Security Analysis of a Probabilistic Non-repudiation Protocol

Alessandro Aldini and Roberto Gorrieri

Università di Bologna, Dipartimento di Scienze dell'Informazione
Mura Anteo Zamboni 7, 40127 Bologna, Italy
{aldini,gorrieri}@cs.unibo.it

Abstract. Non-interference is a definition of security introduced for the analysis of confidential information flow in computer systems. In this paper, a probabilistic notion of non-interference is used to reveal information leakage which derives from the probabilistic behavior of systems. In particular, as a case study, we model and analyze a non-repudiation protocol which employs a probabilistic algorithm to achieve a fairness property. The analysis, conducted by resorting to a definition of probabilistic non-interference in the context of process algebras, confirms that a solely nondeterministic approach to the information flow theory is not enough to study the security guarantees of cryptographic protocols.

1 Introduction

The original notion of non-interference [14] has been proposed to analyze the kind of information flowing through computer systems. More precisely, assuming a classification which divides the information into (high level) confidential data and (low level) public data, the goal of such a property consists of verifying the presence of an insecure information flow from high level to low level. The formalization of non-interference based security properties is a topic that has received a lot of interest in the framework of nondeterministic process algebras (see, e.g., [19, 10, 11]). However, as emphasized in the literature (see, e.g., [15, 20, 1, 9, 22]), the security analysis of real systems should take into consideration further aspects of the system behavior, such as time and probabilities, that may cause undesirable information flows which are not observable in a solely nondeterministic context. In [1, 2] we exploited the well-established theory developed in process algebras to extend the non-interference theory based on nondeterminism to a probabilistic setting, by emphasizing the relation between the two frameworks.

In this paper, we present a simple but effective case study showing that a purely nondeterministic approach to the analysis of information flow is not enough to derive significant results, because the probabilistic behavior of the system we consider is the most important aspect affecting the security guarantees. In particular, we model and analyze a non-repudiation protocol [17] based on a probabilistic algorithm which allows two participants to exchange information in a fair way without resorting to a trusted third party. In the context

H. Hermanns and R. Segala (Eds.): PAPM-PROBMIV 2002, LNCS 2399, pp. 17–36, 2002.
© Springer-Verlag Berlin Heidelberg 2002

of cryptographic protocols, fairness means that during the protocol, either both parties obtain their expected information, or the probability that one of them gains information while the other one gains nothing is less than a negligible threshold ε. Therefore, in order to evaluate the security level of the protocol, we have to estimate ε, which is the most important parameter affecting the satisfaction of the fairness property of the system. With this in view, by employing a qualitative, nondeterministic approach based on the classical, binary view according to which the system is (or is not) secure, we cannot obtain a *quantitative estimate* of potential, unwanted information flows so that an approach which takes into consideration probabilities is needed.

In the following, we describe in Sect. 2 the probabilistic non-repudiation protocol [17] and we introduce in Sect. 3 a process algebraic approach to the probabilistic non-interference theory [1, 2, 4]. In Sect. 4 we define the security property we intend to check and then we analyze the fairness condition for two different models of the probabilistic algorithm employed by the non-repudiation protocol. In Sect. 5 some conclusions terminate the paper.

2 A Probabilistic Non-repudiation Protocol

In a message exchange based protocol, repudiation consists of the denial by one of the entities involved in the communication of having participated in all or part of the protocol (see, e.g., [16, 23]). In this context, the *non-repudiation of origin* is intended to prevent the *originator* of a message from denying having sent the message, while the *non-repudiation of receipt* is intended to prevent the *recipient* of a message from denying having received the message. Especially in electronic commerce, non-repudiation services are mandatory if we want to guarantee the validity of a transaction, against any attempt to repudiate all or part of the transaction (e.g., if a payment for a service is represented by an acknowledgment, the client could refuse to send the ack message when the service has been delivered). Usually, to achieve non-repudiation services, a third party is involved by the participants as a trusted authority that comes into play in case of future disputes. Here, we consider an interesting alternative proposal that offers the non-repudiation service, obtained with a certain probability, without resorting to a trusted third party [17]. Such a protocol guarantees a fair exchange of a message, sent by the originator O which offers a service, for an acknowledgment, sent by the recipient R which is expected to confirm the received service. In particular, the probabilistic protocol is ε-*fair*, i.e. at each step of the protocol run, either both parties receive their expected information, or the probability that a cheating party gains any valuable information, while the other party gains nothing, is less than ε.

Before describing the protocol, we introduce some notation and assumption. We suppose that an authentication phase precedes the non-repudiation protocol, during which the involved parties exchange their public keys of a public key cryptosystem. We denote by $Sign_E(M)$ the message M encrypted with the private key of the entity E. The number n of steps for the protocol is decided

Table 1. Probabilistic non-repudiation protocol

1.	$R \rightarrow O$	: $Sign_R(\text{request}, R, O, t)$
2.	$O \rightarrow R$	: $Sign_O(f_n(M), O, R, t)$
3.	$R \rightarrow O$	: $Sign_R(\text{ack}_1)$
$\vdots$		
$2n$.	$O \rightarrow R$	: $Sign_O(f_1(M), O, R, t)$
$2n+1$.	$R \rightarrow O$	: $Sign_R(\text{ack}_n)$

by the originator, that also computes n functions $f_1, \ldots, f_n$ which are parts of a function composition. In particular, we have $f_n(M) \circ f_{n-1}(M) \circ \ldots \circ f_1(M) = M$. Finally, we use t to denote a timestamp which each message is enriched with.

A description of the protocol is as shown in Table 1, where with the notation $R \rightarrow O : Msg$ we express a message Msg sent by R and received by O. The recipient R starts the protocol by sending a signed, timestamped request for a service to the originator O, that in turns secretly decides the number n of steps and then sends the first signed, timestamped message containing $f_n(M)$. For each received message containing $f_i(M)$, the recipient sends a related signed, timestamped acknowledgment message containing $\text{ack}_i = (i, R, O, t)$. Since the recipient does not know n, he cannot determine when the protocol will end and, as a consequence, when he will receive the final message which allows him to compute M. In particular, since the functions f_i are sent in reverse order and since we assume that the composition operator does not permit the commutativity, the recipient cannot use intermediate results to speed up the computations in order to guess M before the reception of the last message. Upon the reception of the ack related to the last message containing $f_1(M)$, the originator correctly terminates the protocol. Note that an *end of protocol* message sent by the originator is not mandatory. Indeed, after not receiving further messages, the recipient is aware of the protocol state and is able to compute M.

It is worth noting that each message conveys a timestamp, which can be considered as a random value used to guarantee the freshness of the message and to protect the parties against replication attacks. Intuitively, the non-repudiation of origin is guaranteed by the sequence of messages $Sign_O(f_i(M), O, R, t)$, with $i \in \{1, \ldots, n\}$, while the non-repudiation of receipt is given by the last message $Sign_R(\text{ack}_n)$. If the protocol terminates after the delivery of the nth ack message, both parties obtain their expected information and the protocol is fair. If the protocol terminates before sending the message containing $f_1(M)$, then neither the originator nor the recipient obtain any valuable information, so that the fairness is preserved. Since the number n of steps is not revealed by the originator during the protocol run, a strategy for a dishonest recipient can consists of (i) guessing the last message containing $f_1(M)$ by computing the composition of the received functions f_i, and (ii) violating the fairness of the protocol by blocking

the transmission of the last ack message. Therefore, the key to success of the protocol is the immediacy in sending back the ack messages, so that if a malicious recipient delays each ack message in order to compute the composition of the received functions f_i to see if he has obtained the message M, then the originator realizes this unfair strategy and can stop the protocol. To this end, the choice of the functions f_i must be in such a way that the composition computation takes more time than the transmission of an ack message. In this way, the originator decides a deadline for the reception of each ack, after which, if the ack message is not received, the protocol is stopped. Anyway, a malicious recipient can try to randomly guess the number of steps of the protocol. We point out that the probability for the recipient of guessing the last message depends on the size of the sampling space and the probability distribution chosen by the originator to compute n. In the next section, we propose a formal approach which includes all the ingredients needed to evaluate such a potential unwanted behavior of the recipient.

We conclude this section by showing a concrete proposal [17] based on the above generic non-repudiation algorithm, which noticeably simplifies the choice of the functions f_i. We take f_n as a ciphering function which uses a secret key k, namely $f_n(M) = \{M\}_k$, f_{n-1} until f_2 as random garbage, and $f_1(M)$ as the key k in clear. In this way, the recipient immediately obtains M encrypted with an unknown key and, before the last transmission, he does not receive any valuable information. Only upon the reception of the last message, the recipient obtains the key needed to compute M. Against a cheating recipient, the cryptosystem must be adequately chosen, in such a way that the time needed to verify a key, by deciphering the message, is to be too long with respect to the transmission time for an ack message. For instance, the most efficient implementations of RSA guarantee a ciphering of 600 Kbit per second; therefore the greater the message size, the longer the transmission delay for the ack messages which can be tolerated by the originator.

3 A Process Algebraic Approach to the Probabilistic Non-interference Theory

In this section, we describe an adequate technique for the formal description and the security analysis of cryptographic protocols like the one described in Sect. 2. In particular, we employ the process algebra proposed in [3] and used in [1, 2, 4] to extend the classical nondeterministic approach to non-interference theory with probabilities. Such a calculus derives from a simple nondeterministic CSP-like process algebra where actions are syntactically divided into input actions and output actions. The extension is obtained by adding probabilistic information to the algebraic operators and by employing an appropriate model of probabilities. In particular, the model of probabilities adopted in our calculus is an integration of the generative and reactive approaches in the line of [13], where we assume the output actions behaving as *generative* actions (a generative process autonomously decides, on the basis of a probability distribution, which

action will be executed and how to behave after such an event) and the input actions behaving as *reactive* actions (a reactive process reacts internally to an external action of type a, chosen by the environment, on the basis of a probability distribution associated with the reactive actions of type a it can perform).

The labeled transition system associated with a term of our probabilistic calculus, called generative-reactive transition system ($GRTS$), consists of a bundle of generative transitions representing all the output actions that the system can perform, and several bundles of reactive actions (one for each action type) representing the input actions enabled by the system. The choice within a bundle is purely probabilistic, while the choice among bundles is nondeterministic.

Formally, $AType$ denotes the set of action types, ranged over by $a, b, \ldots$, including also the special type τ denoting an internal action. We denote the set of reactive actions by $RAct = \{a_* \mid a \in AType - \{\tau\}\}$ and the set of generative actions by $GAct = AType$ (note that τ is a generative action, because it expresses an autonomous internal move that does not react to external stimuli). The set of actions is denoted by $Act = RAct \cup GAct$, ranged over by $\pi, \pi', \ldots$. The set $\mathcal{L}$ of process terms is generated by the syntax:

$$P ::= \underline{0} \mid \pi.P \mid P +^p P \mid P \parallel^p_S P \mid P \backslash L \mid P/^p_a \mid A$$

where $S, L \subseteq AType - \{\tau\}$, $a \in AType - \{\tau\}$, and $p \in]0, 1[$. The set $\mathcal{L}$ is ranged over by $P, Q, \ldots$. As usual in security models, we distinguish among high level visible actions and low level visible actions by defining two disjoint sets $AType_H$ of high level types and $AType_L$ of low level types that form a covering of $AType - \{\tau\}$, such that $a \in GAct$ and $a_* \in RAct$ are high (low) level actions if $a \in AType_H$ ($a \in AType_L$). Now, we give an informal intuition of the operators, while we delay a complete presentation of their semantics to the appendix.

- $\underline{0}$ represents the terminated or deadlocked term.
- $\pi.P$ performs the action π with probability 1 and then behaves like P.
- $P +^p Q$ is a CCS-like alternative choice operator, where a mixed nondeterministic and probabilistic choice among the actions of P and Q is performed. More precisely, $P +^p Q$ executes a generative (reactive of type a) action of P with probability p and a generative (reactive of type a) action of Q with probability $1 - p$. If one process P or Q cannot execute generative (reactive of type a) actions, $P +^p Q$ chooses a generative (reactive of type a) action of the other process with probability 1. The choice among generative and reactive actions and among reactive actions of different type is purely nondeterministic.
- $P \parallel^p_S Q$ is a CSP-like parallel composition operator, where the actions not belonging to the type set S are locally executed, while the actions belonging to S are constrained to synchronize. In particular, a synchronization between two actions may occur only if either they are both input actions of the same type a (and the result is an input action of type a), or one of them is an output action of type a and the other one is an input action of type a (and the result is an output action of type a). The probabilistic choice mechanism among the actions of P and Q is the same as that described for the choice operator.

- $P\backslash L$ prevents the execution of the actions of type in L.
- $P/^p_a$ turns generative and reactive actions of type a into generative actions τ. Parameter p expresses the probability that actions τ obtained by hiding reactive actions a_* of P are executed with respect to the generative actions previously enabled by P. Parameter p is, instead, not used when hiding generative actions, because the choice among generative actions is already probabilistic. As an example, consider process $P \stackrel{\Delta}{=} a_* +^q b$, where the choice is purely nondeterministic (parameter q is not considered), and hide the action a_*. The semantics of $P/^p_a \stackrel{\Delta}{=} \tau +^p b$ is a probabilistic choice between the action τ obtained by hiding the action a_* and the action b, performed according to probabilities p and $1-p$, respectively. We turn reactive actions into generative internal actions τ, because the hiding operator is used to obtain fully specified systems from open systems (i.e. systems enabling reactive choices). In particular, when fully specifying a system, the nondeterministic choices due to the possible interactions with the environment have to be resolved, and parameter p turns such choices into probabilistic choices.
- Constants A are used to specify recursive systems. In general, when defining an algebraic specification, we assume a set of constants defining equations of the form $A \stackrel{\Delta}{=} P$ (with P a guarded term [18]) to be given.

In the following, we denote by $\mathcal{G}$ the set of guarded and closed terms of $\mathcal{L}$ [18] and by $sort(P)$, with $P \in \mathcal{G}$, the set of actions which syntactically occur in the action prefix operators within P. Moreover, let $\mathcal{G}_H = \{P \in \mathcal{G} \mid sort(P) \subseteq AType_H\}$ be the set of high level terms. For the sake of simplicity, we use the following abbreviations. We assume the parameter p to be equal to $\frac{1}{2}$ whenever it is omitted from any probabilistic operator. Moreover, when it is clear from the context, we use the abbreviation P/S, with $S = \{a_1, \ldots, a_n\}$, to denote any expression of the form $P/^{p_1}_{a_1} \ldots /^{p_n}_{a_n}$, $\forall p_i \in]0,1[, i \in \{1 \ldots n\}$, which hides the actions of type in $S \subseteq AType - \{\tau\}$. This notation simplifies the description of those terms where, e.g., we hide generative actions only and the probabilistic parameters of the hiding operators are not meaningful.

To conclude the presentation of the calculus, we report the definition of weak bisimulation for *GRTS*s [1], based on which we define the security properties. Such an equivalence relation is inspired by [5], where the classical relation $\approx$ of [18] is replaced by a function *Prob* that computes the probability of reaching classes of equivalent states. In particular, we recall that each transition of a *GRTS* is labeled with an action in the set *Act* and a probability, and that the probability associated to a sequence of transitions is given by the product of the probabilities of the transitions. Then, by $Prob(P, \tau^*\hat{a}, C)$ [1] we denote the overall probability associated to the set of transition sequences with initial state P which lead to a system state in the set C via a sequence of transitions, whose ordered labels generate a sequence of the form $\tau^*\hat{a}$.

[1] $\tau^*\hat{a}$ stands for the sequence τ^*a if $a \in GAct - \{\tau\}$ and for the sequence τ^* if $a = \tau$. Note that τ^* means zero or more τ occurrences.

Definition 1. *An equivalence relation $R \subseteq \mathcal{G} \times \mathcal{G}$ is a probabilistic weak bisimulation if and only if, whenever $(P, Q) \in R$ then for all $C \in \mathcal{G}/R$ we have that $Prob(P, \tau^* \hat{a}, C) = Prob(Q, \tau^* \hat{a}, C)$ for all $a \in GAct$, and $Prob(P, a_*, C) = Prob(Q, a_*, C)$ for all $a_* \in RAct$.*

Two terms $P, Q \in \mathcal{G}$ are weakly probabilistic bisimulation equivalent, denoted $P \approx_{PB} Q$, if there exists a probabilistic weak bisimulation R containing the pair (P, Q).

Based on the above process algebra and the related notion of equivalence, several security properties can be defined. The reader interested in a classification of security properties for processes of our calculus should refer to [2, 4]. As an example, here we propose the probabilistic version of the bisimulation based Non Deducibility on Composition (*PBNDC*, for short), which informally says that, from the low level standpoint, the probability distribution of the low level events as observed in a system P in isolation is not to be altered when considering the potential interactions of P with any high level user.

Definition 2. *Let $P \in \mathcal{G}$. We say that $P \in PBNDC$ if and only if*

$$P \backslash AType_H \approx_{PB} ((P \parallel_S^p \Pi)/S) \backslash AType_H \quad \forall p \in]0, 1[, \Pi \in \mathcal{G}_H, S \subseteq AType_H.$$

$P \backslash AType_H$, where the execution of the high level actions is prevented, represents the low behavior of P in isolation, i.e. when every high level activity is absent, while $((P \parallel_S^p \Pi)/S) \backslash AType_H$ models the low behavior of P which possibly interacts with any high level user Π and such interactions are unobservable by the low level user. Hence, the *PBNDC* property checks the equivalence between these two different views of the low behavior of P, which turns out to be secure if the low level user is not able to reveal the potential interference of the high level user on the system behavior. Such a definition extends the original nondeterministic *BNDC* property [10], in that it also considers the influence of the probabilistic high behavior upon the probabilistic low view. We point out that in [5] an algorithm is described that computes probabilistic weak bisimulation equivalence classes in time $\mathcal{O}(n^3)$ and space $\mathcal{O}(n^2)$, where n is the number of states. Hence, by defining a process Π which affects the low level view of the system, we are able to formally capture, through the automatic verification provided by probabilistic weak bisimulation laws, those interferences which reveal confidential information to low level users.

Example 1. Let us consider the system $P \stackrel{\Delta}{=} l.\underline{0} +^p h.h.l.\underline{0}$, where $l \in AType_L$ and $h \in AType_H$. The high level part alters the low view of the system in the case the high level user first executes the first action h and then blocks the execution of the second action h. In such a case a deadlock is caused by the high level user that prevents the action l from being observed by a low level user. This information flow is captured by the *PBNDC* property if we take $S = \{h\}$ and $\Pi \stackrel{\Delta}{=} h_*.\underline{0}$. Indeed, in such a case we have $P \backslash AType_H \approx_{PB} l.\underline{0} \not\approx_{PB} l.\underline{0} +^p \tau.\underline{0} \approx_{PB} (((l.\underline{0} +^p h.h.l.\underline{0}) \parallel_S (h_*.\underline{0}))/S) \backslash AType_H$. □

By modeling the probabilistic behavior a quantitative estimate of the information flowing through the system can be obtained. More precisely, with respect to the nondeterministic case, where we can just verify the presence of an information leakage, in our setting we can measure the probability of observing an unwanted information flow from high level to low level. To this end, we need a quantitative notion of behavioral equivalence for deciding if two probabilistic processes behave almost (up to small ε-fluctuations) the same or, more formally, for measuring the distance between probabilistic transition systems (see, e.g., [8, 7]). In [4], we formalize the notion of bisimulation equivalence which takes into consideration ε-perturbations in the context of *GRTS*s.

Definition 3. *An equivalence relation $R \subseteq \mathcal{G} \times \mathcal{G}$ is a probabilistic weak bisimulation with ε-precision, where $\varepsilon \in]0,1[$, if and only if, whenever $(P,Q) \in R$, then for all $C \in \mathcal{G}/R$*

- $|\, Prob(P, \tau^* \hat{a}, C) - Prob(Q, \tau^* \hat{a}, C) \,| \leq \varepsilon \quad \forall a \in GAct$
- $|\, Prob(P, a_*, C) - Prob(Q, a_*, C) \,| \leq \varepsilon \quad \forall a_* \in RAct$

Two terms $P, Q \in \mathcal{G}$ are ε-bisimulation equivalent, denoted $P \approx_{PB_\varepsilon} Q$, if there exists a probabilistic weak bisimulation with ε-precision R containing the pair (P,Q).

By replacing in the definition of any security property, say *SP*, the relation $\approx_{PB}$ by the relation $\approx_{PB_\varepsilon}$, we make the condition verified by *SP* less restrictive in the sense that it can tolerate those small fluctuations which the observer considers scarcely significant for the security level of the system.

Example 2. Let us consider the system $P \stackrel{\Delta}{=} h.l'.\underline{0} +^p \tau.(l.\underline{0} +^q h.l.\underline{0})$, where $l, l' \in AType_L$ and $h \in AType_H$. P is clearly not secure because the execution of l' reveals to a low level observer that h has been executed. In particular, if we employ the probabilistic weak bisimulation $\approx_{PB}$, then P is not *PBNDC* secure, because $P \backslash AType_H \approx_{PB} \tau.l.\underline{0} \not\approx_{PB} \tau.l'.\underline{0} +^p \tau.(l.\underline{0} +^q \tau.l.\underline{0}) \approx_{PB} ((P \parallel_{\{h\}} (h_*.\underline{0}))/\{h\}) \backslash AType_H$. Anyway, we also have that P is *PBNDC* secure if the equivalence relation we employ is the probabilistic weak bisimulation with p-precision $\approx_{PB_p}$. To this end, we just observe that $\tau.l.\underline{0} \approx_{PB_p} \tau.l'.\underline{0} +^p \tau.l.\underline{0} \approx_{PB} \tau.l'.\underline{0} +^p \tau.(l.\underline{0} +^q \tau.l.\underline{0})$. Hence, depending on a tolerance ε, we can decide if P is to be considered secure enough. □

4 Security Analysis of the Non-repudiation Protocol

In this section, we employ our process algebraic approach to model the concrete proposal of the protocol described in Sect. 2 in order to verify the non-repudiation property. To this end, we resort to the following assumptions. Since the non-repudiation property we intend to check depends on the order and the number of packets which are exchanged, we abstract from the cryptosystem used within the protocol and we simply model the packet exchange between the two involved parties. Moreover, we also abstract from the channel and the transmission delays,

Table 2. Probabilistic non-repudiation protocol - *Originator* model

$$
\begin{aligned}
Orig &\triangleq accept_request_{*}.snd_first_msg.ack_{*}.O' \\
O' &\triangleq O_1 +^{1/\eta} (O_2 +^{1/(\eta-1)} \ldots (O_{\eta-1} +^{1/2} O_\eta) \ldots) \\
O_i &\triangleq snd_msg.(ack_{*}.O_{i-1} + stop_{*}.\underline{0}) \quad \text{if } 2 \leq i \leq \eta \\
O_1 &\triangleq snd_msg.(ack_{*}.stop.\underline{0} + stop_{*}.\mathbf{unfair}.\underline{0})
\end{aligned}
$$

by assuming that a message which is delayed (not sent) by a participant is not delivered to the other participant. As far as the probabilistic algorithm is concerned, we model two different scenarios, depending on the probability distribution used by the originator to sample the number n of protocol steps. In the first scenario we assume an equally distributed probability, while in the second one we assume a Bernoulli distribution. In both cases, we show that the protocol is ε-fair, where the parameter ε is decided by the honest participant.

We start by introducing the models of an originator and of a recipient which behave correctly. The algebraic specification of the originator is shown in Table 2. Process *Orig* is always ready to start the message exchange by (i) accepting a request (action $accept_request_{*}$), (ii) sending the first message containing the information M encrypted with a secret key k (action *snd_first_msg*), and (iii) receiving the first ack message (action ack_{*}). Then, by assuming that the number n of steps of the protocol is chosen in the set $\{1, \ldots, \eta\}$ with an equally distributed probability, in term O' a value for n is sampled by performing a probabilistic choice among η generative actions of type *snd_msg*, where the probability associated with each action is $1/\eta$. If the action *snd_msg* of term O_n is the winning action, then the originator sends the first message (of n messages) to the recipient and waits for the related ack message. Then, if the ack is received through the reactive action ack_{*}, term O_{n-1} is reached and the next step is scheduled. On the other hand, if the ack is not received before the expected deadline the protocol is prematurely stopped. Since we abstract from time, we perform a nondeterministic choice between the reactive action ack_{*} and a reactive action $stop_{*}$ to model the time-out. Moreover, for the sake of simplicity, we assume that whenever an ack message is sent by the recipient then the originator receives it before the deadline. The fair, correct termination of the protocol is reached when the originator receives the last ack message and performs the generative action *stop* (see term O_1). The protocol terminates in an unfair way if and only if the reactive action $stop_{*}$ is executed when the originator is in term O_1. Indeed, in such a case, the originator has already sent the last message containing the key k which reveals M, but he has not received the final ack message which completes the fair exchange of information. We make explicit such a situation by performing the special action **unfair**.

Table 3. Probabilistic non-repudiation protocol - Honest *Recipient* model

$$\begin{aligned} H_Recip &\stackrel{\Delta}{=} accept_request.snd_first_msg_*.ack.HR \\ HR &\stackrel{\Delta}{=} snd_msg_*.ack.HR + stop_*.\underline{0} \end{aligned}$$

In Table 3, we show the algebraic specification of a recipient that behaves correctly. Process *H_Recip* starts the protocol by sending a request (action *accept_request*), and then it waits for the first message (action $snd_first_msg_*$). Upon the reception of such a message, it sends the first ack (action *ack*) and then behaves like term *HR*, where each time a message is received through the action snd_msg_*, a related ack message is sent by executing the action *ack*. The protocol terminates when the reactive action $stop_*$ is executed to denote that the last received message contains the key k in clear needed to decrypt M.

To complete the algebraic description of the protocol, we specify the communication interface between the two parties by composing in parallel process *Orig* and process *H_Recip* in the system $Orig \parallel_S^p H_Recip$, where the two processes interact by synchronizing on the actions in the set $S = \{accept_request, snd_first_msg, ack, snd_msg, stop\}$ containing all the actions except for the special action **unfair**. Note that parameter p is not considered (in the following we omit it), because each state of the composed system is such that a probabilistic choice among actions of process *Orig* and of process *H_Recip* is never to be performed.

In order to verify a security property in the line of the method described in Sect. 3, we have to single out the high level actions and the low level ones. Here, we follow an idea proposed in the general case for the analysis of cryptographic protocols in [12]. Since the recipient, which can be considered a potential malicious party, has the complete control of the network, it is reasonable to assume that the names used for message exchange (i.e., all the actions in S) are the high level actions. Instead, the low level actions are extra observable actions that we include into the model to observe the non-repudiation property. With this in view, it is natural to consider the extra action **unfair**, put in the algebraic specification of the originator, as the unique low level action of the system which is executed when the protocol terminates without preserving the fairness condition.

Now, we have to formalize the non-repudiation property that will be checked. To this end, we intend to employ the *PBNDC* property described in Sect. 3. With respect to the original definition of *PBNDC*, where the system behavior is observed when put in parallel with the external environment, in the case of the non-repudiation property a malicious behavior is due to an internal component of the system itself which tries to obtain a valuable information violating the fairness condition. Therefore, potential attacks by third parties which are not

Table 4. Probabilistic non-repudiation protocol - Malicious *Recipient* model

$$
\begin{aligned}
M_Recip &\stackrel{\Delta}{=} accept_request.snd_first_msg_*.ack.MR_1 \\
MR_i &\stackrel{\Delta}{=} snd_msg_*.(stop.\underline{0} +^{1/(\eta-i+1)} ack.MR_{i+1}) \quad \text{if } 1 \leq i < \eta \\
MR_\eta &\stackrel{\Delta}{=} snd_msg_*.stop.\underline{0}
\end{aligned}
$$

involved in the protocol are not meaningful. In particular, forgery and authentication attacks are prevented by the signed, exchanged messages (we assume that the initial authentication phase is secure so that both parties know the public keys to decrypt the messages), and replication attacks are prevented by the timestamps. Since the originator has not interest in stopping the protocol before the end, the hostile environment is represented by the recipient that possibly tries to obtain its expected information without sending the last ack message. According to this, to verify the non-repudiation property we check the semantic equivalence between the model of the protocol where both parties behave honestly and each possible model of the protocol involving a potentially malicious recipient. More formally, the property to be checked is as follows.

The protocol satisfies the non-repudiation property if and only if:

$$((Orig \parallel_S H_Recip)/S)\backslash AType_H \approx_{PB} ((Orig \parallel^p_{S'} Recip)/S')\backslash AType_H$$

for all $S' \subseteq AType_H, p \in]0,1[$, *and for any* $Recip \in \mathcal{G}_H$.

Intuitively, we observe that the protocol does not satisfy the property above. In particular, if both participants behave honestly, then the unfair behavior cannot be executed; instead, it is possible to find a malicious recipient that receives the expected information and denies sending the final non-repudiation of receipt message.

Formally, process *H_Recip* never stops the protocol prematurely by executing the generative action *stop*, so the reactive action $stop_*$ in term O_1 is never enabled and the low level action **unfair** cannot be executed. Hence, if both participants are honest, the protocol is fair with probability 1. In particular, it is easy to see that $Prob(((Orig \parallel_S H_Recip)/S)\backslash AType_H, \tau^*\mathbf{unfair}, C) = 0 \ \forall C \in \mathcal{G}/\approx_{PB}$ and $\sum_{C \in \mathcal{G}/\approx_{PB}} Prob(((Orig \parallel_S H_Recip)/S)\backslash AType_H, \tau^*, C) = 1$.

On the other hand, we now describe a malicious recipient model *Recip* which allows the low action **unfair** to be executed in $((Orig \parallel^p_{S'} Recip)/S')\backslash AType_H$. To this aim, we take $S' = S$, any $p \in]0,1[$, and as term *Recip* the process *M_Recip* shown in Table 4, which models the malicious recipient with the maximum knowledge of the protocol variables, i.e. he knows both the sampling space size and the probability distribution associated with each possible event of the sampling space chosen by the originator to compute the number n of steps of the protocol. Based on such a knowledge, process *M_Recip* is able to maximize the probability of guessing the last message of the protocol by adopting the following strategy.

For each step of the protocol, the probability of receiving the last message is 1 over the maximum number of steps which can remain before the end of the protocol. Therefore, upon the reception of the first message represented by the execution of the action of type *snd_msg*, the recipient stops the protocol by executing the generative action *stop* with probability $\frac{1}{\eta}$, where η is the size of the sampling space, and sends the ack message by executing the generative action *ack* with probability $1 - \frac{1}{\eta}$. If the recipient decides to send the ack and the protocol is not terminated yet, upon the reception of the second message, the recipient stops the protocol with probability $\frac{1}{\eta-1}$, because the probability of guessing the last step is redistributed among the $\eta - 1$ possible events of the reduced sampling space (see terms MR_i). Finally, if after $\eta-1$ steps the protocol is not terminated yet, the recipient knows that the next step must be the final one, so that he decides to stop the protocol with probability 1 (see term MR_η). Intuitively, if we want to compute by hand the probability for the recipient of cheating with success, we perform the following steps:

- the probability that the originator samples the value 1 is $\frac{1}{\eta}$ and the probability that the recipient stops the protocol after a single step is $\frac{1}{\eta}$, therefore the probability of guessing the event $n = 1$ is $\frac{1}{\eta} \cdot \frac{1}{\eta} = \frac{1}{\eta^2}$;
- the probability that the originator samples the value 2 is $\frac{1}{\eta}$ and the probability that the recipient stops the protocol after two steps is $\frac{\eta-1}{\eta} \cdot \frac{1}{\eta-1} = \frac{1}{\eta}$, where $\frac{\eta-1}{\eta}$ is the probability of sending the first ack and $\frac{1}{\eta-1}$ is the probability of stopping the protocol at the second step. Therefore the probability of guessing the event $n = 2$ is $\frac{1}{\eta} \cdot \frac{1}{\eta} = \frac{1}{\eta^2}$;
- the probability that the originator samples the value $i > 2$ is $\frac{1}{\eta}$ and the probability that the recipient stops the protocol after i steps is $\frac{\eta-1}{\eta} \cdot \frac{\eta-2}{\eta-1} \cdot \ldots \cdot \frac{1}{\eta-i+1} = \frac{1}{\eta}$, therefore the probability of guessing the event $n = i$ is $\frac{1}{\eta} \cdot \frac{1}{\eta} = \frac{1}{\eta^2}$.

Given that the sampling space is composed of η possible events, the overall probability of guessing the last message is $\eta \cdot \frac{1}{\eta^2} = \frac{1}{\eta}$, exactly the value we expected.

Formally, by analyzing the *GRTS* associated to the composed system, we immediately obtain that $Prob(((Orig \parallel_S M_Recip)/S)\backslash AType_H, \tau^* \mathbf{unfair}, C) = \frac{1}{\eta}$ (where C is the equivalence class of the terminated term). Hence, the non-repudiation protocol is not *PBNDC* secure if we employ as the equivalence relation the weak probabilistic bisimulation $\approx_{PB}$. Moreover, if we resort to the definition of ε-bisimilarity, we can add that the protocol is secure with $\frac{1}{\eta}$-tolerance. Indeed, as we can formally verify, we have that $((Orig \parallel_S H_Recip)/S)\backslash AType_H \approx_{PB\frac{1}{\eta}} ((Orig \parallel_S M_Recip)/S)\backslash AType_H$. Therefore, the level of security of the protocol is parameterized by the size of the sampling space η.

By calculating probabilistic weak bisimulation equivalence classes, we have formally derived the same result obtained computationally in [17]. On the other hand, we think that our technique can be automated and generalized, by following an approach inspired by [12], for the analysis of not only non-repudiation,

Table 5. Probabilistic Non-repudiation Protocol - *Originator* alternative model

$$
\begin{aligned}
OrigB &\triangleq accept_request_{*}.snd_first_msg.ack_{*}.OB' \\
OB' &\triangleq snd_msg.Last_msg +^{p} snd_msg.OB'' \\
OB'' &\triangleq ack_{*}.OB' + stop_{*}.\underline{0} \\
Last_msg &\triangleq ack_{*}.stop.\underline{0} + stop_{*}.\mathbf{unfair}.\underline{0}
\end{aligned}
$$

but also other security properties related to the algebraic specification of cryptographic protocols. Moreover, it is worth noting that in a purely nondeterministic setting [2], we could only conclude that the protocol of [17] does not satisfy the non-repudiation of receipt, since an insecure behavior is observable. Instead, in the probabilistic setting we have seen that the insecure behavior caused by the dishonest recipient *M_Recip* can be estimated and can be acceptable for the honest participant, that is in charge to decide the risk factor ε affecting the security condition.

As another example of the relation between the behavior of a cheating recipient and the behavior of a honest originator, we now present an alternative model of the system where the originator follows a Bernoulli distribution with parameter p as the particular probability distribution chosen to sample the duration of the protocol. With such an example we emphasize that the probability of cheating for a dishonest recipient depends on a parameter which is under the control of the honest participant.

The originator model is depicted in Table 5. After the initial set-up phase, at each step of the protocol (see term OB') the originator sends either the last message with probability p and then behaves as term *Last_msg*, or the next garbage message with probability $1-p$ and then behaves as term OB'', where the reception of the related ack message leads to term OB' again. Such a version of the protocol is *time-bounded* (i.e. the protocol will be completed before a finite amount of time), because, even if the protocol is never stopped in term OB'' via the execution of the action $stop_{*}$, the probability of reaching state *Last_msg* (where the protocol terminates with probability 1) from the initial state is given by:

$$\sum_{i=0}^{\infty} p \cdot (1-p)^i = p \cdot \sum_{i=0}^{\infty} (1-p)^i = p \cdot \frac{1}{1-(1-p)} = 1$$

In Table 6, we show the model of a cheating recipient that follows a Bernoulli distribution with parameter q to decide either to send the ack message or to stop the protocol. In particular, after the initial message exchange (see term *RecipB*),

[2] We can omit the probabilistic information from the operators of the probabilistic calculus, thus obtaining a process algebra on which the nondeterministic non-interference theory can be rephrased [2].

Table 6. Probabilistic Non-repudiation Protocol - *Recipient* alternative model

$$\begin{aligned} RecipB &\stackrel{\Delta}{=} accept_request.snd_first_msg_*.ack.RB \\ RB &\stackrel{\Delta}{=} snd_msg_*.(stop.\underline{0} +^q ack.RB) + stop_*.\underline{0} \end{aligned}$$

in term RB the recipient accepts each arriving message (action snd_msg_*) and then either sends the related ack message (the action *ack* is executed with probability $1-q$) or tries to compute M by employing the last received key (the action *stop* is executed with probability q). In Fig. 1 we show the *GRTS* associated to the composed system $((OrigB \parallel_S RecipB)/S)\backslash AType_H$. State s_1 corresponds to term $((OB' \parallel_S RB)/S)\backslash AType_H$ from which the system evolves either into state $s_2 = ((OB'' \parallel_S (stop.\underline{0} +^q ack.RB))/S)\backslash AType_H$ with probability $1-p$, denoting that the protocol is not terminated yet, or into state $s_3 = ((Last_msg \parallel_S (stop.\underline{0} +^q ack.RB))/S)\backslash AType_H$ with probability p, denoting that the last message has been sent. Then, from state s_2 the system evolves either into state s_1 again if the two processes synchronize on the action of type *ack* with probability $1-q$, or into the termination state if they synchronize on the action of type *stop* with probability q. Note that from state s_3 the unfair state is reached only if the two processes synchronize on the action of type *stop* with probability q.

By verifying the *PBNDC*, we obtain $Prob(((OrigB \parallel_S H_Recip)/S)\backslash AType_H, \tau^*\mathbf{unfair}, C) = 0\ \forall C \in \mathcal{G}/\approx_{PB}$, while $Prob(((OrigB \parallel_S RecipB)/S)\backslash AType_H, \tau^*\mathbf{unfair}, C')$, where C' is the equivalence class of the terminated term $\underline{0}$, is equal to:

$$z = p \cdot q \cdot \sum_{i=0}^{\infty} ((1-p)\cdot(1-q))^i = \frac{p \cdot q}{1-(1-p)\cdot(1-q)}$$

Given $0 < p < 1$ chosen by the originator and by assuming the constraint $0 \le q \le 1$, the maximum value for z is p, obtained by taking $q = 1$. Therefore,

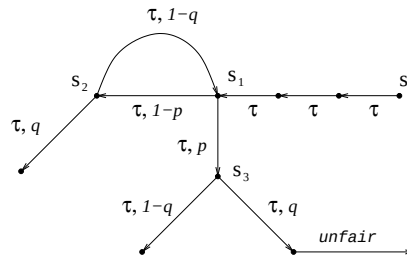

Fig. 1. *GRTS* underlying the non-repudiation protocol specification

it is easy to see that the recipient model which optimizes the probability of violating the fairness condition is obtained by assuming term RB of Table 6 to be $RB \stackrel{\Delta}{=} snd_msg_{*}.stop.\underline{0}$. In general, even if the recipient knows the parameter p of the Bernoulli distribution chosen by the originator, process $RecipB$ cannot adopt a strategy for which the probability of guessing the number of steps of the protocol is greater than p. Hence, such a system can be considered as a secure system if we can tolerate p-perturbations. In particular, independently of the behavior of the recipient, the upper bound p can be kept as low as will fit the security needs of the originator, by simply choosing adequate parameters for the sampling space size and the probability distribution.

5 Conclusion

As the literature has shown (see, e.g., [21, 6]), the role of the probabilistic aspect is important for the analysis of distributed systems. The well-known and established results in the theory of (probabilistic) process algebras allowed us to easily extend the nondeterministic, qualitative notion of non-interference to a probabilistic, quantitative definition of security. The need for such an extension has been emphasized in a simple but effective case study, a probabilistic non-repudiation protocol, whose analysis reveals the importance of quantifying the potential insecure behaviors of systems.

To the best of our knowledge, a formal quantitative estimate of probabilistic information flows is studied also in [22] in the context of a type system for secure information flow and in [9] in the framework of a declarative programming language. However, we think that the formalisms of [22] and [9] lack the expressivity needed to analyze cryptographic protocols like the one presented in this paper.

From a theoretical viewpoint, it would be useful to find a static (i.e., without universal quantification over all possible adversaries) or a stronger, easily verifiable characterization of Probabilistic $BNDC$. Indeed, while in this case study it seems reasonable to guess that M_Recip is the recipient model that optimizes the probability of cheating (but we have not proved it), in a more general setting it would be needed to characterize the most powerful adversary that can interact with the system to be analyzed. Then, the notion of probabilistic weak bisimulation which tolerates small fluctuations can be employed to verify different security properties of cryptographic protocols (not only non-repudiation, but also authentication, secrecy, integrity, and anonymity), depending on the particular observable low level actions we add to the system specification. This approach can help the modeler of complex systems to reveal unexpected, subtle information flows that otherwise are difficult to be captured because of the large number of possible behaviors.

We conclude with two observations concerning the analysis of the system specification. On the one hand, we could exchange the roles of the recipient and the originator, in order to study the behavior of a honest recipient for each potential malicious originator. As already seen, such an analysis would reveal that there does not exist a system state which is unfair from the recipient

standpoint. On the other hand, we could explicitly add a component modeling a lossy channel. In such a case, we would obtain that the probability of observing an unfair behavior cannot be limited to any ε fixed by the originator. This is because the lower bound for the unfairness condition would be represented by the percentage of lost (delayed) packets, which is a parameter that is not under the control of the originator. Our probabilistic approach can be employed to study the trade-off between the security guarantees of the cryptographic protocol and the communication channel conditions, by revealing the adaptiveness of the protocol in tolerating hostile environment conditions.

Acknowledgment

The authors thank anonymous referees for their valuable comments. This research has been funded by Progetto MURST "Metodi Formali per la Sicurezza e il Tempo" (MEFISTO).

References

1. A. Aldini, *"Probabilistic Information Flow in a Process Algebra"*, in Proc. of the 12th Int. Conf. on Concurrency Theory, LNCS 2154, pp. 152-168, Springer-Verlag, 2001
2. A. Aldini, *"On the Extension of Non-interference with Probabilities"*, 2nd ACM SIGPLAN and IFIP WG 1.7 Workshop on Issues in the Theory of Security, 2002
3. A. Aldini, M. Bravetti, *"An Asynchronous Calculus for Generative-Reactive Probabilistic Systems"*, in Proc. of the 8th Int. Workshop on Process Algebra and Performance Modeling, pp. 591-605, Carleton Scientific, 2000
4. A. Aldini, M. Bravetti, R. Gorrieri, *"A Process Algebraic Approach for the Analysis of Probabilistic Non-interference"*, Technical Report UBLCS-2002-2, University of Bologna, Italy, 2002, ftp://ftp.cs.unibo.it/pub/techreports/
5. C. Baier, H. Hermanns, *"Weak Bisimulation for Fully Probabilistic Processes"*, in Proc. of the 9th Int. Conf. on Computer Aided Verification, LNCS 1254, pp. 119-130, Springer-Verlag, 1997
6. C. Baier, M. Kwiatkowska, *"Domain Equations for Probabilistic Processes"*, in Mathematical Structures in Computer Science 10(6), pp. 665-717, 2000
7. F. van Breugel, J. Worrell, *"Towards Quantitative Verification of Probabilistic Systems (extended abstract)"*, in Proc. of the 28th International Colloquium on Automata, Languages and Programming, LNCS 2076, pp. 421-432, Springer-Verlag, 2001
8. J. Desharnais, V. Gupta, R. Jagadeesan, P. Panangaden, *"Metrics for Labeled Markov Processes"*, in Proc. of the 10th Int. Conf. on Concurrency Theory, LNCS 1664, pp. 258-273, Springer-Verlag, 1999
9. A. Di Pierro, C. Hankin, H. Wiklicky, *"On Approximate Non-Interference"*, 2nd ACM SIGPLAN and IFIP WG 1.7 Workshop on Issues in the Theory of Security, 2002
10. R. Focardi, R. Gorrieri, *"A Classification of Security Properties"*, Journal of Computer Security, 3(1):5-33, 1995

11. R. Focardi, R. Gorrieri, *"Classification of Security Properties (Part I: Information Flow)"*, Foundations of Security Analysis and Design - Tutorial Lectures (R. Focardi and R. Gorrieri, Eds.), LNCS 2171, pp. 331-396, Springer-Verlag, 2001
12. R. Focardi, R. Gorrieri, F. Martinelli, *"Non Interference for the Analysis of Cryptographic Protocols"*, in Proc. of the 27th International Colloquium on Automata, Languages and Programming, LNCS 1853, pp. 354-372, Springer-Verlag, 2000
13. R.J. van Glabbeek, S.A. Smolka, B. Steffen, *"Reactive, Generative and Stratified Models of Probabilistic Processes"*, in Information and Computation 121:59-80, 1995
14. J.A. Goguen, J. Meseguer, *"Security Policy and Security Models"*, in Proc. of Symposium on Security and Privacy, pp. 11-20, IEEE CS Press, 1982
15. J.W. Gray III, *"Toward a Mathematical Foundation for Information Flow Security"*, Journal of Computer Security, 1:255-294, 1992
16. Y. Han, *"Investigation of Non-repudiation Protocols"*, in ACISP: Information Security and Privacy: Australasian Conference, LNCS 1172, pp. 38-47, Springer-Verlag, 1996
17. O. Markowitch, Y. Roggeman, *"Probabilistic Non-Repudiation without Trusted Third Party"*, 2nd Conference on Security in Communication Networks, Amalfi, Italy, 1999
18. R. Milner, *"Communication and Concurrency"*, Prentice Hall, 1989
19. P.Y.A. Ryan, *"A CSP Formulation of Non-Interference"*, Cipher, pp. 19-27, IEEE CS Press, 1991
20. A. Sabelfeld, D. Sands, *"Probabilistic Noninterference for Multi-threaded Programs"*, in Proc. of the 13th Computer Security Foundations Workshop, IEEE CS Press, 2000
21. R. Segala, N.A. Lynch, *"Probabilistic Simulations for Probabilistic Processes"*, in Proc. of the 5th Int. Conf. on Concurrency Theory, LNCS 836, pp. 481-496, Springer-Verlag, 1994
22. G. Smith, *"Weak Probabilistic Bisimulation for Secure Information Flow"*, 2nd ACM SIGPLAN and IFIP WG 1.7 Workshop on Issues in the Theory of Security, 2002
23. J. Zhou, D. Gollmann, *"An Efficient Non-repudiation Protocol"*, in Proc. of the 10th Computer Security Foundations Workshop, pp. 126-132, IEEE CS Press, 1997

A Operational Semantics of the Calculus

The formal semantics of our probabilistic process algebra is given by the *GRTS* $(\mathcal{G}, AType, T)$, whose states are terms of the calculus and the transition relation T is the minimal relation satisfying the operational rules reported in Table 7 and in Table 8, where in addition to rules undersigned with l, which refer to the local moves of the lefthand process P, we also consider the symmetric rules (undersigned with r) taking into account the local moves of the righthand process Q, obtained by exchanging the roles of terms P and Q in the premises and by replacing p with $1-p$ in the label of the derived transitions.

As far as the notation is concerned, we use $P \xrightarrow{\pi} P'$ to stand for $\exists\, p$: $P \xrightarrow{\pi,\, p} P'$, meaning that P can execute action π with probability p and then

behaving as P'; $P \xrightarrow{\pi}$ to stand for $\exists\, P' : P \xrightarrow{\pi} P'$, meaning that P can execute action π; $P \xrightarrow{G}$ to stand for $\exists\, a \in G \subseteq AType : P \xrightarrow{a}$, meaning that P can execute a generative action of type belonging to set G.

It is easy to see that rules $(gr1)$, $(gr2)$, and $(r1)$ to $(g2)$ express exactly the informal description of the semantics of the operators given in Sect. 3.

Rules $(r3)$ to $(r5)$ are concerned with reactive transitions executable by $P \parallel_S^p Q$. In particular, rules $(r3)$ and $(r4)$ express the independent reactive transitions of P and Q (note that they follow the same probabilistic mechanism seen for the alternative choice operator). Moreover, as far as reactive actions of a given type $a \in S$ are concerned, if both P and Q may execute some reactive action a_*, the choice of the two actions a_* of P and Q forming the actions a_* executable by $P \parallel_S^p Q$ is made according to the probability they are *independently* chosen by P and Q (see rule $(r5)$).

Rules $(g3)$ to $(g6)$ are concerned with generative transitions executable by $P \parallel_S^p Q$. In particular, since we consider a restricted set of executable actions, we redistribute the probabilities of the generative transitions of P (Q) executable by $P \parallel_S^p Q$ so that their overall probability sums up to 1 (as standard when restricting actions in the generative model [13]). To this aim, in semantics rules we employ the function $\nu_P(G) : \mathcal{P}(AType) \longrightarrow]0,1]$, with $P \in \mathcal{G}$, defined as $\nu_P(G) = \sum \{\!|\, p \mid \exists P',\, a \in G : P \xrightarrow{a,\,p} P' \,|\!\}$ that computes the sum of the probabilities of the generative transitions executable by P whose type belongs to set G. Note that the set of types of the generative transitions of P executable by $P \parallel_S^p Q$ is composed by the action types not belonging to the synchronization set S and the action types belonging to S for which a synchronization between a generative action of P and a reactive action of Q is allowed to be performed. Therefore, if we take as set G in $\nu_P(G)$ the set $G_{S,Q} \subseteq AType$, with $S \subseteq AType - \{\tau\}$ and $Q \in \mathcal{G}$, to be defined as $G_{S,Q} = \{a \in AType \mid a \notin S \vee (a \in S \wedge Q \xrightarrow{a_*})\}$, we have that $\nu_P(G_{S,Q})$ computes the overall probability of the generative transitions of P executable by $P \parallel_S^p Q$ and can be used to normalize the probabilities of the generative transitions of P. Moreover, the probability of the generative transitions a (with $a \in S$) of P are further distributed among the reactive transitions of type a of Q. Finally, parameter p is used to redistribute the probabilities of the normalized generative transitions of P and Q executable by $P \parallel_S^p Q$.

As far as the restriction operator is concerned, rule $(r6)$ states that $P \backslash L$ does not inherit those reactive bundles executable by P whose type belongs to the restriction set L. Moreover, by following the same mechanism seen for the parallel operator, in rule $(g7)$ we use function $\nu_P(G_L)$, which computes the overall probability of the generative transitions of P executable by $P \backslash L$, to normalize the probabilities of the generative transitions of P. In particular, since we want G_L to be the set of action types not belonging to the restriction set L, we assume $G_L \subseteq AType$, with $L \subseteq AType - \{\tau\}$, to be defined as follows: $G_L = \{a \in AType \mid a \notin L\}$.

Table 7. Operational semantics: Part I (symmetric rules omitted)

$$(gr1) \quad \pi.P \xrightarrow{\pi,1} P$$

$$(r1_l) \quad \frac{P \xrightarrow{a_*,q} P' \quad Q \xrightarrow{a_*}}{P +^p Q \xrightarrow{a_*,p\cdot q} P'} \qquad (r2_l) \quad \frac{P \xrightarrow{a_*,q} P' \quad Q \not\xrightarrow{a_*}}{P +^p Q \xrightarrow{a_*,q} P'}$$

$$(g1_l) \quad \frac{P \xrightarrow{a,q} P' \quad Q \xrightarrow{GAct}}{P +^p Q \xrightarrow{a,p\cdot q} P'} \qquad (g2_l) \quad \frac{P \xrightarrow{a,q} P' \quad Q \not\xrightarrow{GAct}}{P +^p Q \xrightarrow{a,q} P'}$$

$$(r3_l) \quad \frac{P \xrightarrow{a_*,q} P' \quad Q \xrightarrow{a_*}}{P \parallel^p_S Q \xrightarrow{a_*,p\cdot q} P' \parallel^p_S Q} \; a \notin S \qquad (r4_l) \quad \frac{P \xrightarrow{a_*,q} P' \quad Q \not\xrightarrow{a_*}}{P \parallel^p_S Q \xrightarrow{a_*,q} P' \parallel^p_S Q} \; a \notin S$$

$$(r5) \quad \frac{P \xrightarrow{a_*,q} P' \; Q \xrightarrow{a_*,q'} Q'}{P \parallel^p_S Q \xrightarrow{a_*,q\cdot q'} P' \parallel^p_S Q'} \; a \in S$$

$$(g3_l) \quad \frac{P \xrightarrow{a,q} P' \quad Q \xrightarrow{G_{S,P}}}{P \parallel^p_S Q \xrightarrow{a,p\cdot q/\nu_P(G_{S,Q})} P' \parallel^p_S Q} \; a \notin S$$

$$(g4_l) \quad \frac{P \xrightarrow{a,q} P' \quad Q \not\xrightarrow{G_{S,P}}}{P \parallel^p_S Q \xrightarrow{a,q/\nu_P(G_{S,Q})} P' \parallel^p_S Q} \; a \notin S$$

$$(g5_l) \quad \frac{P \xrightarrow{a,q} P' \quad Q \xrightarrow{a_*,q'} Q' \quad Q \xrightarrow{G_{S,P}}}{P \parallel^p_S Q \xrightarrow{a,p\cdot q'\cdot q/\nu_P(G_{S,Q})} P' \parallel^p_S Q'} \; a \in S$$

$$(g6_l) \quad \frac{P \xrightarrow{a,q} P' \quad Q \xrightarrow{a_*,q'} Q' \quad Q \not\xrightarrow{G_{S,P}}}{P \parallel^p_S Q \xrightarrow{a,q'\cdot q/\nu_P(G_{S,Q})} P' \parallel^p_S Q'} \; a \in S$$

$$(gr2) \quad \frac{P \xrightarrow{\pi,q} P'}{A \xrightarrow{\pi,q} P'} \; \text{if } A \triangleq P$$

Table 8. Operational semantics: Part II

$$(r6)\quad \frac{P \xrightarrow{a_*,q} P'}{P\backslash L \xrightarrow{a_*,q} P'\backslash L}\ a \notin L \qquad (g7)\quad \frac{P \xrightarrow{a,q} P'}{P\backslash L \xrightarrow{a,q/\nu_P(G_L)} P'\backslash L}\ a \notin L$$

$$(r7)\quad \frac{P \xrightarrow{a_*,q} P' \quad P \xrightarrow{GAct}}{P/^p_a \xrightarrow{\tau,p\cdot q} P'/^p_a} \qquad (r8)\quad \frac{P \xrightarrow{a_*,q} P' \quad P \not\xrightarrow{GAct}}{P/^p_a \xrightarrow{\tau,q} P'/^p_a}$$

$$(r9)\quad \frac{P \xrightarrow{b_*,q} P'}{P/^p_a \xrightarrow{b_*,q} P'/^p_a}\ a \neq b$$

$$(g8)\quad \frac{P \xrightarrow{b,q} P' \quad P \xrightarrow{a_*}}{P/^p_a \xrightarrow{b,(1-p)\cdot q} P'/^p_a}\ a \neq b \qquad (g9)\quad \frac{P \xrightarrow{b,q} P' \quad P \not\xrightarrow{a_*}}{P/^p_a \xrightarrow{b,q} P'/^p_a}\ a \neq b$$

$$(g10)\quad \frac{P \xrightarrow{a,q} P' \quad P \xrightarrow{a_*}}{P/^p_a \xrightarrow{\tau,(1-p)\cdot q} P'/^p_a} \qquad (g11)\quad \frac{P \xrightarrow{a,q} P' \quad P \not\xrightarrow{a_*}}{P/^p_a \xrightarrow{\tau,q} P'/^p_a}$$

Finally, as far as the hiding operator is concerned, the generative bundle executable by $P/^p_a$ is obtained from the generative bundle and the reactive bundle of type a executable by P. In particular, we distinguish the following cases:

- P enables both some reactive action a_* and some generative action. In such a case, in order to form the generative bundle of $P/^p_a$, the probabilities of the transitions a_* of P are multiplied by p when turned into the generative transitions τ (rule $(r7)$), while the probabilities of the generative transitions of P are multiplied by $1-p$ and transitions a are turned into transitions τ (rules $(g8)$ and $(g10)$);
- P does not enable generative actions, therefore transitions a_* of P are simply turned into generative transitions τ and parameter p is not used (rule $(r8)$);
- P does not enable reactive actions a_*, therefore the probability distribution of the generative transitions does not change, transitions a are turned into transitions τ, and parameter p is not used (rules $(g9)$ and $(g11)$);
- for each $b \in AType$ such that $b \neq a$, $P/^p_a$ simply inherits the reactive bundle of type b executable by P (rule $(r9)$).