

Preserving Security Properties under Refinement*

Fabio Martinelli
IIT-CNR
fabio.martinelli@iit.cnr.it

Illaria Matteucci
IIT-CNR
ilaria.matteucci@iit.cnr.it

ABSTRACT

Communication is one of the cornerstone of our everyday life. Guaranteeing the security of a communication is a very important challenge. In this paper, we propose a formal top-down approach for assuring that security properties are preserved during the development of a complex and concurrent system, *i.e.*, within passage from specification to implementation of the components of the system. Indeed, we investigate on the set of requirements a *refinement function* has to satisfy for preserving a class of properties that can be formalized as specific instances of a general scheme, called *Generalized Non Deducibility on Composition* (GNDC). Hence, we show that it is possible to guarantee that the refinement of a considered system that is verified to be GNDC at a high level of abstraction, is GNDC also at a lower one without checking it again.

Categories and Subject Descriptors: D4.6 Security and Protection, F3 Logics and meanings of programs, I6.5 Model Development

General Terms: Security, Theory, Verification

Keywords: Action Refinement: Specification of Security Properties: Controlled General Schema for Security Properties.

1. OVERVIEW

The efficacy of integrating and composing different software agents making them interoperable one each other in a secure way becomes a meaningful challenge. Furthermore, from a software engineering viewpoint, it may be required to compare systems that belong to conceptually different abstraction levels in order to verify if they satisfy the same properties. This is because it could happen that some lack of information occurs or, some detail is revealed to malicious agents, for instance, during the development of software agents.

In order to avoid this, we need to point out how the transition from the specification (*high level* of abstraction) to the implementation (*low level* of abstraction) of a system, namely *refinement*, can be done in order to preserve some properties. Indeed, once the sets of

*Work partially supported by EU-funded project "CONNECT", EU-funded project "ANIKETOS" and by EU-funded network of excellence "NESSoS".

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...\$10.00.

actions at the different abstraction levels are defined, a technique for controlling the complexity of concurrent system specification is by means of refinement, *i.e.*, a complex system can be first described succinctly as a simple, abstract specification and then refined stepwise to the actual, complex implementation.

Moreover, it is quite often required that the system properties that have been already checked in some abstraction level need not to be validated again at a lower abstraction level.

Here we focus on the *action refinement* theory [22], typically used in formal methods for converting the specification of an (abstract) action into a (concrete) process. Our goal is to describe a refinement strategy for introducing a mechanism for transforming high-level primitives/action into lower level processes, *i.e.*, processes built using low level action, in such a way that some security properties are preserved within the transformation.

We consider the family of security properties that can be expressed by using the GNDC schema [6], say GNDC property, and we identify which is the set of features an action refinement function has to satisfy for preserving such properties through different levels of abstraction.

The GNDC schema is a uniform approach for defining security properties derived from the *Non deducibility on Composition* (NDC) properties [8, 4].

The GNDC scheme expresses uniformly many security properties as, *e.g.*, fault tolerance properties (*fail stop*, *fail silent*, *fail safe* and *fault tolerant* behaviour [14, 7]) or, also, many security properties of cryptographic protocols as, *e.g.*, *secrecy* (confidential information should be available only to the partners of the communication), *authentication* (capability of identifying the other partner engaged in a communication), *integrity* (assurance of no alteration of message content), etc. [5].

Here we extend the definition of GNDC by introducing a *controller program* as parameter of the definition. A controller program is a process that, by monitoring the behaviour of a possible malicious component according to a strategy defined by a *controller operator* \triangleright , guarantees that a considered system is secure [19].

Hence, at high level of abstraction we have the following formalization, given in terms of process algebra [9], for our family of properties:

$$P \in GNDC_q^{\alpha, Y} \text{ iff } \forall X \in \mathcal{E}_H : (P|_H(Y \triangleright X))/H \triangleleft \alpha(P) \quad (1)$$

where $H \subseteq Act$ is the set of high actions, \mathcal{E}_H is the set of processes that perform only high level actions, $\triangleleft \in \mathcal{E} \times \mathcal{E}$ is a relation between processes and $\alpha : \mathcal{E} \rightarrow \mathcal{E}$ is a function between processes. The $|_H$ is the synchronization operator stating that all actions in H are performed by the system if and only if both P and $(Y \triangleright X)$ perform them, and the $/H$ is the hiding operator that, as recalling by the name, hides all actions in H .

Informally, the $GNDC_q^{\alpha, Y}$ property requires that the behaviour of the process P , once it is composed with any possible malicious

agent $X \in \mathcal{E}_H$ controlled by a controller program Y , is *compliant* with the correct behaviour described by the function α . The notion of compliance depends on the \triangleleft relation we chose for comparing the behaviours of $(P|_H(Y \triangleright X))/H$ and $\alpha(P)$.

Starting from Statement (1), we point out a possible set of features that a refinement function r needs to have in order to assure that, if $P \in \text{GNDC}_\triangleleft^{\alpha, Y}$ then the refined system obtained by applying r , $r(P)$, is in $\text{GNDC}_\triangleleft^{\alpha, r(Y)}$, regardless of the behaviour of the implementation of the possible malicious component. An advantage of this result is that, once we have proved that a *GNDC* property holds at the high specification level, we are able to implement it in such a way that the property holds also at lower level of abstraction, regardless of the implementation of the untrusted component and we guarantee this result without proving it again. The only requirement is that the level of abstraction of the unknown component (or the refinement procedure of the unknown component) is the same of the rest of the system.

This paper is organized as follows: Next section recalls the state of the art about refinement theory applied to security. Section 3 presents background notions about process algebra and action refinement theory. Section 4 shows the *GNDC* scheme with our extension of a controller operator. Section 5 proposes our approach to the application of the refinement theory for specifying and characterizing *GNDC* properties at different levels of abstraction. Section 6 concludes our work and proposes some ideas for future work.

2. STATE OF THE ART

Several research areas have been deal with the notion of process refinement. For that reason there are several definitions of refinement. [27] proposes the notion of *Stepwise refinement* as the top-down presentation of a software system's functionality. It consists in a sequence of layers of increasing detail, from abstract to concrete, in which each layer is an incremental "refinement" of the previous one. Making the separation between layers small means that each refinement step can be kept under conceptual control, in many cases even verified correct. Even though actually developing systems in this way remains a theoretical ideal (sometime achieved), the refinement provides a framework for encouraging correct, accountable and even efficient code. In [11], the authors have considered the notion of stepwise refinements as a useful paradigm for system development. Indeed, the paper is focused on the study of relevant properties, in particular security properties, that have to be preserved from abstract to concrete specification level. The idea is that a process specification preserves the secrecy of a piece of data if the process never sends out any information from which the data itself could be derived, even in interaction with an adversary. In general, it is slightly coarser than the first kind in that it may not prevent implicit information flow, but both kinds of security properties seem to be roughly equivalent in practice. Also, more fine-grained security properties may be hard to ensure in practice. For that reason they have proposed a secrecy-preserving refinement, one can also address situation where implementations of formally verified security protocols turn out be insecure. [22] presents the notion of action refinement as a function that converts a specification of an action on a system into an implementable program (e.g., a procedure). The basic idea of the process refinement is that, given two processes P and $R(P)$, $R(P)$ is the implementation of P if it is described at a deeper level of details than P . It is possible to refine policies by using the process algebra Communicating Sequential Processes (CSP), e.g., [9] to model security policies at different levels of abstraction and check their refinement and correctness [23]. Similarly, [2] presents the notion of refinement of CSP processes as the passage to a "less deterministic" processes, i.e., a process P_1

is refined by a process P_2 if every possible behaviour of P_2 is a possible behaviour of P_1 . Security issues are not addressed.

The notion of refinement can be also seen as the passage from a coarser level of specification to a finer one. [25] presents a case study of Event-B applied for the refinement of a controller for a security property along the different network layer of the TCP/IP stack. They also prove that the refinement is valid. Formal verification can be performed on the resulting model. One of the main advantages of this approach is the use of formal methods. Indeed policies are formally defined and the refinements can be checked. It is very suitable for critical systems and those which need traceability link between models. However, the approach is difficult to use in a large problem. [18] proposes an idea of a possible application of action refinement theory, for enforcing safety policies at different levels of abstraction by using process algebra controller operators. It focuses on the enforcement of safety properties. [1] uses action refinement for, in particular, the analysis of information flow properties. It gives a definition of the refinement function on process algebra in terms of simulation relation among processes. This notion of action refinement is different from the one we have adopted here. Furthermore, we deal with a family of security properties in which the information flow can be included. [15] proposes a collection of refinement operators able to preserve information flow properties. The notion of refinement considered in this paper corresponds to the removal of non-determinism. This is not the notion of refinement we consider. Furthermore, [15] focuses the attention on a single definition of information flow (perfect security property). In [10], the author refers to MAKS proposed by Mantel in [16] for specifying and verifying possibilistic information flow policies. In particular, the author considers some security predicates of MAKS and shows how they can be specified in term of configuration structures and which are the conditions under which these predicates are preserved under action refinement. Hence, the authors points out the pre-conditions of the structure for preserving properties under refinement. On the contrary, our goal is to point out the features of the refinement function for preserving a certain class of security properties whatever the system we want to develop is.

[24] proposes a formal framework for refining non-deterministic probabilistic processes that capture sufficient conditions to preserve probabilistic, entropy-based information flow properties. Also in this work, as in [10], the authors investigate the system conditions necessary for preserving security properties. Furthermore the considered properties and the definition of refinement are different with respect to the one we use in the following.

Regarding refinement applied to information flow, [26] proposes an architectural approach. Hence the notion of refinement here defined is slightly different from the one we have exploited in our work. As in [12], here the refinement is architectural, i.e., each component of the system can be viewed as being composed of subcomponents. Hence also the specification of the flow of information between different components of the system has to be specified in such a way that also the flow of information between subcomponents is permitted. From this point of view, policy refinement is the process that decomposes the high level policy relevant to a composite system into a set of policies that are executed in its constituent parts to implement the behaviour intended by the high level policy. Policy refinement is thus capable of mapping high level goals of a composite system automatically down to low level policies. A policy language satisfying refinement is a language that can be refined to some low level details. Also in [13] the authors refer to the notion of refinement as decomposition. They say that, for large systems, the implementation is not expected to be extracted directly from the initial specification. Each intermediate specification may be viewed as a refinement step that simply consists of a refinement of

one component into a combination of a number of subcomponents that are not yet implementable.

3. PROCESS ALGEBRA AND ACTION REFINEMENT

Process algebra [9] formalisms are used for formally specifying the behaviour of *processes* that autonomously and concurrently can proceed in their computation but they have also the possibility to communicate and synchronize among themselves. The actions they perform represent computation steps.

Once we have specified a process, we need to define how it could be *refined*. Indeed, we need a formal way for assuring that the implementation of a system matches with the specification of the same systems.

3.1 A Process Algebra

Let Act be a set of action names, ranged over by a, b, c, \dots and an invisible action τ that models the internal, non observable action. Furthermore, let \surd be a termination predicate and \mathcal{E} be a set of processes ranged over by P, Q, E, F, \dots

The syntax of the considered process algebra is the following:

$$P ::= \mathbf{0} \mid a.P \mid P; P \mid P + P \mid P|_A P \mid P[f] \mid P/A$$

where $A \subseteq Act$ and the relabelling function $f : Act \mapsto Act$ has to satisfy $f(\tau) = \tau$.

The informal meaning of these operators is the following one: $\mathbf{0}$ is the term that does nothing; a (closed) term $a.P$ represents a process that performs an action a and then behaves as P . The term $P; P$ describes the sequences of two components: At the beginning it executes all actions of the first component then starts to execute the second one. The term $P + P$ represents the non-deterministic choice: Choosing the action of one of the two components means dropping the other; the term $P|_A P$ is the synchronous parallel operator on the set of action A . Any action in A is performed when both the components of the term perform it. On the other hand, all action not in A are performed whenever one of the two components performs it. The process P/A is the hiding operator and behaves like P but the actions in A are replaced by τ ; the process $P[f]$ behaves like P , but its actions are renamed through relabelling function f .

The operational semantics of the presented process algebra is described by a labelled transition system LTS $(\mathcal{E}, Act, \rightarrow)$, where \mathcal{E} is the set of all terms and $\rightarrow \subseteq \mathcal{E} \times Act \times \mathcal{E}$ is a transition relation defined by structural induction as the least relation generated by the set of the following structural operational semantics rules.

$$\frac{P \surd \quad Q \surd \quad P \surd \quad Q \surd \quad P \surd \quad Q \surd \quad P \surd \quad P \surd}{\mathbf{0} \surd \quad (P+Q) \surd \quad (P;Q) \surd \quad (P|_A Q) \surd \quad (P/A) \surd \quad (P[f]) \surd}$$

$$\frac{a.P \xrightarrow{a} P \quad P; Q \xrightarrow{a} P; Q \quad P; Q \xrightarrow{a} Q'}{P \xrightarrow{a} P' \quad P \surd \quad Q \xrightarrow{a} Q'}$$

$$\frac{P \xrightarrow{a} P' \quad Q \xrightarrow{a} Q'}{P+Q \xrightarrow{a} P' \quad P+Q \xrightarrow{a} Q'}$$

$$\frac{P \xrightarrow{a} P' a \notin A \quad Q \xrightarrow{a} Q' a \notin A}{P|_A Q \xrightarrow{a} P'|_A Q \quad P|_A Q \xrightarrow{a} P|_A Q'}$$

$$\frac{P \xrightarrow{a} P' \quad Q \xrightarrow{a} Q' a \in A}{P|_A Q \xrightarrow{a} P'|_A Q'}$$

$$\frac{P \xrightarrow{a} P' a \notin A \quad P \xrightarrow{a} P' a \in A}{P/A \xrightarrow{a} P'/A \quad P/A \xrightarrow{\tau} P'/A}$$

The transition relation \rightarrow defines the usual concept of derivation in one step. As a matter of fact $P \xrightarrow{a} P'$ means that process P evolves in one step into process P' by executing action

$a \in Act$. The transitive and reflexive closure of $\bigcup_{a \in Act} \xrightarrow{a}$ is written \rightarrow^* . In particular, we use the notation $P \xrightarrow{\tau}^* P'$ ($P \xrightarrow{\epsilon} P'$ or $P \xrightarrow{\tau} P'$) in order to denote that P and P' belongs to the reflexive and transitive closure of $\xrightarrow{\tau}$. Also, $P \xrightarrow{a}^* P'$ if $P \xrightarrow{\tau}^* P_\tau \xrightarrow{a} P'_\tau \xrightarrow{\tau}^* P'$ where P_τ and P'_τ denote intermediate states¹. Moreover, the basic one-step transitions are extended to τ -abstracting multi-steps transitions in the usual way:

$$P \xrightarrow{\sigma} P' \Leftrightarrow P \xrightarrow{a_1 \dots a_n} P' \Leftrightarrow P \xrightarrow{\tau}^* \xrightarrow{a_1} \xrightarrow{\tau}^* \dots \xrightarrow{\tau}^* \xrightarrow{a_n} \xrightarrow{\tau}^* P'$$

where Act^* is the set of sequences of actions and $\sigma \in Act^*$, namely a *trace*.

Given a process P , $Der(P) = \{P' \mid P \rightarrow^* P'\}$ is the set of its derivatives. A process P is said *finite state* if $Der(P)$ is finite. $Sort(P)$ is the set of names of actions that syntactically appear in the process P .

The behaviour of two processes can be compared by using some *behavioural relation* as, for instance, *trace inclusion* and *weak simulation* [21]. We define *trace pre-order* (\leq) as follows.

DEFINITION 3.1. For any $P \in \mathcal{E}$ the set $T(P)$ of traces associated with P is $T(P) = \{\sigma \in Act^* \mid \exists P' : P \xrightarrow{\sigma} P'\}$. Q can execute all traces of P (notation $P \leq Q$) if and only if $T(P) \subseteq T(Q)$.

We also recall the notion of *weak simulation* as it is given in [21].

DEFINITION 3.2. Let $(\mathcal{E}, Act, \rightarrow)$ be an LTS of concurrent processes over the set of actions Act , and let \mathcal{R} be a binary relation over \mathcal{E} . Then \mathcal{R} is called *weak simulation*, denoted by \preceq , over $(\mathcal{E}, Act, \rightarrow)$ if and only if, whenever $(P, Q) \in \mathcal{R}$ we have: if $P \xrightarrow{a} P'$ then $\exists Q'$ such that $Q \xrightarrow{a} Q'$ and $(P', Q') \in \mathcal{R}$

3.2 Action Refinement

Let Act_A be the set of *abstract* actions and Act_C be the set of *concrete* actions. Moreover, let \mathcal{E}_{Act_A} and \mathcal{E}_{Act_C} be the set of high and low level processes, whose actions are in Act_A and Act_C , respectively. We recall the following formal definition.

DEFINITION 3.3 ([22]). A refinement function r maps abstract actions to concrete processes, where the notion of abstract and concrete are accompanied by a change of alphabet, the implementation of a specification S is given by the syntactic substitution of a concrete processes $r(a)$ for each occurrences of the action a in S .

For our purpose, in order to avoid unnecessary complications, we single out the fragment of refinement terms, \mathbf{R} , that can be used as the refinement of abstract actions as follows:

$$P ::= \mathbf{0} \mid a.P \mid P + P$$

According to [22], the terms in \mathbf{R} has to satisfy some criteria. Indeed, if a process P is a refinement of an atomic action, then it is required that P is: (i) non-empty, *i.e.*, a visible abstract action cannot simply disappear during refinement and (ii) eventually terminating, *i.e.*, the refinement of a given action cannot “get stuck” during execution.

A particular instance of an action refinement function is the *syntactic action refinement* defined as follows.

DEFINITION 3.4 ([22]). Let $r : Act_A \rightarrow \mathbf{R}$ be a refinement function. A syntactic action refinement is formalized as a partial

¹We can use the short notation $P \xrightarrow{\tau}^* \xrightarrow{a} \xrightarrow{\tau}^* P'$ when the intermediate states are not relevant.

function

$$r^* : \mathcal{E}_{Act_A} \rightarrow \mathcal{E}_{Act_C}$$

defined according to the set of rules in Figure 1.

$r^*(0)$	$:= 0$
$r^*(a.P)$	$:= r(a); r(P)$
$r^*(P + Q)$	$:= r^*(P) + r^*(Q)$
$r^*(P; Q)$	$:= r^*(P); r^*(Q)$
$r^*(P _A Q)$	$:= r^*(P) _{\bar{r}(A)} r^*(Q)$ if r is distinct on A
$r^*(P[f])$	$:= r^*(P)[f]$
	if $f \upharpoonright [H_A \cup H_C]$ is the identity function
$r^*(P/A)$	$:= r^*(P)/\bar{r}(A)$ if r preserves A

Figure 1: Syntactic Refinement.

4. A CONTROLLED GENERAL SCHEMA FOR SECURITY PROPERTIES

In this section we extend the definition of the *general schema* for the definition of security properties, *Generalized NDC* (*GNDC* for short) [6] by considering a controlled version of this schema. Originally, the *GNDC* schema was developed for finding a unique way for describing a set of security properties. A system P is said to be *GNDC*, i.e., it satisfies the security properties that can be expressed through this schema, if and only if, for every possible malicious agent, whose behaviour is a priori unknown, that cooperate with P , the combined system is *GNDC* too. This means that every possible attacks performed by every possible malicious agents do not violate the security of the system P . More formally, note that the *GNDC* schema is defined in a parametric way. The two parameters in the definition are α , that denotes the properties as a function of the system P , and \triangleleft , that denotes the considered “observational” behavioural relation. Let $H \subseteq Act$ be the set of high level actions and \mathcal{E}_H be the set of processes that perform only high level actions. A system P is $GNDC_{\triangleleft}^{\alpha}$ if and only if, for every possible *malicious component* $X \in \mathcal{E}_H$, the composition of the system P with the process X on actions in H satisfies a specification $\alpha(P)$. This means that $GNDC_{\triangleleft}^{\alpha}$ guarantees that the property α is satisfied by P with respect to \triangleleft relation regardless the behaviour of the malicious component X .

DEFINITION 4.1 (STANDARD GNDC). *Let $P \in \mathcal{E}$ be a process. We say that*

$$P \in GNDC_{\triangleleft}^{\alpha} \text{ iff } \forall X \in \mathcal{E}_H : (P|_H X)/H \triangleleft \alpha(P)$$

where $H \subseteq Act$ is the set of high actions, \mathcal{E}_H is the set of processes that perform only high level actions, $\triangleleft \in \mathcal{E} \times \mathcal{E}$ is a relation between processes and $\alpha : \mathcal{E} \rightarrow \mathcal{E}$ is a function between processes².

Let us consider a *controller program*, denoted by $Y \in \mathcal{E}_H$, embedded into the system we are going to analyse. In particular a controller program is a process that, according to the semantics definition of the controller operator, denoted by \triangleright , monitors the behaviour of all possible malicious components which behaviour (unknown a priori) aims to violate the security of the system. Hereafter, we consider a controller operator \triangleright [19] semantically de-

²This is not the definition of GNDC given in [6]. Here we use the CSP parallel composition combined with the hiding operator. It is conceptually equivalent to the previous one.

defined as follows:

$$\frac{Y \xrightarrow{a} Y' \quad X \xrightarrow{a} X'}{Y \triangleright X \xrightarrow{a} Y' \triangleright X'}$$

This operator works by monitoring a possible malicious component and terminating any execution of it that is about to violate the security policy we are considering. Its semantics rule states that if Y and X perform the same action a then such action is allowed, so the controlled process $Y \triangleright X$ performs a , otherwise it halts. It is important to note that this operator is similar to the parallel operator $|_A$ when A is the whole set of actions Act .

Hence, we formally define a *controlled* version of $GNDC_{\triangleleft}^{\alpha, Y}$ parametrized also with respect to a controller process Y as follows.

DEFINITION 4.2. *Controlled GNDC Let $P \in \mathcal{E}$ be a process. We say that*

$$P \in GNDC_{\triangleleft}^{\alpha, Y} \text{ iff } \forall X \in \mathcal{E}_H : (P|_H (Y \triangleright X))/H \triangleleft \alpha(P)$$

where $Y \in \mathcal{E}_H$ is a controller program, $H \subseteq Act$ is the set of high actions, \mathcal{E}_H is the set of processes that perform only high level actions, $\triangleleft \in \mathcal{E} \times \mathcal{E}$ is a relation between processes and $\alpha : \mathcal{E} \rightarrow \mathcal{E}$ is a function between processes.

Informally, the system P satisfies $GNDC_{\triangleleft}^{\alpha, Y}$ if and only if P , composed in parallel with any process $X \in \mathcal{E}_H$ controlled by a control program $Y \in \mathcal{E}_H$, shows (w.r.t. the process relation \triangleleft) the same behaviour as $\alpha(P)$.

Many security properties can be defined as instances of the *GNDC* schema:

- (i) *NDC* and *BNDC* properties (“no high level activity can change the low level observational behaviour”). The *NDC* is an instance of *GNDC* defined in terms of a trace equivalence, where $Y = \mathbf{0}$ and $\alpha(P)$ is P/H^3 . The version of *NDC* that uses weak bisimulation, instead of trace equivalence, is called Bisimulation-based *NDC* (*BNDC*).
- (ii) According to [7, 14] also fault tolerance properties, as, e.g., *fail stop*, *fail silent*, *fail safe* and *fault tolerant* behaviour can be seen an instance of the *GNDC* property given in terms of both trace equivalence and weak simulation relation.
- (iii) Referring to [5], also many security properties of cryptographic protocols as, e.g., *secrecy*, *authentication*, *integrity* and so on, can be formalized as different instances of the unique *GNDC* schema given in terms of different behavioural equivalences.

4.1 Static Verification of the Controlled GNDC

In order to deal with the universal quantification on all possible behaviour of the unknown component $X \in \mathcal{E}_H$, it is possible to provide a sufficient criterion for a static characterization, i.e., a characterization that does not involve the universal quantification on all possible malicious components, of $GNDC_{\triangleleft}^{\alpha, Y}$ properties by the introduction of the notion of *most powerful malicious agent*, denoted by $Top \in \mathcal{E}_H$.

We recall that a pre-order \triangleleft is said to be a *pre-congruence*

- (i) w.r.t. the operator $|_A$ where $A \subseteq Act$, if for every $P, Q, R \in \mathcal{E}$, if $Q \triangleleft R$ then $P|_A Q \triangleleft P|_A R$.
- (ii) w.r.t. the operator $/A$ where $A \subseteq Act$, if for every $P, Q \in \mathcal{E}$, if $P \triangleleft Q$ then $P/A \triangleleft Q/A$.

The following result holds.

PROPOSITION 4.1. *If \triangleleft is a pre-congruence w.r.t. parallel and hiding operators and if there exists a process $Top \in \mathcal{E}_H$ such that*

³In literature, e.g., [3], we can find the definition of *NDC* and *BNDC* in which α is the CCS restriction operator [20]. These are conceptually equivalent.

for every process $X \in \mathcal{E}_H$, $X \triangleleft Top$, then:

$$P \in GNDC_{\triangleleft}^{\alpha, Y} \text{ iff } (P|_H(Y \triangleright Top))/H \triangleleft \alpha(P)$$

Proof: (\Leftarrow) By the hypothesis that \triangleleft is a pre-congruence w.r.t. parallel operator, being \triangleright semantically equivalent to $|_A$ when $A = Act$ then, \triangleleft is a pre-congruence also w.r.t. \triangleright . Since it is a pre-congruence also w.r.t. hiding operators, we have that if $X \triangleleft X'$ then $(Y \triangleright X) \triangleleft (Y \triangleright X')$, hence, $(P|_H(Y \triangleright X)) \triangleleft (P|_H(Y \triangleright X'))$ and, consequently, $(P|_H(Y \triangleright X))/H \triangleleft (P|_H(Y \triangleright X'))/H$.

Thus, if there exists a process Top s.t. for every process $X \in \mathcal{E}_H$, $X \triangleleft Top$, we have that $(Y \triangleright X) \triangleleft (Y \triangleright Top)$, hence, $(P|_H(Y \triangleright X)) \triangleleft (P|_H(Y \triangleright Top))$ and, consequently, $(P|_H(Y \triangleright X))/H \triangleleft (P|_H(Y \triangleright Top))/H$.

Since for hypothesis, $(P|_H(Y \triangleright Top))/H \triangleleft \alpha(P)$ then we obtain that, for every process $X \in \mathcal{E}_H$ $(P|_H(Y \triangleright X))/H \triangleleft (P|_H(Y \triangleright Top))/H \triangleleft \alpha(P)$ that means that $P \in GNDC_{\triangleleft}^{\alpha, Y}$.

(\Rightarrow) This implication holds since $Top \in \mathcal{E}_H$. \square

This allows to directly check that $\alpha(P)$ is satisfied when P is composed with the most powerful malicious agent Top controlled by a controller program Y . In order to better explain the meaning of Proposition 4.1, we give the following example.

EXAMPLE 4.1. Let $P = l.0 + h.l_1.0$ where $h \in H$ and $l, l_1 \in Act \setminus H$. According with the behavioural relation we chose, the process Top is different. Let us consider the trace inclusion relation \succeq , defined as \leq^{-1} , as pre-congruence. In this case, $Top = 0$. We prove that $P \notin GNDC_{\succeq}^{\alpha, Y}$ whatever Y is. Indeed,

$$(P|_H(Y \triangleright Top))/H = (l.0 + h.l_1.0|_H(Y \triangleright 0))/H$$

According with the semantics definition of \triangleright , the process $Y \triangleright 0$ behaves as 0, regardless the behaviour of Y . Hence, according to the semantics definition of $|_H$ and $/H$, $(l.0 + h.l_1.0|_H(Y \triangleright 0))/H$ behaves as $l.0$.

On the other hand, P/H behaves as $l.0 + \tau.l_1.0$.

Since $l.0 \not\succeq l.0 + \tau.l_1.0$ we can conclude that $P \notin GNDC_{\succeq}^{\alpha, Y}$. \diamond

5. REFINEMENT OF CONTROLLED GNDC PROPERTIES

In this section we point out the set of features of a refinement function and the general conditions under which security properties expressed by the controlled GNDC schema are preserved under refinement, i.e., under which assumptions, if a controlled GNDC properties holds at the specification level it holds also at a lower (implementation) level.

The main advantage of the provided result is that, once we have proved that a system is secure at specification level and its implementation follows a certain strategy, here referred to as *refinement function*, we are able to assure that a system described at a low, possible implementation, level is secure without check it again. Indeed, the refinement function represents an implementation strategy according to with the system preserves its security properties from its specification to its implementation.

Let us consider an action refinement function $r : Act_A \rightarrow Act_C$. For each action $a \in Act_a$, let $\tilde{r}(a)$ be the alphabet of a refinement term $r(a)$, i.e., $\tilde{r}(a) = Sort(r(a))$. For our purpose we require that r has the following features:

(1) the refinement function r preserves a certain set $A \subseteq Act_A$. This means that there is no overlap between the actions occurring in the refinement of (the elements of) A and of (the elements of) $Act_A \setminus A$.

(2) the refinement function r is *distinct* on Act . This means that (i) for all distinct $a \in A$ and $b \in Act_A$, $\tilde{r}(a) \cap \tilde{r}(b) = \emptyset$ and (ii) for all $a \in A$ and all sub processes $P+Q$ of $r(a)$, $Sort(P) \cap Sort(Q) =$

\emptyset . In other words, r is *distinct* on A if also the refinement of different actions in A have disjoint alphabets, and the images of individual actions in A contain no more than a single instance of any action. This implies that a distinct refinement function is also deterministic.

(3) the refinement function r is surjective on H . This means that, $\tilde{r}(H_A) = H_C$ where H_C is the set of all high concrete actions.

Under these assumptions on the refinement function, we aim to provide which are the necessary conditions on the function α and the relation \triangleleft to guarantee the following result holds:

$$P \in GNDC_{\triangleleft}^{\alpha, Y} \Rightarrow r(P) \in GNDC_{\triangleleft}^{\alpha, r(Y)} \quad (2)$$

Referring to the function α , we make the following assumption.

ASSUMPTION 5.1. Let $\alpha : \mathcal{E} \rightarrow \mathcal{E}$ be a function between process algebra terms without distinction on the level of abstraction. The following holds:

$$r(\alpha(P)) \triangleleft \alpha(r(P))$$

Referring to \triangleleft , we assume that it is a *pre-congruence* with respect to both parallel and hiding operators. Furthermore we make the following assumption.

ASSUMPTION 5.2.

$$\begin{aligned} \forall X \in \mathcal{E}_{H_A}^A \quad (P|_{H_A}(Y \triangleright X))/H_A \triangleleft \alpha(P) \\ \Downarrow \\ \forall r(X) \in \mathcal{E}_{H_C}^C \quad (r(P)|_{H_C}(r(Y) \triangleright r(X)))/H_C \triangleleft r(\alpha(P)) \end{aligned}$$

Hence we have the following result.

THEOREM 5.1. Let \triangleleft be a pre-congruence with respect to both parallel and hiding operators. If Assumption 5.1 and Assumption 5.2 hold and if the chosen controller operator is defined in such a way that

$$\forall X \in \mathcal{E}_{H_C}^C \quad (r(Y) \triangleright X) \triangleleft r(Y) \quad (3)$$

then Statement 2 holds, i.e.,

$$P \in GNDC_{\triangleleft}^{\alpha, Y} \Rightarrow r(P) \in GNDC_{\triangleleft}^{\alpha, r(Y)}$$

Proof: By the hypothesis that the relation in Statement 3 holds, then:

$$\forall X \in \mathcal{E}_{H_C}^C \quad (r(Y) \triangleright X) \triangleleft r(Y) \triangleleft (r(Y) \triangleright r(Y))$$

By the hypothesis that \triangleleft is a pre-congruence with respect to both parallel and hiding operators, for all $X \in \mathcal{E}_{H_C}^C$, the following chain holds:

$$\begin{aligned} (r(P)|_{H_C}(r(Y) \triangleright X))/H_C &\triangleleft (r(P)|_{H_C}r(Y))/H_C \\ &\triangleleft (r(P)|_{H_C}(r(Y) \triangleright r(Y)))/H_C \end{aligned}$$

Referring to Assumption 5.2, if the implication holds for all $r(X) \in \mathcal{E}_{H_C}^C$, in particular, it holds for $r(Y) \in \mathcal{E}_{H_C}^C$ hence we obtain the following relation:

$$(r(P)|_{H_C}(r(Y) \triangleright r(Y)))/H_C \triangleleft r(\alpha(P))$$

According to Assumption 5.1,

$$(r(P)|_{H_C}(r(Y) \triangleright r(Y)))/H_C \triangleleft r(\alpha(P)) \triangleleft \alpha(r(P))$$

Hence, starting from the fact that $P \in GNDC_{\triangleleft}^{\alpha, Y}$, we have that

$$\forall X \in \mathcal{E}_{H_C}^C \quad (r(P)|_{H_C}(r(Y) \triangleright X))/H_C \triangleleft \alpha(r(P))$$

that means that $r(P) \in GNDC_{\triangleleft}^{\alpha, r(Y)}$. \square

In order to deal with the universal quantification on possible behaviour of a malicious component also at concrete level, according to Proposition 4.1 and Theorem 2, if there exists a pro-

cess $Top_C \in \mathcal{E}_{H_C}^C$, that is the most powerful malicious agent expressed at the concrete level, we obtain a static characterization of $GNDC_{\triangleleft}^{\alpha, r(Y)}$ properties as follows:

$$r(P) \in GNDC_{\triangleleft}^{\alpha, r(Y)} \text{ if and only if } (r(P)|_H(r(Y) \triangleright Top_C))/H_C \triangleleft \alpha(r(P))$$

In the following, we show two possible instances of the $GNDC_{\triangleleft}^{\alpha, Y}$ properties by showing that trace inclusion and weak simulation relation respect Assumption 5.2 and the hiding operator $/H$ respects Assumption 5.1. Finally we present two corollaries to Theorem 5.1 about the two instances of $GNDC_{\triangleleft}^{\alpha, Y}$.

5.1 Refinement of $GNDC_{\triangleleft}^{/H, Y}$ Properties

In this section we consider an instance of controlled $GNDC$ properties given in terms of trace inclusion and hiding operator. We show that this family of controlled $GNDC$ properties is preserved by a refinement function that satisfies the features listed above. In particular, whenever \triangleleft is the trace inclusion and $\alpha(P) = P/H_A$ then we have the following result.

COROLLARY 5.1. *Let $P \in \mathcal{E}^A$ be a process described at specification level. Let $Y \in \mathcal{E}_{H_A}^A$ be a controller program described at specification level.*

If $P \in GNDC_{\triangleleft}^{/H_A, Y}$ then $r(P) \in GNDC_{\triangleleft}^{/H_C, r(Y)}$.

According to Definition 4.2, in order to prove Theorem 5.1 we have to show that, if $P \in GNDC_{\triangleleft}^{/H_A, Y}$ the following statement holds:

$$\forall X \in \mathcal{E}_{H_C}^C (r(P)|_{H_C}(r(Y) \triangleright X))/H_C \leq r(P)/H_C \quad (4)$$

First of all, it is possible to note that, according to the rules in Figure 1, since r preserves and is distinct on Act , and consequently on H_A , and $\tilde{r}(H_A) = H_C$ then we have that:

$$r(P/H_A) = r(P)/\tilde{r}(H_A) = r(P)/H_C$$

Hence, the hiding operator satisfies the Assumption 5.1.

Moreover,

$$\begin{aligned} r((P|_{H_A}(Y \triangleright X))/H_A) &= (r(P)|_{\tilde{r}(H_A)}(r(Y) \triangleright r(X)))/\tilde{r}(H_A) \\ &= (r(P)|_{(H_C)}(r(Y) \triangleright r(X)))/(H_C) \end{aligned}$$

According to the semantics definition of parallel operator, being $Sort((r(Y) \triangleright r(X))) \subseteq H_C$ for all $r(X) \in \mathcal{E}_{H_C}^C$ then

$$(r(P)|_{H_C}(r(Y) \triangleright r(X))) \leq r(P)$$

Recalling that trace inclusion is a pre-congruence w.r.t. hiding operators, the following relation holds:

$$\forall r(X) \in \mathcal{E}_{H_C}^C (r(P)|_{H_C}(r(Y) \triangleright r(X)))/H_C \leq r(P)/H_C \quad (5)$$

This implies that the trace inclusion satisfies Assumption 5.2. In order to obtain the result in Statement (4) from the one in (5), we note that, for the semantics definition of the controller operator \triangleright , the following relation holds:

$$\forall X \in \mathcal{E}_{H_C}^C r(Y) \triangleright X \leq r(Y)$$

Since trace equivalence is a pre-congruence w.r.t. both parallel and hiding operators, for all $X \in \mathcal{E}_{H_C}^C$, the following chain holds:

$$\begin{aligned} (r(P)|_{H_C}(r(Y) \triangleright X))/H_C &\leq (r(P)|_{H_C}r(Y))/H_C \\ &\leq (r(P)|_{H_C}(r(Y) \triangleright r(Y)))/H_C \end{aligned}$$

Referring to Statement (4), if the implication holds for all $r(X) \in \mathcal{E}_{H_C}^C$, in particular, it holds for $r(Y) \in \mathcal{E}_{H_C}^C$ hence we obtain the following relation:

$$(r(P)|_{H_C}(r(Y) \triangleright r(Y)))/H_C \leq r(P/H_A) = r(P)/H_C$$

Hence, starting from the fact that $P \in GNDC_{\triangleleft}^{\alpha, Y}$, we have that

$$\forall X \in \mathcal{E}_{H_C}^C (r(P)|_{H_C}(r(Y) \triangleright X))/H_C \leq r(P)/H_C$$

that means that $r(P) \in GNDC_{\triangleleft}^{/H_C, r(Y)}$.

5.2 Refinement of $GNDC_{\triangleleft}^{/H, Y}$ Properties

Similarly to what we have done in the previous section, here we instantiate controlled $GNDC$ properties in terms of weak simulation and hiding operator. We show that this family of controlled $GNDC$ properties is preserved by a refinement function that satisfies the features listed above. In particular, whenever \triangleleft is the weak simulation and $\alpha(P) = P/H_A$ then we have the following result.

COROLLARY 5.2. *Let $P \in \mathcal{E}^A$ be a process described at specification level. Let $Y \in \mathcal{E}_{H_A}^A$ be a controller program described at specification level.*

If $P \in GNDC_{\triangleleft}^{/H_A, Y}$ then $r(P) \in GNDC_{\triangleleft}^{/H_C, r(Y)}$.

According to Definition 4.2, in order to prove Theorem 5.1 we have to show that, if $P \in GNDC_{\triangleleft}^{/H_A, Y}$ the following statement holds:

$$\forall X \in \mathcal{E}_{H_C}^C (r(P)|_{H_C}(r(Y) \triangleright X))/H_C \preceq r(P)/H_C \quad (6)$$

First of all, it is possible to note that, according to the rules in Figure 1, since r preserves and is distinct on Act , and consequently on H_A , and $\tilde{r}(H_A) = H_C$ then we have that:

$$r(P/H_A) = r(P)/\tilde{r}(H_A) = r(P)/H_C$$

Hence, the hiding operator satisfies the Assumption 5.1.

Moreover,

$$\begin{aligned} r((P|_{H_A}(Y \triangleright X))/H_A) &= (r(P)|_{\tilde{r}(H_A)}(r(Y) \triangleright r(X)))/\tilde{r}(H_A) = \\ &= (r(P)|_{(H_C)}(r(Y) \triangleright r(X)))/(H_C) \end{aligned}$$

According to the semantics definition of parallel operator, being $Sort((r(Y) \triangleright r(X))) \subseteq H_C$ for all $r(X) \in \mathcal{E}_{H_C}^C$ then $(r(P)|_{H_C}(r(Y) \triangleright r(X))) \preceq r(P)$. Recalling that weak simulation is a pre-congruence w.r.t. hiding operators, the following relation holds:

$$\forall r(X) \in \mathcal{E}_{H_C}^C (r(P)|_{H_C}(r(Y) \triangleright r(X)))/H_C \preceq r(P)/H_C \quad (7)$$

This implies that the weak simulation satisfies Assumption 5.2.

In order to obtain the result in Statement (6) from the one in (7), we note that, for the semantics definition of the controller operator \triangleright , the following relation holds:

$$\forall X \in \mathcal{E}_{H_C}^C r(Y) \triangleright X \preceq r(Y)$$

Since weak simulation is a pre-congruence w.r.t. both parallel and hiding operators, for all $X \in \mathcal{E}_{H_C}^C$, the following chain holds:

$$\begin{aligned} (r(P)|_{H_C}(r(Y) \triangleright X))/H_C &\preceq (r(P)|_{H_C}r(Y))/H_C \\ &\preceq (r(P)|_{H_C}(r(Y) \triangleright r(Y)))/H_C \end{aligned}$$

Referring to Statement (6), if the implication holds for all $r(X) \in \mathcal{E}_{H_C}^C$, in particular, it holds for $r(Y) \in \mathcal{E}_{H_C}^C$ hence we obtain the following relation:

$$(r(P)|_{H_C}(r(Y) \triangleright r(Y)))/H_C \preceq r(P/H_A) = r(P)/H_C$$

Hence, starting from the fact that $P \in GNDC_{\triangleleft}^{\alpha, Y}$, we have that

$$\forall X \in \mathcal{E}_{H_C}^C (r(P)|_{H_C}(r(Y) \triangleright X))/H_C \preceq r(P)/H_C$$

that means that $r(P) \in GNDC_{\triangleleft}^{/H_C, r(Y)}$.

6. CONCLUSION AND FUTURE WORK

This work investigates on the set of features refinement procedures and general assumptions necessary for preserving *GNDC* properties through different levels of abstractions. Indeed, by applying the theory on action refinement to the definition of *GNDC* properties we have pointed out which are the conditions under which we are able to infer that the validity of a *GNDC* property at a high level of abstraction is preserved through the passage to a lower (possible implementable) level of abstraction. This work consists in a formal definition of a refinement procedure for preserving security properties and it does not investigate on the feasibility of the described approach. As future work, we plan to investigate on this direction in order to be able to implement refinement functions that satisfy the features pointed out in this paper. We have focused our attention, in particular, on the analysis of two kinds of behavioural relation for comparing processes. We aim to extend this work by analysing and studying other behavioural relations and equivalences. We also aim to apply the action refinement theory to cryptographic protocols specified by using Crypto-CCS [17] and *GNDC*. We also would like to go further in this direction by dealing also with the synthesis of controller program Y as in [19].

7. REFERENCES

- [1] A. Bossi, C. Piazza, and S. Rossi. Action refinement in process algebra and security issues. In *LOPSTR*, pages 201–217, 2007.
- [2] S. Cattani and M. Kwiatkowska. A refinement-based process algebra for timed automata. *Formal Aspects of Computing*, 17(2):138–159, 2005.
- [3] R. Focardi and R. Gorrieri. A taxonomy of security properties for ccs. In *CSFW'94*, pages 126–136. IEEE Computer Society, 1994.
- [4] R. Focardi and R. Gorrieri. Classification of security properties (part I: Information flow). In *FOSAD '00*, pages 331–396. Springer-Verlag, 2001.
- [5] R. Focardi, R. Gorrieri, and F. Martinelli. Classification of security properties - part ii: Network security. In *FOSAD*, volume 2946 of *Lecture Notes in Computer Science*, pages 139–185, 2002.
- [6] R. Focardi and F. Martinelli. A uniform approach for the definition of security properties. In *FM '99*, pages 794–813, London, UK, 1999. Springer-Verlag.
- [7] S. Gnesi, G. Lenzini, and F. Martinelli. Applying generalized non deducibility on compositions (*GNDC*) approach in dependability. *Electr. Notes Theor. Comput. Sci.*, 99:111–126, 2004.
- [8] J. A. Goguen and J. Meseguer. Security policy and security models. In *Proc. of the 1982 Symposium on Security and Privacy*, pages 11–20. IEEE Press, 1982.
- [9] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, Englewood Cliffs, NJ, 1985.
- [10] D. Hutter. Possibilistic information flow control in majs and action refinement. In *ETRICS*, pages 268–281, 2006.
- [11] J. Jurjens. Secrecy-preserving refinement. In *FME '01*, pages 135–152, London, UK, 2001. Springer-Verlag.
- [12] V. W. Kevin Carey, David Lewis. Automated policy-refinement for managing composite services. In *M-Zones White Paper June 04*, whitepaper 06/04, 2004.
- [13] K. G. Larsen and L. Xinxin. Compositionality through an operational semantics of contexts. *Journal of Logic and Computation*, 1(6):761–795, Dec. 1991.
- [14] G. Lenzini, F. Martinelli, I. Matteucci, and S. Gnesi. A uniform approach to security and fault-tolerance specification and analysis. In *WADS*, pages 172–201, 2008.
- [15] H. Mantel. Preserving information flow properties under refinement. In *SP '01: Proceedings of the 2001 IEEE Symposium on Security and Privacy*, Washington, DC, USA, 2001. IEEE Computer Society.
- [16] H. Mantel. *A Uniform Framework for the Formal Specification and Verification of Information Flow Security*. PhD thesis, Universität des Saarlands, 2003.
- [17] F. Martinelli. Analysis of security protocols as open systems. *Theor. Comput. Sci.*, 290(1):1057–1106, 2003.
- [18] F. Martinelli and I. Matteucci. Idea: Action refinement for security properties enforcement. In *Engineering Secure Software and Systems, First International Symposium ESSoS 2009*, pages 37–42, 2009.
- [19] F. Martinelli and I. Matteucci. A framework for automatic generation of security controller. *STVR Journal*, 2010.
- [20] R. Milner. Operational and algebraic semantics of concurrent processes. In J. van Leewen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, chapter 19, pages 1201–1242. The MIT Press, New York, N.Y., 1990.
- [21] R. Milner. *Communicating and mobile systems: the π -calculus*. Cambridge University Press, 1999.
- [22] A. Rensink and R. Gorrieri. Vertical implementation. *Inf. Comput.*, 170(1):95–133, 2001.
- [23] A. W. Roscoe. On the expressive power of csp refinement. *Form. Asp. Comput.*, 17(2):93–112, 2005.
- [24] T. Santen. Preservation of probabilistic information flow under refinement. *Inf. Comput.*, 206(2-4):213–249, 2008.
- [25] N. Stouls and M. L. Potet. Security policy enforcement through refinement process. In *The 7th International B Conference*, volume 4355 of *Lecture Notes in Computer Science*, 2007.
- [26] R. van der Meyden. Architectural refinement and notions of intransitive noninterference. In *ESSoS'09*, pages 60–74.
- [27] N. Wirth. Program development by stepwise refinement. *Commun. ACM*, 14(4):221–227, 1971.