourth case is the most difficult one. This proof proceeds along similar lines as the proof of the zipping lemma in [9] but differs in technical details:

An induction on the length of λ is performed. In the base case, the proposition holds for the choice $t = \langle\rangle$. In the step case, a case distinction on the first event v in λ is made. The four cases are (a) $v \in V_1 \cap V_2 \cap \nabla_1$, (b) $v \in V_1 \cap V_2 \cap \nabla_2$, (c) $v \in V_1 \setminus E_2$, and (d) $v \in V_2 \setminus E_1$. Due to the symmetry of the conditions, the proofs of (b) and (d) are similar to the proofs of (a) and (c), respectively.

In the proof of case (a), t_1 is split into subsequences before and after v, resulting in $r_1.v.s_1$ ($r_1.v.s_1 = t_1$, $r_1|_{V_1} =$

$\langle\rangle$). Then $r_1|_{E_2}$ is inserted into $\tau|_{E_2}.t_2$ using $BSI_{\mathcal{V}_2}(Tr_2)$, resulting in $\tau|_{E_2}.t_2'$. t_2' is split into subsequences before and after v, resulting in $r_2'.v.s_2'$ ($r_2'.v.s_2' = t_2'$, $r_2'|_{V_2} = \langle\rangle$). The sequence $r_2'|_{E_1}$ is inserted by an inductive argument from left to right into $\tau|_{E_1}.r_1.v.s_1$ directly before v using $FCI^{\nabla_1,\Delta_1,\Upsilon_1}_{\mathcal{V}_1}(Tr_1)$, resulting in $\tau|_{E_1}.r_1.q_1'.v.s_1'$. The sequence $q_1' \in ((C_1 \cap \Upsilon_1) \cup (N_1 \cap \Delta_1))^*$ need not be identical to $r_2'|_{E_1}$ but it may differ in events from $N_1 \cap \Delta_1$. The proof of case (a) is concluded by an application of the induction hypothesis.

The proof of case (c) is identical to the proof of case (a) until the step where $\tau|_{E_2}.t_2'$ is constructed. After this step, the induction hypothesis can be applied directly.

Proof. [Sketch of Theorem 3] We prove the first of the six propositions in the Theorem as an example. The proofs of the other propositions proceed along similar lines and, in particular also make use of the generalized zipping lemma.

Proposition 1: Let $\tau' \in Tr$ be arbitrary. We have $\tau'|_{E_1} \in Tr_1$ and $\tau'|_{E_2} \in Tr_2$. According to $R_{\mathcal{V}_1}(Tr_1)$ and $R_{\mathcal{V}_2}(Tr_2)$, there are $\tau_1' \in Tr_1$ and $\tau_2' \in Tr_2$ with $\tau_1'|_V = \tau'|_{E_1 \cap V}$, $\tau_1'|_{C_1} = \langle\rangle$, $\tau_2'|_V = \tau'|_{E_2 \cap V}$, $\tau_2'|_{C_2} = \langle\rangle$. Lemma 1 yields for $\tau = \langle\rangle$, $\lambda = \tau'|_V$, $t_1 = \tau_1'$, and $t_2 = \tau_2'$ that there is a sequence $t \in E^*$ with $t \in Tr$, $t|_V = \tau'|_V$, and $t|_C = \langle\rangle$. Thus, $R_{\mathcal{V}}(Tr)$ holds.

Proof. [Sketch of Theorem 4] Assume that, for $i \in \{1,2\}$, $BSD_{\mathcal{V}^{LHI}_{L_i}}(Tr_i)$, $BSI_{\mathcal{V}^{LHI}_{L_i}}(Tr_i)$, and $FCI^{\nabla_i,\Delta_i,\Upsilon_i}_{\mathcal{V}^{LHI}_{L_i}}(Tr_i)$ hold for $\mathcal{V}^{LHI}_{L_i} = (L_i, H_i \backslash HI_i, HI_i)$, $\Upsilon_i = I_i$, $\nabla_i = I_i$, and $\Delta_i = E_i \setminus (I_i \cup O_i)$. Consequently, the fourth condition of the generalized zipping lemma is satisfied, i.e. $V_1 \cap V_2 \subseteq \nabla_1 \cup \nabla_2$, $C_1 \cap N_2 \subseteq \Upsilon_1$, $C_2 \cap N_1 \subseteq \Upsilon_2$, $\Delta_1 \cap E_2 = \emptyset$, and $\Delta_2 \cap E_1 = \emptyset$ hold. Moreover, the conditions of the second, third, and sixth case of Theorem 3 are satisfied. Note that $\Upsilon \cap E_i \subseteq \Upsilon_i$, $\nabla \cap E_i \subseteq \nabla_i$, and $\Delta \supseteq ((\Delta_1 \cap N_1) \cup (\Delta_2 \cap N_2))$ hold for $i \in \{1,2\}$, $\Upsilon = I$, $\nabla = I$, and $\Delta = E \setminus (I \cup O)$. Moreover, $\Delta_1 \cap E_2 = \emptyset$ and $\Delta_2 \cap E_1 = \emptyset$ hold. Therefore, Theorem 3 implies that $BSD_{\mathcal{V}^{LHI}_L}(Tr)$, $BSI_{\mathcal{V}^{LHI}_L}(Tr)$, and $FCI^{\nabla,\Delta,\Upsilon}_{\mathcal{V}^{LHI}_L}(Tr)$ are all satisfied.

Proof. [Sketch of Theorem 6] The proof proceeds by induction on the size of the index set J.

In the base case ($J = \{j\}$), $BSD_{\mathcal{V}_j}(Tr_j)$ and $BSI_{\mathcal{V}_j}(Tr_j)$ hold by assumption. Since $\bigcup_{k \in J, k \neq j} E_k = \emptyset$ holds, the remaining three propositions reduce to $FCI^{E_j,E_j,E_j}_{\mathcal{V}_j}(Tr_j)$, $FCI^{\emptyset,\emptyset,E_j}_{\mathcal{V}_j}(Tr_j)$, and $FCI^{E_j,\emptyset,\emptyset}_{\mathcal{V}_j}(Tr_j)$. These propositions follow from Proposition 3 of Theorem 1 and Proposition 3 of Theorem 5.

In the step case, $J = J' \cup \{i\}$, $J' \neq \emptyset$, and $i \notin J'$ holds. Let $ES' = (E', I', O', Tr')$ be defined by $ES' = \|_{j \in J'} ES_j$. Hence, it remains to compose ES' and ES_i. In order to apply Theorem 3, it is shown that all assumptions of Lemma 1 can be satisfied for ES' and ES_i. For this purpose, firstly, J' is partitioned into two disjoint subsets J_1', J_2' that are defined by $J_1' = \{j \in J' \mid N_j \cap E_i = \emptyset\}$ and $J_2' = J' \setminus J_1'$ ($N_i \cap E_j = \emptyset$ holds for every $j \in J_2'$). Secondly, a view $\mathcal{V}' = (V', N', C')$ in E' is defined by:

$$\begin{aligned} V' &= \textstyle\bigcup_{j \in J'} V_j, \\ N' &= \textstyle\bigcup_{j \in J'} N_j, \text{ and} \\ C' &= \{c \in E' \mid \forall j \in J'. (c \in E_j \Longrightarrow c \in C_j)\}. \end{aligned}$$

Thirdly, it is shown that $FCI^{\nabla',\Delta',\Upsilon'}_{\mathcal{V}'}(Tr)$ holds where the sets $\nabla', \Delta', \Upsilon' \subseteq E'$ are defined by:

$$\begin{aligned} \nabla' &= E' \setminus \{e \in E_k \cap E_l \mid k, l \in J' \wedge k \neq l\}, \\ \Delta' &= \textstyle\bigcup_{j \in J_1'} E_j, \text{ and} \\ \Upsilon' &= (\textstyle\bigcup_{j \in J_1'} E_j) \setminus \{e \in E_k \cap E_l \mid k, l \in J' \wedge k \neq l\}. \end{aligned}$$

Therefore, $BSI_{\mathcal{V}'}(Tr')$ (induction hypothesis), $BSI_{\mathcal{V}_i}(Tr_i)$, $FCI^{\nabla',\Delta',\Upsilon'}_{\mathcal{V}'}(Tr')$, and $FCI^{\nabla_i,\Delta_i,E_i}_{\mathcal{V}_i}(Tr_i)$ hold where $\nabla_i = \emptyset$, $\Delta_i = \emptyset$, and $\Upsilon_i = E_i$ (last statement follows from Proposition 3 of Theorem 5). All remaining assumptions of Condition 4 in Lemma 1 are fulfilled for ∇', Δ', Υ', ∇_i, Δ_i, and Υ_i. The fact that no event can belong to more than two components (follows from $I_j \cap O_j = \emptyset$ and $E_j \cap E_k \subseteq (I_j \cap O_k) \cup (I_k \cap O_j)$) is important when proving some of these assumptions.

$BSD_{\mathcal{V}}(Tr)$ and $BSI_{\mathcal{V}}(Tr)$ follow from Propositions 2 and 3 of Theorem 3, respectively. The propositions

$$FCI_{\mathcal{V}}^{E_j \backslash (\bigcup_{k \in J, k \neq j} E_k),\ E_j,\ E_j \backslash (\bigcup_{k \in J, k \neq j} E_k)}(Tr) \quad (3)$$

$$FCI_{\mathcal{V}}^{(\bigcup_{k \in J, k \neq j} E_k) \backslash E_j,\ \emptyset,\ E_j \backslash (\bigcup_{k \in J, k \neq j} E_k)}(Tr) \quad (4)$$

$$FCI_{\mathcal{V}}^{E_j \backslash (\bigcup_{k \in J, k \neq j} E_k),\ \emptyset,\ (\bigcup_{k \in J, k \neq j} E_k) \backslash E_j}(Tr) \quad (5)$$

follow from Proposition 6 of Theorem 3. To prove that all side conditions of Proposition 6 are, indeed, fulfilled, is somewhat tedious – but possible. For each of Formula 3, 4, and 5, a case distinction on $j = i$ or $j \in J'$ is made. The six resulting cases follow from the induction hypothesis and the following propositions: $FCI^{\emptyset,\emptyset,E'}_{\mathcal{V}'}(Tr')$, $FCI^{E',\emptyset,\emptyset}_{\mathcal{V}'}(Tr')$, $FCI^{E_i,E_i,E_i}_{\mathcal{V}_i}(Tr_i)$, $FCI^{\emptyset,\emptyset,E_i}_{\mathcal{V}_i}(Tr_i)$, and $FCI^{E_i,\emptyset,\emptyset}_{\mathcal{V}_i}(Tr_i)$. These propositions hold according to Theorems 1 and 5.