Foundations and Applications of Security Analysis

York, UK, March 28-29, 2009
Revised Selected Papers
Preface


ARSPA is a series of workshops on “Automated Reasoning for Security Protocol Analysis,” bringing together researchers and practitioners from both the security and the formal methods communities, from academia and industry, who are working on developing and applying automated reasoning techniques and tools for the formal specification and analysis of security protocols. The first two ARSPA workshops were held as satellite events of the Second International Joint Conference on Automated Reasoning (IJCAR 2004) and of the 32nd International Colloquium on Automata, Languages and Programming (ICALP 2005), respectively. ARSPA then joined forces with the workshop FCS (Foundations of Computer Security): FCS-ARSPA 2006 was affiliated with LICS 2006, in the context of FLoC 2006, and FCS-ARSPA 2007 was affiliated with LICS 2007 and ICALP 2007.

WITS is the official annual workshop organized by the IFIP WG 1.7 on “Theoretical Foundations of Security Analysis and Design,” established to promote the investigation on the theoretical foundations of security, discovering and promoting new areas of application of theoretical techniques in computer security and supporting the systematic use of formal techniques in the development of security-related applications. This is the ninth meeting in the series. In 2008, ARSPA and WITS joined with the workshop on Foundations of Computer Security FCS for a joint workshop, FCS-ARSPA-WITS 2008, associated with LICS 2008 and CSF 21.

In 2009, ARSPA and WITS again joined forces with the aim to provide a forum for continued activity in different areas of computer security, bringing computer security researchers in closer contact with the ETAPS community and giving ETAPS attendees an opportunity to talk to experts in computer security, on the one hand, and to contribute to bridging the gap between logical methods and computer security foundations, on the other.

There were 27 submissions of high quality, from countries in Asia, Europe, and North America. All the submissions were evaluated by at least three referees and the Program Committee then selected the 15 research contributions that were presented at the workshop. Out of these, 12 were further revised by their authors and are included in this volume. The workshop program was enriched by invited talks by Peter Ryan and David Sands, whose contributions are also included here.

We would like to thank all the people who contributed to the organization of the ARSPA-WITS 2009 Workshop, and acknowledge the support from the IFIP WG 1.7, the AVANTSSAR Project (FP7-ICT-2007-1, Project No. 216471), and
the SENSORIA Project (EU-FETPI Global Computing Project IST-2005-16004). In particular, we are deeply indebted to the other members of the Program Committee and the additional referees, who allowed us to review the papers in a very short time, while maintaining a very high standard: their help has been invaluable. We are also grateful to Andrei Voronkov, who allowed us to use the free conference software system EasyChair, which greatly simplified the work of the Program Committee. Last but not least, warm thanks to the organizers of ETAPS 2009.

June 2009

Pierpaolo Degano
Luca Viganò
Program Committee

Lujo Bauer CMU, USA
Luca Compagna SAP Research, France
Veronique Cortier LORIA INRIA-Lorraine, France
Pierpaolo Degano (Chair) Università di Pisa, Italy
Sandro Etalle Technical University of Eindhoven, The Netherlands
Riccardo Focardi Università di Venezia, Italy
Dieter Gollman Technische Universität Hamburg-Harburg, Germany
Roberto Gorrieri Università di Bologna, Italy
Joshua Guttman MITRE, USA
Jerry den Hartog Technical University of Eindhoven, The Netherlands
Jan Jürgens The Open University, UK
Gavin Lowe Oxford University, UK
Catherine Meadows Naval Research Laboratory, USA
Jonathan Millen MITRE, USA
Sebastian Mödersheim IBM Zurich Research Lab, Switzerland
Mark Ryan University of Birmingham, UK
Luca Viganò (Chair) Università di Verona, Italy

Additional Reviewers

Misha Aizatulin Francois Dupressoir Ben Smyth
Alessandro Aldini Deepak Garg Bruno Pontes
Andreas Bauer Claudio Guidi Soares Rocha
Giampaolo Bella Volkmar Lotz Fred Spiessens
Mario Bravetti Ilaria Matteucci Angelo Troina
Roberto Carbone Toby Murray Mathieu Turuani
Kostas Chatzikokolakis Arnab Roy Tjark Weber
Stéphanie Delaune Theoodor Scholte
Alessandra Di Pierro Boris Skoric
# Table of Contents

A Policy Model for Secure Information Flow

*Adedayo O. Adetoye and Atta Badii*

A General Framework for Nondeterministic, Probabilistic, and Stochastic Noninterference

*Alessandro Aldini and Marco Bernardo*

Validating Security Protocols under the General Attacker

*Wihem Arsac, Giampaolo Bella, Xavier Chantry, and Luca Compagna*

Usage Automata

*Massimo Bartoletti*

Static Detection of Logic Flaws in Service-Oriented Applications

*Chiara Bodei, Linda Brodo, and Roberto Bruni*

Improving the Semantics of Imperfect Security

*Niklas Broberg and David Sands*

Analysing PKCS#11 Key Management APIs with Unbounded Fresh Data

*Sibylle Fröschle and Graham Steel*

Transformations between Cryptographic Protocols

*Joshua D. Guttman*

Formal Validation of OFEPPSP+ with AVISPA

*Jorge L. Hernandez-Ardieta, Ana I. Gonzalez-Tablas, and Benjamin Ramos*

On the Automated Correction of Protocols with Improper Message Encoding

*Dieter Hutter and Raúl Monroy*

Finite Models in FOL-Based Crypto-Protocol Verification

*Jan Jürjens and Tjark Weber*

Towards a Type System for Security APIs

*Gavin Keighren, David Aspinall, and Graham Steel*

Separating Trace Mapping and Reactive Simulatability Soundness: The Case of Adaptive Corruption

*Laurent Mazaré and Bogdan Warinschi*
Table of Contents

How Many Election Officials Does It Take to Change an Election? .... 211
P.Y.A. Ryan

Author Index .......................................................... 223
A General Framework for Nondeterministic, Probabilistic, and Stochastic Noninterference

Alessandro Aldini and Marco Bernardo
University of Urbino “Carlo Bo” – Italy
Information Science and Technology Institute

Abstract. We introduce a notion of stochastic noninterference aimed at extending the classical approach to information flow analysis with fine-grain information describing the temporal behavior of systems. In particular, we refer to a process algebraic setting that joins durational activities expressing time passing through exponentially distributed random variables, zero duration activities allowing for prioritized/probabilistic choices, and untimed activities with unspecified duration. In this setting unifying time, priority, probability, and nondeterminism, we highlight the expressive power of stochastic noninterference with respect to the existing definitions of nondeterministic and probabilistic noninterference. From this comparison, we obtain that stochastic noninterference turns out to be very strict and limiting in real-world applications and, therefore, requires the use of relaxation techniques. Among them we advocate performance evaluation as a means for achieving a reasonable balance between security requirements and quality.

1 Fine-Grain Models and Noninterference

Information flow analysis is a basic approach to the verification of security properties of systems which, in general, require to control who has access to what and when. Among the several conditions that describe the characteristics of unauthorized information flows, called covert channels, one of the most interesting, for its intuitive and wide-used idea, is the noninterference requirement [11]. Very briefly, in a multi-level secure system, the user at the high security level should not be able to affect what the user at the low security level can observe. Independently of the specific formalization of this notion, the underlying approach is based on the idea that checking noninterference is actually checking the indistinguishability of the different low-level views of the system that are obtained by changing the high-level behavior, see e.g. [9,17,22] and the references therein.

Along the 90s generalized notions of noninterference were designed to analyze deterministic and nondeterministic systems. However, it was immediately clear that moving to a quantitative framework including fine-grain information, such as probability distributions associated with event execution, augments the distinguishing power of the observer [3,14,18,19]. More recently, the awareness that perfect noninterference is difficult to achieve has urged to estimate the information flow by employing this quantitative information and a relaxed notion of
noninterference, see e.g. [4] and the references therein. A quantitative notion of noninterference can be based not only on probabilistic information, but also on temporal information. In particular, in this paper we refer to the representation of time passing that uses non-negative random variables, which is particularly appropriate when the time taken by an event fluctuates according to some probability distribution. Among the many probability distributions that can be used to model event timing, we concentrate on exponential distributions, which are equipped with a widely developed theory.

As known in the probabilistic setting, the more information is added to the system, the higher the number of potential vulnerabilities revealed through fine-grain notions of noninterference. Hence, a noninterference property taking into account the timing of events, described through exponentially distributed random variables, would be sensitive to stochastically timed covert channels. In the literature, some proposals concerned with discrete-time extensions of noninterference have been proposed, e.g. in the setting of real-time process algebra [8] and probabilistic timed automata [7, 12], while other time-based aspects of non-interference have been investigated in the general setting of Shannon information theory, see e.g. [13]. However, to the best of our knowledge, no formal definition of noninterference has been proposed in the case of stochastic time.

This paper faces the problem of defining noninterference in the setting of a stochastic process algebra encompassing nondeterminism, probability, priority, and stochastic time. The semantics of this paradigm is defined in terms of a fine-grain notion of bisimulation that extends the classical bisimulation relation of Milner [15]. In this general framework, it is possible to formalize not only stochastic noninterference properties, but also to evaluate the expressive power that is gained when adding fine-grain information in an incremental way. The result we obtain is that when moving from nondeterministic noninterference to fine-grain definitions of noninterference, security constraints become severe and hard to accomplish in practice. In the most restrictive scenario where temporal behavior and stochastic noninterference come into play, we will see that these constraints may be impractical, so that in real-world applications we need relaxation techniques aiming at trading the amount of (unavoidable) information leakage with the complexity of the adopted securing countermeasures.

We will discuss the expressive power of the different notions of noninterference through a client-server example with two access clearance levels, high and low. The clients compete for the resource on the basis of their access clearance (assume that high-level clients take precedence over low-level clients). The service center is composed of two buffers of capacity 1 – one for each security level – and one shared server providing a service to the clients of both levels. This toy example is more than enough to highlight the severeness of the constraints to which a completely secure system should be subjected.

The paper is organized as follows. In Sect. 2 we describe the stochastic process algebra framework, on the base of which in Sect. 3 we introduce the stochastic version of noninterference together with a comparison with the nondeterministic/probabilistic setting. Some conclusions are drawn in Sect. 4.
2 Stochastic Process Algebra Framework

The general framework for the analysis of fine-grain noninterference we use is based on a Markovian process calculus extended with prioritized/weighted immediate actions, in which multiway communication is based on a mixture of the generative and reactive models of \[10\]. All these features, which are equipped with a widely developed theory, provide the expressiveness needed to model fine-grain details of real systems.

In the following, we explain the syntax and semantics of this calculus, how to derive nondeterministic and probabilistic sub-calculi, and the bisimulation semantics.

2.1 Markovian Process Calculus

The basic elements of the calculus are durational actions. They are represented as pairs of the form \(<a, \lambda>\), where \(a\) is the action name and \(\lambda\) is the action rate. There are three kinds of actions: exponentially timed, immediate, and passive.

Exponentially timed actions are of the form \(<a, \lambda>\) with \(\lambda \in \mathbb{R}_{>0}\). The duration of each such action is exponentially distributed with parameter equal to the action rate (hence its average duration is the inverse of its rate). Whenever several exponentially timed actions are enabled, the race policy is adopted, hence the action that is executed is the fastest one. As a consequence, the execution probability of an exponentially timed action is proportional to its rate.

Immediate actions are of the form \(<a, \infty_{l,w}>\), where \(l \in \mathbb{N}_{>0}\) is the priority level and \(w \in \mathbb{R}_{>0}\) is the weight. Each immediate action has duration zero and takes precedence over exponentially timed actions, which are assumed to have priority level 0. Whenever several immediate actions are enabled, the generative preselection policy is adopted. This means that the lower priority immediate actions are discarded, whereas each of the highest priority immediate actions is given an execution probability equal to the action weight divided by the sum of the weights of all the highest priority immediate actions.

Passive actions are of the form \(<a, \ast_{l,w}'>\), where \(l' \in \mathbb{N}\) is the priority constraint and \(w \in \mathbb{R}_{>0}\) is the weight. Each passive action with priority constraint \(l'\) can synchronize only with another passive action having the same name and the same priority constraint, or with a non-passive action having the same name and priority level \(l\) such that \(l = l'\). Whenever several passive actions are enabled, the reactive preselection policy is adopted. This means that, within every set of enabled passive actions having the same name, each such action is given an execution probability equal to the action weight divided by the sum of the weights of all the actions in the set. Instead, the choice among passive actions having different names is nondeterministic. Likewise, the choice between a passive action and a non-passive action is nondeterministic.

**Definition 1.** Let \(\text{Act} = \text{Name} \times \text{Rate}\) be a set of actions, with \(\text{Name}\) being a set of action names containing a distinguished symbol \(\tau\) for the invisible action and \(\text{Rate} = \mathbb{R}_{>0} \cup \{\infty_{l,w} | l \in \mathbb{N}_{>0} \land w \in \mathbb{R}_{>0}\} \cup \{\ast_{l,w}' | l' \in \mathbb{N} \land w \in \mathbb{R}_{>0}\}\) being
a set of action rates (ranged over by \( \tilde{\lambda} \)). The set \( \mathcal{L} \) of process terms is generated by the following syntax:

\[
P ::= \emptyset \mid \langle a, \tilde{\lambda} \rangle . P \mid P + P \mid A \mid P/L \mid P \parallel S P
\]

where \( L, S \subseteq \text{Name} - \{ \tau \} \) and \( A \) is a process constant defined through the (possibly recursive) equation \( A \triangleq P \).

The semantics for the set \( \mathcal{P} \) of closed and guarded process terms of \( \mathcal{L} \) is defined in the usual operational style. The behavior of each term \( P \) is given by a labeled multitransition system \([ [ P ] ]\), whose states correspond to process terms and whose transitions – each of which has a multiplicity equal to the number of proofs of its derivation – are labeled with actions.

The null term \( \emptyset \) cannot execute any action, hence the corresponding semantics is given by a state with no transitions. The action prefix term \( \langle a, \tilde{\lambda} \rangle . P \) can execute an action with name \( a \) and rate \( \tilde{\lambda} \) and then behaves as \( P \):

\[
\langle a, \lambda \rangle . P \xrightarrow{a, \lambda} P \quad \langle a, \infty \rangle . P \xrightarrow{a, \infty} P \quad \langle a, * \rangle . P \xrightarrow{a, *} P
\]

The alternative composition \( P_1 + P_2 \) behaves as either \( P_1 \) or \( P_2 \) depending on whether \( P_1 \) or \( P_2 \) executes an action first:

\[
P_1 \xrightarrow{a, \tilde{\lambda}} P' \quad P_2 \xrightarrow{a, \tilde{\lambda}} P'
\]

\[
P_1 + P_2 \xrightarrow{a, \tilde{\lambda}} P'
\]

The process constant \( A \) behaves as the right-hand side process term in its defining equation:

\[
P \xrightarrow{a, \tilde{\lambda}} P' \quad A \triangleq P
\]

\[
A \xrightarrow{a, \tilde{\lambda}} P'
\]

The hiding term \( P/L \) behaves as \( P \) with the difference that the name of every action executed by \( P \) that belongs to \( L \) is turned into \( \tau \):

\[
P \xrightarrow{a, \tilde{\lambda}} P' \quad a \in L
\]

\[
P/L \xrightarrow{\tau, \tilde{\lambda}} P'/L
\]

\[
P \xrightarrow{a, \tilde{\lambda}} P' \quad a \notin L
\]

\[
P/L \xrightarrow{a, \tilde{\lambda}} P'/L
\]

The parallel composition \( P_1 \parallel S P_2 \) behaves as \( P_1 \) in parallel with \( P_2 \) as long as actions are executed whose name does not belong to \( S \):

\[
P_1 \parallel S P_2 \xrightarrow{a, \tilde{\lambda}} P'_1 \parallel S P'_2
\]

\[
P_1 \parallel S P_2 \xrightarrow{a, \tilde{\lambda}} P'_1 \parallel S P'_2
\]

Generative-reactive synchronizations are forced between any non-passive action executed by one term and any passive action executed by the other term that have the same name belonging to \( S \) and the same priority level/constraint:
\[
\begin{align*}
  &P_1 \xrightarrow{a,\lambda} P_1' \quad P_2 \xrightarrow{a,\star w} P_2' \quad a \in S \\
  &P_1 \parallel P_2 \xrightarrow{a,\lambda} P_1' \parallel P_2' \quad a \in S \\
  &P_1 \parallel P_2 \xrightarrow{a,\star w} P_1' \parallel P_2' \quad a \in S \\
  &P_1 \parallel P_2 \xrightarrow{a,\lambda} P_1' \parallel P_2' \quad a \in S
\end{align*}
\]

where \( \text{weight}(P,a,l) = \sum \{ w \mid \exists P' \in \mathcal{P}, P \xrightarrow{a,\star w} P' \} \). Following \([6]\), the adopted synchronization discipline mixes generative and reactive probabilistic models. Among all the enabled timed actions whose names belong to a synchronization set, the proposal of an action name is generated after a selection based on the rates of those actions. Then the enabled passive actions that have the same name as the proposed one react by means of a selection based on their weights. Finally, the timed action winning the generative selection and the passive action winning the reactive selection synchronize with each other.

Reactive-reactive synchronizations are forced between any two passive actions of the two terms that have the same name belonging to \( S \) and the same priority constraint:

\[
\begin{align*}
  &P_1 \xrightarrow{a,\star w_1} P_1' \quad P_2 \xrightarrow{a,\star w_2} P_2' \quad a \in S \\
  &P_1 \parallel P_2 \xrightarrow{a,\star w_1} P_1' \parallel P_2' \quad \text{weight}(P_1,a,l) + \text{weight}(P_2,a,l)
\end{align*}
\]

We will use the abbreviation \( P \parallel S \) to stand for \( P \parallel S \{ 0 \} \), which intuitively describes the behavior of a restriction operator. Moreover, if \( P \xrightarrow{a_1,\lambda_1} \cdots \xrightarrow{a_n,\lambda_n} P' \), with \( n \in \mathbb{N} \), then we say that \( P' \) is a derivative of \( P \) and we denote with \( \text{Der}(P) \) the set of derivatives of \( P \). We denote with \( \mathcal{P}_{pc} \) the set of performance closed process terms of \( P \), i.e. those terms with no passive transitions. The stochastic process underlying \( P \in \mathcal{P}_{pc} \) is a continuous-time Markov chain (CTMC) possibly extended with immediate transitions. States having outgoing immediate transitions are called vanishing as the sojourn time in these states is zero. In order to retrieve a pure CTMC stochastically equivalent to an extended CTMC, all the vanishing states can be adequately eliminated.

**Example 1.** Let us exemplify the use of the operators through the running example modeling the client-server system. The arrival process for clients of level \( k \in \{ L, H \} \), where \( L \) stands for low access level and \( H \) stands for high access level, can be modeled as:

\[
A_L \overset{\Delta}{=} <a_L,\lambda_L>.A_L \quad \text{and} \quad A_H \overset{\Delta}{=} <a_H,\lambda_H>.A_H
\]
where $\lambda_k$ is the parameter of the Poisson arrival process for clients of access level $k$. The buffer for clients of access level $k$ – which we assume to have capacity 1 – can be modeled as a completely passive entity as follows:

\[ B_k \triangleq <a_k, *_w^0>, <d_k, *_{f(k)}^w>.B_k \]

where, for instance, we assume $f(L) = 1$ and $f(H) = 2$ to denote that high-level requests pre-empt low-level ones. The shared server can be modeled as follows:

\[ S \triangleq \sum_{k \in \{L,H\}} <d_k, \infty_{f(k),w}>. <s_k, \mu_k>.S \]

where $\mu_k$ is the service rate for clients of access level $k$. Hence, the overall client-server system can be modeled as follows:

\[ CS \triangleq (A_L \parallel \emptyset A_H) \parallel \{a_L, a_H\} (B_L \parallel \emptyset B_H) \parallel \{d_L, d_H\} S \]

### 2.2 Nondeterministic and Probabilistic Sub-calculi

From the calculus above it is straightforward to derive two sub-calculi: a nondeterministic one and a probabilistic one. The former is obtained by removing all the fine-grain information related to time, probabilities, and priorities. Syntactically, durational actions of the form $<a, \lambda>$ and $<a, \infty_{l,w}>$ are replaced by the output action $a$, while passive actions of the form $<a, *_{l,w}>$ become the input action $a_*$. Semantically, the result is a classical calculus with CSP-like communication, whose underlying model is a labeled transition system. We denote with $nd(P)$ the nondeterministic version of process $P$ obtained in this way.

The latter sub-calculus is obtained by removing all the fine-grain information related to time and priorities. Syntactically, the exponentially timed action $<a, \lambda>$ is mapped to the immediate action $<a, \infty_{0,\lambda}>$. In this way, the action rate is interpreted as the action weight that governs the execution probability of the action. The immediate action $<a, \infty_{l,w}>$ is simply mapped to $<a, \infty_{0,w}>$, thus losing the information about its priority. Similarly, the passive action $<a, *_{l,w}>$ is turned into the action $<a, *_{0,w}>$. Semantically, the result is a calculus where the execution probabilities of the actions are calculated on the basis of the associated weights. The underlying model for this calculus is a mixture of the generative and reactive probabilistic transition systems. A variant of this calculus, where the probabilities are associated with the operators rather than attached to the actions as weights, has been introduced in [6] and used in [3] to define probabilistic noninterference. We denote with $pr(P)$ the probabilistic version of process $P$ obtained in this way.

### 2.3 Bisimulation Semantics

We now recall from [5] an extension of Markovian bisimilarity, whose basic idea is to compare the exit rates of the process terms by taking into account the various kinds of actions (exponentially timed, immediate, and passive). This is accomplished by parameterizing the notion of exit rate with respect to a number in $\mathbb{Z}$ representing the priority level of the action, which is 0 if the action is exponentially timed, $l$ if the action rate is $\infty_{l,w}$, $-l' - 1$ if the action rate is $*_{l,w}^{l'}$. 
Definition 2. Let \( P \in \mathcal{P} \), \( a \in \text{Name} \), \( l \in \mathbb{Z} \), and \( C \subseteq \mathcal{P} \). The exit rate of \( P \) when executing actions with name \( a \) and priority level \( l \) that lead to \( C \) is defined through the following non-negative real function:

\[
\text{rate}(P, a, l, C) = \begin{cases} 
\sum \{|\lambda| \exists P' \in C. P \xrightarrow{a, \lambda} P' \} & \text{if } l = 0 \\
\sum \{|w| \exists P' \in C. P \xrightarrow{a, \infty, l, w} P' \} & \text{if } l > 0 \\
\sum \{|w| \exists P' \in C. P \xrightarrow{a, \ast, l-1} P' \} & \text{if } l < 0
\end{cases}
\]

where each sum is taken to be zero whenever its multiset is empty.

Extended Markovian bisimilarity compares the process term exit rates for all possible action names and priority levels, except for those actions that will always be pre-empted by higher priority actions of the form \(<\tau, \infty_1, w>\). We denote by \( \text{pri}_{\infty}(P) \) the priority level of the highest priority internal immediate action enabled by \( P \), and we set \( \text{pri}_{\infty}(P) = 0 \) if \( P \) does not enable any internal immediate action. Moreover, given \( l \in \mathbb{Z} \), we use \( \text{no-pre}(l, P) \) to denote that no action of level \( l \) can be pre-empted in \( P \). Formally, this is the case whenever \( l \geq \text{pri}_{\infty}(P) \) or \(-l - 1 \geq \text{pri}_{\infty}(P)\).

Definition 3. An equivalence relation \( \mathcal{B} \subseteq \mathcal{P} \times \mathcal{P} \) is an extended Markovian bisimulation iff, whenever \((P_1, P_2) \in \mathcal{B}\), then for all action names \( a \in \text{Name} \), equivalence classes \( C \in \mathcal{P}/\mathcal{B} \), and priority levels \( l \in \mathbb{Z} \) such that \( \text{no-pre}(l, P_1) \) and \( \text{no-pre}(l, P_2) \):

\[
\text{rate}(P_1, a, l, C) = \text{rate}(P_2, a, l, C)
\]

Extended Markovian bisimilarity, denoted by \( \sim_{\text{EMB}} \), is the union of all the extended Markovian bisimulations.

We relax the definition of exit rate by means of a notion of reachability that involves the internal actions with zero duration \(<\tau, \infty_1, w>\), which are unobservable. The idea is that, if a given class of process terms is not reached directly after executing an action with a certain name and level, then we have to explore the possibility of reaching that class indirectly via a finite-length unobservable path starting from the term reached after executing the considered action.

Definition 4. Let \( P \in \mathcal{P} \) and \( l \in \mathbb{N}_{>0} \). We say that \( P \) is \( l \)-unobservable iff \( \text{pri}_{\infty}(P) = l \) and \( P \) does not enable any immediate non-\( \tau \)-action with priority level \( l' \geq l \), nor any passive action with priority constraint \( l' \geq l \).

Definition 5. Let \( n \in \mathbb{N}_{>0} \) and \( P_1, P_2, \ldots, P_{n+1} \in \mathcal{P} \). A path \( \pi \) of length \( n \):

\[
P_1 \xrightarrow{\tau, \infty_1, w_1} P_2 \xrightarrow{\tau, \infty_2, w_2} \cdots \xrightarrow{\tau, \infty_n, w_n} P_{n+1}
\]

is unobservable iff for all \( i = 1, \ldots, n \) process term \( P_i \) is \( l_i \)-unobservable. In that case, the probability of executing path \( \pi \) is given by:

\[
\text{prob}(\pi) = \prod_{i=1}^{n} \frac{w_i}{\text{rate}(P_i, \tau, l_i, P)}
\]

Definition 6. Let \( P \in \mathcal{P} \), \( a \in \text{Name} \), \( l \in \mathbb{Z} \), and \( C \subseteq \mathcal{P} \). The weak exit rate at which \( P \) executes actions of name \( a \) and level \( l \) that lead to \( C \) is defined through the following non-negative real function:
rate_w(P, a, l, C) = \sum_{P' \in C_w} rate(P, a, l, \{P'\}) \cdot prob_w(P', C)

where C_w is the weak backward closure of C:

\[
C_w = C \cup \{Q \in P - C \mid Q \text{ can reach } C \text{ via unobservable paths}\}
\]

and prob_w is a \(\mathbb{R}_{[0,1]}\)-valued function representing the sum of the probabilities of all the unobservable paths from a term in C_w to C:

\[
prob_w(P', C) = \begin{cases} 1 & \text{if } P' \in C \\ \sum \{\|prob(\pi)| \pi \text{ unobs. path from } P' \text{ to } C\} & \text{if } P' \in C_w - C \end{cases}
\]

When comparing process term weak exit rates, besides taking pre-emption into account, we also have to skip the comparison for classes that contain certain unobservable terms. More precisely, we distinguish among observable, initially unobservable, and fully unobservable terms. An observable term is a term that enables a non-\(\tau\)-action that cannot be pre-empted by any enabled immediate \(\tau\)-action. An initially unobservable term is a term in which all the enabled non-\(\tau\)-actions are pre-empted by some enabled immediate \(\tau\)-action, but at least one of the paths starting at this term with one of the higher priority enabled immediate \(\tau\)-actions reaches an observable term. A fully unobservable term is a term in which all the enabled non-\(\tau\)-actions are pre-empted by some enabled immediate \(\tau\)-action, and all the paths starting at this term with one of the higher priority enabled immediate \(\tau\)-actions are unobservable (note that \(\_\) is fully unobservable).

The weak exit rate comparison with respect to observable and fully unobservable classes must obviously be performed. In order to maximize the abstraction power in the presence of quantitative information attached to immediate \(\tau\)-actions, the comparison should be conducted with respect to the whole set \(P_{fu}\) of fully unobservable process terms of \(P\). By contrast, the comparison with respect to initially unobservable classes should be skipped as each of them can reach an observable class.

**Definition 7.** An equivalence relation \(B \subseteq P \times P\) is a weak extended Markovian bisimulation iff, whenever \((P_1, P_2) \in B\), then for all action names \(a \in Name\) and levels \(l \in \mathbb{Z}\) such that no-pre\((l, P_1)\) and no-pre\((l, P_2)\):

\[
rate_w(P_1, a, l, C) = rate_w(P_2, a, l, C) \quad \text{for all observable } C \in P/B
\]

\[
rate_w(P_1, a, l, P_{fu}) = rate_w(P_2, a, l, P_{fu})
\]

Weak extended Markovian bisimilarity, denoted by \(\approx_{EMB}\), is the union of all the weak extended Markovian bisimulations.

As examples of weakly extended Markovian bisimilar process terms we mention:

\[P_1 \equiv <a, \lambda>.<\tau, \infty_{l, w}>.<b, \mu>.\_\]
\[P_2 \equiv <a, \lambda>.<b, \mu>.\_\]

and also:

\[P_3 \equiv <a, \lambda>.(<\tau, \infty_{l, w_1}>.<b, \mu>.\_ + <\tau, \infty_{l, w_2}>.<c, \gamma>.\_)
\]
\[P_4 \equiv <a, \lambda.\frac{\mu}{w_1 + w_2}>.<b, \mu>.\_ + <a, \lambda.\frac{\mu}{w_1 + w_2}>.<c, \gamma>.\_
\]

which is related to vanishing state elimination. We also point out that \(\approx_{EMB}\) can abstract not only from intermediate immediate \(\tau\)-actions, but also from intermediate unobservable self-loops, consistently with the fact that the probability to escape from them is 1. Moreover, \(\approx_{EMB}\) cannot abstract from initial immediate
τ-actions, otherwise compositionality with respect to the alternative composition operator would be broken. The careful classification of states on the basis of their functional and performance observability is a key ingredient thanks to which congruence and axiomatization can be achieved for \( \approx_{\text{EMB}} \). In particular, compositionality with respect to parallel composition is preserved by restricting to a well-prioritized subset of non-divergent process terms of \( \mathcal{P} \). [5]

3 Noninterference Properties

In a typical two-level access system we distinguish between the high security access level and the low security access level. Information flowing from low level to high level is authorized, while the inverse flow is not. The noninterference analysis aims at revealing direct and indirect information flows, called covert channels, in the unauthorized path. In a process algebraic setting this classification of security levels is realized by assuming that the set \( \text{Name} \) of action names includes a set \( \text{Name}^L \subseteq \text{Name} \) of low-level names and a set \( \text{Name}^H \subseteq \text{Name} \) of high-level names, such that \( \text{Name}^L \cap \text{Name}^H = \emptyset \). These sets describe the interactions between the system and the low-level and high-level parts of the environment, respectively. We can also have that \( \text{Name}^L \cup \text{Name}^H = \text{Name} \). If this is not the case, we assume that the remaining action names can be disregarded as they are not related to interactions with the users at different access clearances. For instance, they could represent synchronizing activities performed by different system components that cooperate to handle the service requests. Hence, they will be hidden by turning them into unobservable actions.

In the following, we describe two noninterference properties relying on this classification and on the bisimulation semantics, named strong noninterference property and strong local noninterference property.

3.1 Bisimulation-Based Strong Noninterference

The requirement at the base of the lack of any interference from high level to low level can be easily expressed by a property called Strong Nondeterministic Noninterference, which informally says that a system is secure if its observable low-level behavior is the same in the presence and in the absence of high-level interactions. The stochastic version of this property is called Bisimulation-based Strong Stochastic Noninterference (BSSNI) and is defined as follows, where \( P/\text{Name}^H \) expresses the system view in the presence of high-level actions, which are hidden because they cannot be explicitly seen by a low-level observer, while \( P\setminus\text{Name}^H \) represents the system view whenever the high-level actions are absent, i.e. they are prevented from execution by means of the restriction operator.

\[ \text{Definition 8. (BSSNI)} \quad P \in \mathcal{P}_{pc} \text{ is secure iff } P/\text{Name}^H \approx_{\text{EMB}} P\setminus\text{Name}^H. \]

For brevity, we assume that \( P \) is performance closed. For the formal treatment of open systems with respect to high-level inputs, the interested reader is referred to [3]. The nondeterministic version of BSSNI, termed BSNNI [9], is obtained
in this framework by replacing $\approx_{\text{EMB}}$ with the Milner’s weak bisimilarity $\approx_{B}$,
denoted by $\approx_{B}$.

**Definition 9.** *(BSNNI)* $\text{nd}(P)$ is secure iff $\text{nd}(P)/\text{Name}_H \approx_B \text{nd}(P)\setminus\text{Name}_H$.

Similarly, the probabilistic version of BSSNI, termed BSPNI $\text{[3]}$, is defined in
the same framework by replacing $\approx_{\text{EMB}}$ with the weak probabilistic bisimilarity
of $\text{[6]}$, denoted by $\approx_{PB}$.

**Definition 10.** *(BSPNI)* $\text{pr}(P)$ is secure iff $\text{pr}(P)/\text{Name}_H \approx_{PB} \text{pr}(P)\setminus\text{Name}_H$.

**Example 2.** We now discuss an incremental analysis of the client-server system
introduced in Ex. $\text{[1]}$. This is done by considering the three fine-grain notions
of noninterference in order to show which kind of information flow can be re-
vealed and what strategies should be implemented to circumvent the related
covert channels. We start by classifying the activities performed by the client-server system into low level and high level. The low-level client is represented by process $A_L$, thus the action name modeling the arrival process, $a_L$, and the action name modeling the delivered service, $s_L$, are the unique action names in $Name_L$. Similarly, given that $A_H$ represents the high-level client, we assume that $Name_H = \{a_H, s_H\}$. The communications between the buffers and the server represent internal activities among components of the service center. Thus, they should not be observed by any client. Hence, the noninterference check is applied to system $CS/\{d_L, d_H\}$ even if, for brevity, we will continue to refer to $CS$.

First, consider the property $BSNNI$ and the system $nd(CS)$. The underlying semantic model is represented by the labeled transition system $[nd(CS)]$ in the upper part of Fig. 1 which also includes in its lower part the two labeled transition systems to compare according to the noninterference property. The weak bisimulation relating these two sub-systems is illustrated by graphically representing in the same way the states that belong to the same class. Hence, it is easy to see that the noninterference check is satisfied and the functional behavior of the system is secure. Intuitively, the availability to serve the low-level requests is never compromised independently of the behavior of the high-level client.

Second, consider the property $BSPNI$ and the system $pr(CS)$. The semantics of this system is obtained from $[nd(CS)]$ by enriching every edge in Fig. 1 with a weight. The intuition is that with this extension we add fine-grain information which allow nondeterministic choices to be solved according to probability distributions modeling the quantitative behavior of system activities. In spite of this enriched, detailed description, the noninterference check based on $BSPNI$ is satisfied and the system turns out to be secure. As a sketch of the proof of this result, consider the semantics of the sub-system $pr(CS)\setminus Name_H$.

From the unique state of class • it is possible to reach, through the action $a_L$, class □ with probability 1. Then, after an internal move within the same class, the unique probabilistic choice of this sub-system governs the choice among the action $s_L$ leading to class • and the action $a_L$ leading to class ○. Finally, from class ○ the unique enabled transition, which is labeled with $s_L$, leads to class □ with probability 1. Now, consider the semantics of the sub-system $pr(CS)\setminus Name_H$. From states of class • it is possible to pass, through the weak transition $\tau^*a_L$, to class □ with probability 1. With the same probability, we then reach, by following unobservable paths, one of two states of class □ enabling the same probabilistic choice between $s_L$ and $a_L$ that characterizes the former sub-system. Finally, the two states of class ○ lead with probability 1 to states of class □ through the weak transition $\tau^*s_L$. Summing up, adding probabilistic information to the system does not increase the discriminating power of the low-level observer, because any high-level activity is not able to alter the unique low-level probabilistic choice.

Third, consider the property $BSSNI$ and the system $CS$. The intuition is that with this extension the clients observe the passage of time and the priority-based behavior of the server. The introduction of this fine-grain information causes an interfering information flow from high level to low level, which is revealed by the
violation of the BSSNI condition. In the following, we show the nature of these covert channels and how the diagnostic information provided by \( \approx_{\text{EMB}} \) has been exploited to avoid them.

On the one hand, the high-level process \( A_H \) is immediately revealed by the low-level client. Indeed, \( A_H / \text{Name}_H \) describes a working process that, according to the race policy, competes with the other durational processes, while \( A_H \setminus \text{Name}_H \) does not (first interference). On the other hand, from the viewpoint of the low-level client, the time spent by the server to deliver the high-level services describes an observable busy waiting phase (second interference).

From a performance-oriented perspective, these interferences affect the low-level throughput of the client-server system, in terms of number of \( s_L \) actions executed per unit of time, which is higher in the absence of high-level interactions. The removal of the two interferences requires strong control mechanisms that, as expected, aim at degrading the performance of the system in order to make the behavior of the high-level client transparent to the low-level client.

As far as the first interference is concerned, it is necessary to confine the behavior of the high-level client in order to hide its impact on the timing of the system activities. This can be done by employing a sort of high-level box which, somehow, limits and controls the activities performed by the high-level client. Formally, we replace process \( A_H \) with the following one:

\[
H_{\text{box}} \triangleq <\tau, \infty_{l,w} >.
\]

\[
(\infty_{h+1,w} < a_H, \lambda_H > . H_{\text{box}} + \infty_{h,w} < \tau > . H_{\text{box}})
\]

The action \( <\tau, \infty_{l,w} > \) represents the activation of the box and is technically needed because it allows \( \approx_{\text{EMB}} \) to abstract from the subsequent intermediate immediate \( \tau \)-actions. The branch guarded by the action \( <\tau, \infty_{h,w} > \) is enabled if and only if the high-level client is absent. Its role is to simulate the presence of such a client in a way that makes the high-level behavior invisible to the low-level client. Intuitively, process \( H_{\text{box}} \) is a black box acting as an interface between the system and the high-level client, who can interact with such a box by pushing a button whenever he decides to communicate with the system.

As far as the second interference is concerned, making the server transparent to the low-level client is achieved by following an approach borrowed by round-robin scheduling strategies. In practice, the intuition is similar to that underlying the definition of process \( H_{\text{box}} \). The service activities are divided into temporal slots, each one dedicated to a class of clients in a round-robin fashion. Independently of the presence of a pending request from a client of the currently managed class, the temporal slot is spent. In this way a low-level client cannot deduce whether the high-level slot has been actively exploited by the high-level client. Formally, we replace process \( S \) with the following one:

\[
S_L \triangleq <d_L, \infty_{f(L),w} > . <s_L, \mu_L > . S_H + <\tau, \mu_L > . S_H
\]

\[
S_H \triangleq <d_H, \infty_{f(H),w} > . <s_H, \mu_H > . S_L + <\tau, \mu_H > . S_L
\]

This is not enough to hide completely the interference of the high-level client. Indeed, whenever such a client is blocked because the related buffer is not available to accept further requests, then process \( H_{\text{box}} \) does not compete for the resource time. This observable behavior would reveal to the low-level client that