Intransitive Non-Interference for Cryptographic Purposes

Michael Backes
IBM Zurich Research Laboratory
Rüschlikon, Switzerland
mbc@zurich.ibm.com

Birgit Pfitzmann
IBM Zurich Research Laboratory
Rüschlikon, Switzerland
bpf@zurich.ibm.com

Abstract

Information flow and non-interference have recently become very popular concepts for expressing both integrity and privacy properties. Because of the enormous potential of transmitting information using probabilistic methods of cryptography, interest arose in capturing probabilistic non-interference. We investigate the notion of intransitive probabilistic non-interference in reactive systems, i.e., downgrading of probabilistic information and detection of probabilistic information flow by one or more involved third parties. Based on concrete examples, we derive several definitions that comprise cryptography-related details like error probabilities and computational restrictions. This makes the definitions applicable to systems involving real cryptography. Detection of probabilistic information flow is significantly more complicated to define if several third parties are involved because of the possibilities of secret sharing. We solve this problem by graph-theoretic techniques.

1 Introduction

Information flow and in particular non-interference have become powerful possibilities for expressing both privacy and integrity requirements. Mainly, definitions for non-interference can be categorized by two aspects. The first aspect is the complexity-theoretic background addressed, i.e., whether we consider *non-deterministic* behavior (also called possibilistic), or the more fine-grained *probabilistic* flow of information. The second aspect is the particular point of view of what should be regarded as non-interference. Early definitions of non-interference were always based on transitive flow policies, i.e., they follow the intuition that if a user shall not be able to influence another user directly, it shall also not be able to influence it by involving additional users, called third parties. However, the use of such policies has been quite limited in practice, as important concepts like information filters, channel control, or explicit downgrading cannot be captured. Therefore the notion of intransitive non-interference arose to deal with these issues.

We present the first definitions for intransitive probabilistic non-interference. Our definitions are very general in several ways: They are designed for reactive scenarios. We do not only consider perfect non-interference as in Gray's commonly accepted definition of probabilistic non-interference (which is restricted to transitive flow policies), but we further allow error probabilities. Further, our definitions comprise complexity-theoretic reasoning like polynomially bounded adversaries. Because of the last two points, we are confident that this work is a major step in relating cryptography to the notion of information flow.

Compared with prior definitions handling intransitive flow policies (for a non-probabilistic definition of flow), probabilistic behaviors are much more difficult to capture because two pieces carrying absolutely no information about a secret in the probabilistic sense, might reveal the entire secret when joint. This causes severe problems if multiple third parties are involved as the secret might be sent via different parties using this concept of secret sharing. Dealing with aspects of this kind is the topic of this work.

Outline. We start with a brief overview of the underlying model of asynchronous reactive systems, and introduce flow policies as the formal concept of expressing information flow, along with motivation for intransitive flow policies (Section 2). Our main contributions are novel definitions, which can cope with intransitive policies for a probabilistic definition of information flow, even in the presence of cryptographic techniques (Section 3). Relating an intransitive notion of flow and cryp-

Proceedings of the 2003 IEEE Symposium on Security and Privacy (SP'03)
1081-6011/03 $17.00 © 2003 IEEE

tography (even more generally, probabilism) was out of scope of prior approaches. As the most common application of intransitive flow policies, we show in Section 4 how probabilistic downgrading can be captured using our ideas of the previous section. Moreover, we show that our definitions are preserved under simulatability, which is a common concept in cryptography (Section 5). This significantly simplifies the proof that a system fulfills a non-interference property, since simulatability helps to get rid of cryptography-related details like error probabilities and computational restrictions. We show this for an example in Section 6. We conclude this article by discussing related work (Section 7) and summarizing our results (Section 8).

2 Preliminary Definitions and Ideas

In this section, we give an introduction to information flow in general, before we turn our attention to intransitive flow in subsequent sections. In Section 2.1 we briefly introduce the model for asynchronous reactive systems on which we base our definitions of non-interference. In Section 2.2 we introduce flow policies and briefly review the definition of computational probabilistic non-interference for transitive flow policies [2], which serves as the foundation of our upcoming definitions. The problem of probabilistic non-interference with intransitive flow policies is addressed in Section 2.3.

2.1 General System Model for Reactive Systems

In this section we briefly recapitulate the model for probabilistic reactive systems in asynchronous networks, including computational aspects as needed for cryptography, from [21]. All details of the model which are not necessary for understanding are omitted; they can be looked up in the original paper.

Usually one considers real systems consisting of a set $\hat{M}$ of machines $\{\mathsf{M}_1, \ldots, \mathsf{M}_n\}$, one for each user. The machine model is probabilistic state-transition machines, similar to I/O automata as introduced in [12]. For complexity every automaton is considered to be implemented as a probabilistic Turing machine; complexity is measured in the length of its initial state, i.e., the initial worktape content (often a security parameter k, given in unary representation).

Communication between different machines is done via ports. Similar to the CSP-Notation [8], output and input ports are written as $\mathsf{p}!$ and $\mathsf{p}?$, respectively. The ports of a machine M are denoted by $\mathsf{ports}(\mathsf{M})$. Connections are defined implicitly by naming convention, i.e., port $\mathsf{p}!$ sends messages to $\mathsf{p}?$. To achieve asynchronous timing, a message is not sent directly to its recipient, but first stored in a special machine $\widetilde{\mathsf{p}}$ called a buffer and waits to be scheduled. If a machine wants to schedule the i-th message held in $\widetilde{\mathsf{p}}$, it must have the corresponding clock-out port $\mathsf{p}^{\triangleleft}!$, and it sends i at $\mathsf{p}^{\triangleleft}!$. The i-th message is then forwarded by the buffer and removed from the buffer's internal list. Most buffers are either scheduled by a specific master scheduler or the adversary, i.e., one of those has the corresponding clock-out port.

Formally, a *structure* is a pair $(\hat{M}, S)$, where $\hat{M}$ is a finite set of machines with pairwise different machine names and disjoint sets of ports, and $S \subseteq \mathsf{free}(\hat{M})$, the so-called *specified ports*, are a subset of the free ports of $\hat{M}$.[1] Roughly speaking the ports of S guarantee certain services to the users. A structure is completed to a *configuration* by adding a set of machines U and a machine A, modeling users and the adversary. The machines in U connect to the specified ports S, while A connects to the remaining free ports $\bar{S}$ of the structure and can interact with the honest users.

Scheduling of machines is done sequentially, so there is exactly one active machine M at any time. If this machine has clock-out ports, it can select the next message to be scheduled as explained above. If that message exists, it is delivered by the buffer and the unique receiving machine is the next active machine. If M tries to schedule multiple messages, only one is taken, and if it schedules none or the message does not exist, the special master scheduler is scheduled.

This means that a configuration has a well-defined notion of *runs*, also called *traces* or executions. Formally a run is essentially a sequence of *steps*, and each step is a tuple of the name of the active machine in this step and its input, output, and old and new local state. As the underlying state-transition functions of the individual machines are probabilistic, we also get a probability space on the possible runs. We call it $run_{conf,k}$ for a configuration $conf$ and the security parameter k.

One can restrict a run r to a machine M or a set of machines $\hat{M}$ by retaining only the steps of these machines; this is called the *view* of these machines. Similarly, one can restrict a run to a set $\mathcal{S}$ of ports by retaining only the in- or outputs at the chosen ports from the steps where such in- or outputs occur. This is denoted by $r\lceil_{\mathsf{M}}$ and $r\lceil_{\hat{M}}$ and $r\lceil_{\mathcal{S}}$, respectively. For a configuration $conf$, we obtain corresponding random variables over the probability space of all possible runs; for the

[1] A port is *free* if its corresponding port is not in the system. These ports are available for the users and the adversary. By $\mathsf{free}(\hat{M})$ we abbreviate the precise notation $\mathsf{free}([\hat{M}])$ from [21], which indicates that the free ports are taken at the other end of the buffers.

view of a machine or machine set they are denoted by $view_{conf,k}(\mathsf{M})$ and $view_{conf,k}(\hat{M})$, respectively.

2.2 Flow Policies

Flow policies specify restrictions on the information flow within a system. They presuppose the existence of security domains $\mathcal{S}$, between which information flow should either be permissible or forbidden.

Definition 2.1 *(General Flow Policy)* A *general flow policy* is a graph $\mathcal{G} = (\mathcal{S}, \mathcal{E})$ with a non-empty set $\mathcal{S}$ and $\mathcal{E} \subseteq \mathcal{S} \times \mathcal{S}$. For $(s_1, s_2) \in \mathcal{E}$, we write $s_1 \rightsquigarrow s_2$, and $s_1 \not\rightsquigarrow s_2$ otherwise. Furthermore we demand $s \rightsquigarrow s$ for all $s \in \mathcal{S}$. $\diamond$

Here $s_1 \rightsquigarrow s_2$ means that information may flow from s_1 to s_2, whereas $s_1 \not\rightsquigarrow s_2$ means that it must not. If we want to use a general flow policy for our purpose, we have to refine it so that it can be applied to a structure $(\hat{M}, S)$ of the underlying model. The intuition is to define a graph on the possible protocol participants, i.e., the users and the adversary. However, to be independent of the details of the actual user and adversary machines, we represent users by the ports they connect to in the structure $(\hat{M}, S)$, and the adversary by the remaining free ports of the structure. Thus our flow policy only depends on the specified ports S.

Definition 2.2 *(Flow Policy)* Let a structure $(\hat{M}, S)$ be given, and let $\Gamma_{(\hat{M},S)} = \{S_i \mid i \in \mathcal{I}\}$ denote a partition of S for a finite index set $\mathcal{I}$. Hence $\Delta_{(\hat{M},S)} := \Gamma_{(\hat{M},S)} \cup \{\bar{S}\}$ is a partition of $\mathsf{free}(\hat{M})$. A *flow policy* $\mathcal{G}_{(\hat{M},S)}$ of the structure $(\hat{M}, S)$ is now defined as a general flow policy $\mathcal{G}_{(\hat{M},S)} = (\Delta_{(\hat{M},S)}, \mathcal{E}_{(\hat{M},S)})$. $\diamond$

We write $\mathcal{G}$, Δ, and $\mathcal{E}$ instead of $\mathcal{G}_{(\hat{M},S)}$, $\Delta_{(\hat{M},S)}$, and $\mathcal{E}_{(\hat{M},S)}$ if the underlying structure is clear from the context.

The relation $\not\rightsquigarrow$ is the non-interference relation of $\mathcal{G}$. Hence $S_\mathsf{H} \not\rightsquigarrow S_\mathsf{L}$ for two port sets $S_\mathsf{H}, S_\mathsf{L} \in \Delta$ means that no information must flow from the user connected to the ports S_H to the user connected to the ports S_L. For transitive flow policies, we recently introduced a probabilistic definition suited for cryptographic purposes (i.e., comprising computational restrictions, error probabilities, etc.) in [2]. Roughly, if we consider a non-interference requirement $S_\mathsf{H} \not\rightsquigarrow S_\mathsf{L}$, then the user H (which is connected to S_H by naming convention, using the notation of [2]) gets a randomly distributed bit b at the start of the run and should try to transmit this bit to L (connected to S_L). The user L then outputs a bit b^*, its guess of the bit b. To model this, the distinguished users have special ports for receiving the initial bit and for outputting their guess, respectively. Formally, to close the configuration, special machines are added that produce the bit b and consume the bit b^* at ports $\mathsf{p}_{\mathsf{H\_bit}}!$ and $\mathsf{p}^*_{\mathsf{L\_bit}}?$, respectively, connected to the special ports of the users H and L. Moreover, a specific fair master scheduler is added to the configuration because if the adversary were allowed to schedule it could always achieve probabilistic information flow. The resulting configurations are called *non-interference configurations* for S_H and S_L and typically denoted by $conf^{n\_in}_{\mathsf{H},\mathsf{L}}$. We call the set of them $\mathsf{Conf}^{n\_in}_{\mathsf{H},\mathsf{L},\mathcal{I}}(\hat{M}, S)$.[2]

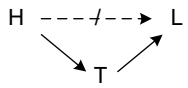

Figure 1. Standard intransitive flow policy, consisting of three users H, L and T

Then the underlying structure is defined to fulfill the non-interference requirement $S_\mathsf{H} \not\rightsquigarrow S_\mathsf{L}$ in the computational sense iff for all non-interference configurations for S_H and S_L, the probability of a correct guess $b = b^*$ is only negligibly greater than pure guessing; see [2] for a rigorous definition, also of a perfect and a statistical case. For readability, we identify users and the specified ports they connect to, i.e., we might write $\mathsf{H} \not\rightsquigarrow \mathsf{L}$ instead of $S_\mathsf{H} \not\rightsquigarrow S_\mathsf{L}$ etc. in the following.

2.3 The Problem with Intransitive Flow Policies

For intransitive flow policies, however, the definition sketched in Section 2.2 is meaningless. Consider the flow policy shown in Figure 1, which can be seen as the standard flow policy for intransitive information flow.

Obviously, the non-interference relation $\mathsf{H} \not\rightsquigarrow \mathsf{L}$ cannot be achieved according to the above definition if information flow from H to L is possible via T. Although these kinds of policies do not match our intuition of a satisfiable flow policy at first glance, they can be interpreted as *conditional* information flow capturing lots of typical situations in real life. For instance consider a user that wants to send certain documents to the printer. Assume that the system administrator has set up some nice predefined functions for printing documents, augmenting them with special company-related frames, or some internal handling of possible errors. Then a typical intransitive flow policy could say that information flow

[2] Here and in some further places we change some notation of [21, 2] from so-called systems to structures. These systems contain several possible structures, derived from an intended structure with a trust model. Here we can always work with individual structures.

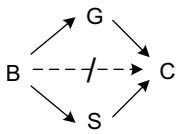

Figure 2. Flow policy for the secretary example. All missing edges are of the form $\rightsquigarrow$, i.e., flow is allowed.

from the user to the printer should not be possible unless these special services have been used, so some condition on information flow is imposed here. In the next section, we examine several further examples with the goal of investigating arbitrary probabilistic behaviors. The examples motivate different points of view of intransitive probabilistic non-interference, resulting in different possible definitions.

Naming convention. Throughout the following, we consider users H and L such that $\mathsf{H} \not\rightsquigarrow \mathsf{L}$ should hold unless explicitly state otherwise. The remaining users are usually called third parties and named $\mathsf{T}_1, \ldots, \mathsf{T}_n$ or simply T if there is only one.

3 Examples and Definitions

In this section, we investigate what probabilistic non-interference means for intransitive flow policies, resulting in several possible definitions, each one motivated by a concrete example.

3.1 Example: Secretary

Consider a small company selling goods or services over the Internet. Ordinarily, its CEO (chief executive officer) C does not want to be disturbed by customers, and thus all normal correspondence is handled by a secretary S. However, some specific good customers are allowed to contact the CEO directly; we denote the others by B for "bad" customers. This corresponds to the information flow graph of Figure 2 for one good customer G and one bad customer B. This is a typical example where indirect information flow from B to C via the secretary S is certainly allowed. Further, "bad" customers that know a good customer can also bypass the secretary by having the good customer vouch for them. For digital communication, we may want to enforce this flow policy by cryptographic authentication and filtering, see Section 6.

3.2 Blocking Non-Interference

How do we express this notion of intransitive information flow? According to our intuition, we would like to model that B can only influence C if B has help from either S or G. Equivalently, we can define that for all C and all B, there exists an S and a G such that the considered information flow is prohibited in the resulting configuration. This means that the secretary might refuse to put the customer through to C and G might refuse to help. This yields our first definition of intransitive non-interference, which we call *blocking non-interference*. We first make it for precisely this flow policy, and generalize it below.

Definition 3.1 *(Blocking Non-Interference for One Policy)* Let the flow policy $\mathcal{G} = (\Delta, \mathcal{E})$ of Figure 2 for a structure $(\hat{M}, S)$ be given. We say that $(\hat{M}, S)$ fulfills the blocking non-interference requirement

a) **perfectly** (written $(\hat{M}, S) \models^{\mathsf{block\_intrans}}_{\mathsf{perf}} \mathcal{G}$) iff for all B, C there exists S, G such that for all non-interference configurations $conf^{n\_in}_{\mathsf{B},\mathsf{C}} = (\hat{M}, S, \{\mathsf{B}, \mathsf{C}, \mathsf{S}, \mathsf{G}\}, \mathsf{A}) \in \mathsf{Conf}^{n\_in}_{\mathsf{B},\mathsf{C},\mathcal{I}}(\hat{M}, S)$ of this structure the inequality

$$P(b = b^* \quad | \quad r \leftarrow run_{conf^{n\_in}_{\mathsf{B},\mathsf{C}},k}; \; b := r\lceil_{\mathsf{p\_B\_bit}}!; \; b^* := r\lceil_{\mathsf{p}^*\_\mathsf{C\_bit}}?) \leq \frac{1}{2}$$

holds. (Typically it is then $= \frac{1}{2}$, but C might destroy its guessing chances by not outputting any Boolean value.)

b) **statistically** for a class $SMALL$ of functions $((\hat{M}, S) \models^{\mathsf{block\_intrans}}_{SMALL} \mathcal{G})$ iff for all B, C there exists S, G such that for all non-interference configurations $conf^{n\_in}_{\mathsf{B},\mathsf{C}} = (\hat{M}, S, \{\mathsf{B}, \mathsf{C}, \mathsf{S}, \mathsf{G}\}, \mathsf{A}) \in \mathsf{Conf}^{n\_in}_{\mathsf{B},\mathsf{C},\mathcal{I}}(\hat{M}, S)$ of this structure the inequality

$$P(b = b^* \quad | \quad r \leftarrow run_{conf^{n\_in}_{\mathsf{B},\mathsf{C}},k}; \; b := r\lceil_{\mathsf{p\_B\_bit}}!; \; b^* := r\lceil_{\mathsf{p}^*\_\mathsf{C\_bit}}?) \leq \frac{1}{2} + s(k)$$

holds for some $s \in SMALL$. $SMALL$ must be closed under addition and with a function g also contain every function $g' \leq g$.

c) **computationally** $((\hat{M}, S) \models^{\mathsf{block\_intrans}}_{\mathsf{poly}} \mathcal{G})$ iff for all polynomial-time B, C there exists polynomial-time S, G such that for all polynomial-time non-interference configurations $conf^{n\_in}_{\mathsf{B},\mathsf{C}} = (\hat{M}, S,$

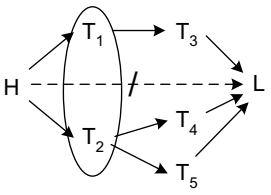

Figure 3. Flow policy with five third parties. All missing edges are the form $\not\rightsquigarrow$, i.e., flow is not allowed. Users T_1 and T_2 form a cut for H and L.

$\{\mathsf{B},\mathsf{C},\mathsf{S},\mathsf{G}\},\mathsf{A}) \in \mathsf{Conf}^{n\_in}_{\mathsf{B},\mathsf{C},\mathcal{I}}(\hat{M},S)$ of this structure the inequality

$$P(b = b^* \quad | \quad r \leftarrow run_{conf^{n\_in}_{\mathsf{B},\mathsf{C}},k};\; b := r\lceil_{\mathsf{p}_{\mathsf{B}\_\mathsf{bit}}!};\; b^* := r\lceil_{\mathsf{p}^*_{\mathsf{C}\_\mathsf{bit}}?}) \leq \frac{1}{2} + s(k)$$

holds for some $s \in NEGL$, the set of all negligible functions.[3]

We write "$\models^{\mathsf{block\_intrans}}$" if we want to treat all cases together. ◇

In the upcoming definitions, we will only define statistical fulfillment, as perfect fulfillment is comprised by the class $SMALL = \{0\}$. Similarly, computational fulfillment is statistical fulfillment for the class $SMALL = NEGL$ with the further restriction to polynomial-time configurations.

More generally, we can consider n third parties $\mathsf{T}_1,\ldots,\mathsf{T}_n$ instead of S and G. The obvious extension of our above definition to this case would be to replace the statement "there exists S,G" with "there exists $\mathsf{T}_1,\ldots,\mathsf{T}_n$" without any further work. Although this yields a meaningful definition, we can significantly strengthen it as follows: We no longer demand that *all* users should try to prohibit information flow between H and L, because this is fairly unrealistic. Instead we demand that certain subsets of the third parties are successful in interrupting the connection. Obviously, this cannot work for arbitrary subsets; we need that for each path from H to L in the flow graph, at least one node of the path is contained in this subset. According to graph theory, we then call the subset a *cut* for H and L of the given flow graph. An example is shown in Figure 3.

Definition 3.2 *(Cut)* Let a flow graph $\mathcal{G} = (\Delta,\mathcal{E})$ be given. Then a *cut* for two nodes $\mathsf{M}_1,\mathsf{M}_n \in \Delta$ is a set $\hat{C} \subseteq \Delta$ of nodes that cut all paths from M_1 to M_n. I.e., let $\mathcal{G}_{\hat{C}} := (\Delta_{\hat{C}},\mathcal{E}_{\hat{C}})$ with $\Delta_{\hat{C}} := \Delta \setminus \hat{C}$ and $\mathcal{E}_{\hat{C}} := \{(\mathsf{M},\mathsf{M}') \mid (\mathsf{M},\mathsf{M}') \in \mathcal{E} \wedge \mathsf{M},\mathsf{M}' \notin \hat{C}\}$. Then M_1 and M_n should lie in unconnected components of $\mathcal{G}_{\hat{C}}$, i.e., there should be no sequence $\mathsf{M}_2,\ldots,\mathsf{M}_{n-1}$ such that $(\mathsf{M}_i,\mathsf{M}_{i+1}) \in \mathcal{E}_{\hat{C}}$ for $j = 1,\ldots,n-1$. ◇

[3] We have $s \in NEGL$ iff for all positive polynomials $Q\ \exists n_0\ \forall n > n_0 : s(n) < \frac{1}{Q(n)}$.

Using cuts, we can now give a general definition of blocking non-interference (using the general convention of $\mathsf{H} \not\rightsquigarrow \mathsf{L}$ again). It states that whatever users $\mathsf{T}_{i_1},\ldots,\mathsf{T}_{i_l}$ might do to "help" H to transmit the bit, the remaining users $\mathsf{T}_{j_1},\ldots,\mathsf{T}_{j_t}$ can still prohibit information flow, provided that they are a cut for H and L.

Definition 3.3 *(Blocking Non-Interference)* Let a flow policy $\mathcal{G} = (\Delta,\mathcal{E})$ for a structure $(\hat{M},S)$ be given, consisting of H, L, and third parties $\mathsf{T}_1,\ldots,\mathsf{T}_n$. We say that $(\hat{M},S)$ fulfills the blocking non-interference requirement ($(\hat{M},S) \models^{\mathsf{block\_intrans}}_{SMALL} \mathcal{G}$) iff for all H,L and for all cuts $\hat{C}$ for H and L, there exist users $\{\mathsf{T}_{j_1},\ldots,\mathsf{T}_{j_t}\} := \hat{C}$ such that for all non-interference configurations $conf^{n\_in}_{\mathsf{H},\mathsf{L}} = (\hat{M},S,\{\mathsf{H},\mathsf{L},\mathsf{T}_1,\ldots,\mathsf{T}_n\},\mathsf{A}) \in \mathsf{Conf}^{n\_in}_{\mathsf{H},\mathsf{L},\mathcal{I}}(\hat{M},S)$ of this structure the inequality

$$P(b = b^* \quad | \quad r \leftarrow run_{conf^{n\_in}_{\mathsf{H},\mathsf{L}},k};\; b := r\lceil_{\mathsf{p}_{\mathsf{H}\_\mathsf{bit}}!};\; b^* := r\lceil_{\mathsf{p}^*_{\mathsf{L}\_\mathsf{bit}}?}) \leq \frac{1}{2} + s(k)$$

holds for a function $s \in SMALL$. ◇

Unfortunately, our above definition is too coarse-grained to capture the full range of probabilistic behaviors. It mainly states that the secretary can interrupt the entire connection, while in reality the secretary should be able to see the content of an attempted information flow and judge whether the CEO will want to see it or not. We will deal with that next.

3.3 Example: Firewall

As another well-known example, consider a firewall guarding some honest users from malicious adversaries. Here the firewall should prohibit any negative influence from outside, so the firewall itself has to detect when a specific information flows to the user. Moreover, probabilism shows up if we consider cryptographic firewalls, i.e., firewalls whose filtering functions are based on cryptographic authentication. Here the firewall maintains a set of "allowed" users and checks each incoming message for a signature belonging to that specific set. All other messages are discarded. This motivates a new definition of non-interference, which is based on the notion of recognizing when information flow occurs.

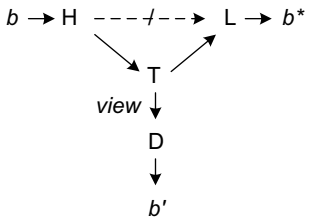

Figure 4. Sketch of the definition of recognition non-interference for the flow policy of Figure 1. The bits b^* and b' are guesses at b.

3.4 Recognition Non-Interference

Similar to the previous example we first develop our ideas for a simple policy, the one with only three users of Figure 1. Intuitively, our definition states that if there is information flow from H to L, then the user T can also recognize this flow if it wants to, i.e., it could also guess the initial bit b with non-negligible advantage. We call this notion *recognition non-interference*. It is shown in Figure 4.

The notion that an arbitrary given machine T "could" guess a bit is formalized by a machine D, called a *distinguisher*, which outputs the bit based only on the information the third party T got during the run, i.e., the view of T. More formally, a machine D is called a distinguisher if it only performs one single (probabilistic) transition resulting in precisely one non-empty output that only contains one bit. As the view of T in a run r is written $r\lceil_{\mathsf{T}}$, this probabilistic assignment of the bit b' is written $b' \leftarrow \mathsf{D}(r\lceil_{\mathsf{T}}, 1^k)$. The second input 1^k represents the security parameter k; it is given in unary representation because polynomial-time is measured in this security parameter.

Definition 3.4 *(Recognition Non-Interference for One Policy)* Let the standard flow policy of Figure 1 for a structure $(\hat{M}, S)$ be given. Then $(\hat{M}, S)$ fulfills the recognition non-interference requirement for this policy iff for all non-interference configurations $conf^{n\text{-}in}_{\mathsf{H},\mathsf{L}} = (\hat{M}, S, \{\mathsf{H}, \mathsf{L}, \mathsf{T}\}, \mathsf{A}) \in \mathsf{Conf}^{n\text{-}in}_{\mathsf{H},\mathsf{L},\mathcal{I}}(\hat{M}, S)$ the following holds: If

$$P(b = b^* \quad | \quad r \leftarrow run_{conf^{n\text{-}in}_{\mathsf{H},\mathsf{L}},k}; \; b := r\lceil_{\mathsf{p\_H\_bit}!}; \; b^* := r\lceil_{\mathsf{p}^*_{\mathsf{L\_bit}?}}) \geq \frac{1}{2} + ns(k)$$

holds for $ns(k) \notin SMALL$, then there exists a distinguisher D and a function $ns' \notin SMALL$ such that

$$P(b = b' \quad | \quad r \leftarrow run_{conf^{n\text{-}in}_{\mathsf{H},\mathsf{L}},k}; \; b := r\lceil_{\mathsf{p\_H\_bit}!}; \; b' \leftarrow \mathsf{D}(r\lceil_{\mathsf{T}}, 1^k)) \geq \frac{1}{2} + ns'(k).$$

For the computational case, the distinguisher has to be polynomial-time. ◇

We can extend our definition with arbitrary predicates $\mathsf{pred}(\cdot,\cdot)$ on the functions ns and ns' to model concrete complexity. For instance, fulfillment for the predicate $\mathsf{pred}(ns, ns') := (ns \leq ns')$ ensures that the advantage of the distinguisher is at least as good as the advantage of the user L.

The above definition works fine as long as we only consider *one* possible path from H to L where information flow is allowed to occur. Consider our secretary example of the previous section where B has two possible ways of sending information to C. Assume that it wants to transmit a codeword $m \in \Sigma^*$ (e.g., the bit b). Instead of sending the codeword in cleartext over one of these paths, B divides it into two parts such that none of them gives *any* information on the codeword on its own. The standard construction is to choose a random string m^* of the same length as m and to send $m \oplus m^*$ via S and m^* via G. If C receives both messages, she can easily reconstruct the original codeword by computing $(m \oplus m^*) \oplus m^*$, but neither S nor G can notice that the codeword has been sent. This principle can be extended to arbitrary users and trust models by the common concept of secret sharing [26].

As we can see, a probabilistic definition of information is very fine-grained, and cryptographic primitives like secret sharing even ensure that pieces carrying no information at all may be combined to provide the full amount of secret knowledge.

In our example, the joint view of S and G is sufficient to determine the desired information. More generally, the solution for arbitrary flow graphs is again to consider a cut in the flow graph, cf. Definition 3.2. The joint view of the machines contained in the cut should be sufficient to determine whether or not the specific information has been sent. This yields a more general definition of recognition non-interference, based on n third parties $\mathsf{T}_1, \ldots, \mathsf{T}_n$.

Definition 3.5 *(Recognition Non-Interference)* Let a flow policy $\mathcal{G} = (\Delta, \mathcal{E})$ for a structure $(\hat{M}, S)$ be given. Then $(\hat{M}, S)$ fulfills this policy $\mathcal{G}$ iff for all H, L and for all non-interference configurations $conf^{n\text{-}in}_{\mathsf{H},\mathsf{L}} = (\hat{M}, S, \{\mathsf{H}, \mathsf{L}, \mathsf{T}_1, \ldots, \mathsf{T}_n\}, \mathsf{A}) \in \mathsf{Conf}^{n\text{-}in}_{\mathsf{H},\mathsf{L},\mathcal{I}}(\hat{M}, S)$ the follow-

ing holds: If

$$P(b = b^* \quad | \quad r \leftarrow run_{conf^{n\_in}_{\mathsf{H},\mathsf{L}},k}; \; b := r\lceil_{\mathsf{p\_H\_bit}!}; \; b^* := r\lceil_{\mathsf{p}^*_{\mathsf{L\_bit}?}}) \geq \frac{1}{2} + ns(k)$$

holds for $ns \notin SMALL$, then for all cuts $\hat{C}$ for H, L, there exists a distinguisher D and a function $ns' \notin SMALL$ such that

$$P(b = b' \quad | \quad r \leftarrow run_{conf^{n\_in}_{\mathsf{H},\mathsf{L}},k}; \; b := r\lceil_{\mathsf{p\_H\_bit}!}; \; b' \leftarrow \mathsf{D}(r\lceil_{\hat{C}}, 1^k)) \geq \frac{1}{2} + ns'(k).$$

For the computational case, the distinguisher has to be polynomial-time. ◇

The definition can be extended to predicates on ns, ns' like Definition 3.4.

3.5 Example: Dealing with Public-Key Operations

Definition 3.5 is very general. However, if we look closely, we see that it is too strict for many cases. Recall our example of the secretary. Assume that the machine of the CEO is guarded by a cryptographic firewall that only allows messages from the secretary and the good customers to pass. However, if the CEO is malicious and wants to receive information from outside that the secretary cannot recognize, she can do so by creating a public key of an asymmetric encryption system and broadcast it to the bad customers. It does not matter whether or not the secretary and the good customers notice the transmission of this key. Now the bad customer simply encrypts its bit b and sends it to the CEO via the secretary. The secretary cannot guess the bit b with non-negligible probability using this information, or he could break the underlying encryption scheme. Thus this implementation with a cryptographic firewall does not fulfill Definition 3.5.

The problem here is that the knowledge of the secretary and the CEO is not shared, i.e., only the CEO has the secret key to decrypt incoming messages.

The problem can be solved by either prohibiting that the CEO outputs messages to outside users, which seems fairly unrealistic, or by letting the secretary know the secrets of the CEO. The second possibility is quite realistic for this kind of example, as the CEO probably does not want to be influenced by bad customers, and hence shares a lot of knowledge with her secretary already by letting him screen her mail. Such a CEO that wants to be protected will surely not create a public encryption key on her own and spread it over the net. We model this by additionally giving the distinguisher the content of the random tape of C, where C is considered as a probabilistic Turing machine. We can thus keep the distinguisher non-interactive. This is illustrated in Figure 5.

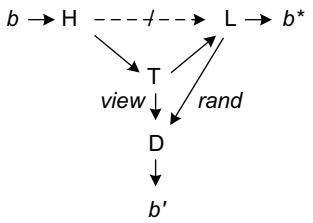

Figure 5. Sketch of recognition non-interference for trusted recipients for the flow policy of Figure 4.

3.6 Recognition Non-Interference for Trusted Recipients

This example yields a less strict definition of recognition non-interference, which we call *recognition non-interference for trusted recipients*. In the following, we denote the content of the random tape of a probabilistic Turing machine M in a run r by $\mathsf{random}_r(\mathsf{M})$, and similarly for sets.

For arbitrary policies things again become a bit more complicated. If we want to prove $\mathsf{H} \not\rightsquigarrow \mathsf{L}$, and if we have a cut $\hat{C}$ with respect to H and L, then we need the random tapes of all users "between" the cut and L. More formally, this means that for a given cut $\hat{C}$ in a flow policy, the random tape of a user T_i is needed iff T_i and L lie in the same component of the graph after removing the cut from the graph. We denote the set of these users by $T_{\hat{C}}$.

Definition 3.6 *(Recognition Non-Interference for Trusted Recipients)* Let a flow policy $\mathcal{G} = (\Delta, \mathcal{E})$ for a structure $(\hat{M}, S)$ be given. Then $(\hat{M}, S)$ fulfills the recognition non-interference requirement for trusted recipients for this policy $\mathcal{G}$ iff for all H, L and for all non-interference configurations $conf^{n\_in}_{\mathsf{H},\mathsf{L}} = (\hat{M}, S, \{\mathsf{H}, \mathsf{L}, \mathsf{T}_1, \ldots, \mathsf{T}_n\}, \mathsf{A}) \in \mathsf{Conf}^{n\_in}_{\mathsf{H},\mathsf{L},\mathcal{I}}(\hat{M}, S)$ the following holds: If

$$P(b = b^* \quad | \quad r \leftarrow run_{conf^{n\_in}_{\mathsf{H},\mathsf{L}},k}; \; b := r\lceil_{\mathsf{p\_H\_bit}!}; \; b^* := r\lceil_{\mathsf{p}^*_{\mathsf{L\_bit}?}}) \geq \frac{1}{2} + ns(k)$$

holds for $ns \notin SMALL$, then for all cuts $\hat{C}$ for H, L, there exists a distinguisher D and a function $ns' \notin SMALL$ such that

$$P(b = b' \quad | \quad r \leftarrow run_{conf^{n\text{-}in}_{\mathsf{H},\mathsf{L}},k}; \; b := r\lceil_{\mathsf{p\_H\_bit}}!; \; b' \leftarrow \mathsf{D}(r\lceil_{\hat{C}}, \mathsf{random}_r(T_{\hat{C}}), 1^k)) \geq \frac{1}{2} + ns'(k).$$

For the computational case, the distinguisher has to be polynomial-time. ◇

The definition can be extended to predicates on ns, ns' like Definition 3.4.

4 Downgrading of Information

Requiring that *no* information flow at all shall occur is too strict for many practical purposes. As a typical example consider the model of Bell-LaPadula, consisting of several groups of users along with a total relation $\geq$ on the groups. Here $\mathsf{H}_1 \geq \mathsf{H}_2$ for two groups H_1 and H_2 means that no information should flow from H_1 to H_2. This corresponds to a flow graph with $\mathsf{H}_i \not\rightsquigarrow \mathsf{H}_j$ iff $\mathsf{H}_i \geq \mathsf{H}_j$. Assume for instance that the groups H_i are ascending military ranks. As people of higher rank are allowed to learn more secret information, no information shall flow to people of lower rank. So far, this can be expressed nicely using our above examples and definitions. However, this also means that an officer is not allowed to send *any* message to its soldiers. This problem is typically solved in practice by allowing information to be downgraded.

A typical way of downgrading is that a trusted user may do it. In the military example, an officer may traditionally be trusted to classify information correctly by attaching paper labels to it. Similarly, he may explicitly attach labels to digital inputs, e.g., H (high) or L (low) to each input m. This situation is not restricted to trusted human users, but also applies to trusted processes "above" the currently considered system. This system may be a multi-level network or a distributed operating system, and certain applications may be allowed to operate on multiple levels, including downgrades. The security to be defined is that of the underlying system, without knowing details of the trusted applications.

Security for this case essentially means that a low user L should only get information from H via the system if that information was explicitly downgraded by H. Thus, if L can guess a bit b that H knows, a distinguisher should exist that can also guess that bit, based only on the explicitly downgraded information. This is sketched in Figure 6.

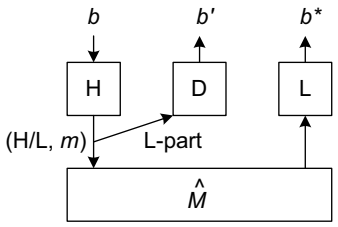

Figure 6. Downgrading by trusted user H. The untrusted user L should learn nothing from H via the system except what can be derived from H's downgraded information alone.

We first define our downgrading syntax, and then the non-interference notion.

Definition 4.1 *(Downgraded information)* Let a structure $(\hat{M}, S)$ and a subset $H \subseteq S$ of its specified ports be given. Let r be a run of a configuration of $(\hat{M}, S)$. Then the *downgraded information* at ports H in r, written $\mathsf{down}_H(r)$, is defined by further restricting $r\lceil_H$ to input ports, and to inputs that are pairs (L, m) of the constant L and an arbitrary message m. ◇

Definition 4.2 *(Intransitive probabilistic noninterference with downgrading)* Let a flow policy $\mathcal{G} = (\Delta, \mathcal{E})$ with $\Delta_{(\hat{M},S)} := \{S_i \mid i = 1, \ldots, n\} \cup \{\bar{S}\}$ for a structure $(\hat{M}, S)$ be given, and let one port set $H = S_i$ be given as that of the trusted user H. Then $(\hat{M}, S)$ fulfills the policy $\mathcal{G}$ if the following holds for all H, L with $\mathsf{H} \not\rightsquigarrow \mathsf{L}$, and all non-interference configurations $conf^{n\text{-}in}_{\mathsf{H},\mathsf{L}} \in \mathsf{Conf}^{n\text{-}in}_{\mathsf{H},\mathsf{L},\mathcal{I}}(\hat{M}, S)$: If

$$P(b = b^* \quad | \quad r \leftarrow run_{conf^{n\text{-}in}_{\mathsf{H},\mathsf{L}},k}; \; b := r\lceil_{\mathsf{p\_H\_bit}}!; \; b^* := r\lceil_{\mathsf{p}^*_{\mathsf{L}\_\mathsf{bit}}?}) \geq \frac{1}{2} + ns(k)$$

with $ns(k) \notin SMALL$, then there exists a distinguisher D and a function $ns'(k) \notin SMALL$ such that

$$P(b = b' \quad | \quad r \leftarrow run_{conf^{n\text{-}in}_{\mathsf{H},\mathsf{L}},k}; \; b := r\lceil_{\mathsf{p\_H\_bit}}!; \; b' \leftarrow \mathsf{D}(\mathsf{down}_H(r), 1^k)) \geq \frac{1}{2} + ns'(k).$$

For the computational case, the distinguisher has to be polynomial-time. ◇

The definition can be extended to predicates on ns, ns' like Definition 3.4.

5 Intransitive Non-Interference and Simulatability

In this section we investigate how intransitive non-interference behaves under simulatability. In a nutshell, we show that the relation "at least as secure as", which is the cryptographic analog of secure implementation, does not destroy the properties from our definitions. As defining a cryptographic system usually starts with an abstract specification of what the system should do, possible implementations have to be proven to be at least as secure as this specification. The abstract specification usually consists of a monolithic idealized machine containing neither cryptographic details nor probabilism. Thus, its properties can typically be validated quite easily by formal proof systems, e.g., formal theorem proving or even automatic model checking. Hence it would simplify life a lot if these properties would automatically carry over to the real system. This is already known for transitive non-interference [2]. In the following, we show that it is also true for recognition non-interference for trusted recipients (Definition 3.6). A similar preservation theorem also holds for the remaining definitions, but we omit the proofs due to space constraints.

Simulatability essentially means that whatever might happen to honest users U of the real structure $(\hat{M}_1, S)$ can also happen to the same honest users with the ideal structure $(\hat{M}_2, S)$, i.e., with the abstract specification. The structures have identical sets of specified ports. Formally speaking, for every configuration $conf_1$ of $(\hat{M}_1, S)$ there is a configuration $conf_2$ of $(\hat{M}_2, S)$ with the same users yielding indistinguishable views of U in both systems [30]. This is written $(\hat{M}_1, S) \geq_{\mathsf{sec}} (\hat{M}_2, S)$ (spoken "the real structure is at least as secure as the ideal structure"). The indistinguishability of the views of U is denoted by $view_{conf_1}(U) \approx view_{conf_2}(U)$.

The following theorem states that an intransitive non-interference property is in fact preserved under the relation "at least as secure as".

Theorem 5.1 *(Preservation of Recognition Non-Interference for Trusted Recipients) Let a flow policy $\mathcal{G} = (\Delta, \mathcal{E})$ for a structure $(\hat{M}_2, S)$ be given, such that this structure fulfills Definition 3.6 for this policy. Furthermore, let a structure $(\hat{M}_1, S) \geq_{\mathsf{sec}} (\hat{M}_2, S)$ be given. Then $(\hat{M}_1, S)$ also fulfills Definition 3.6 for the flow policy $\mathcal{G}$. This holds for the perfect, statistical, and computational case.* □

Proof. First, $\mathcal{G} = (\Delta, \mathcal{E})$ is a well-defined flow policy also for $(\hat{M}_1, S)$ because the two structures have the same set of specified ports. (This was the reason to define the partition Δ of all free ports via a partition Γ of the specified ports.)

We now show that $(\hat{M}_1, S)$ fulfills recognition non-interference for trusted recipients for the policy $\mathcal{G} = (\Delta, \mathcal{E})$.

Let arbitrary users H, L, a cut $\hat{C}$ for H, L, and a non-interference configuration $conf^{n\text{-}in}_{\mathsf{H},\mathsf{L},1} = (\hat{M}_1, S, \{\mathsf{H}, \mathsf{L}, \mathsf{T}_1, \ldots, \mathsf{T}_n\}, \mathsf{A}) \in \mathsf{Conf}^{n\text{-}in}_{\mathsf{H},\mathsf{L},\mathcal{I}}(\hat{M}_1, S)$ be given. Because of $(\hat{M}_1, S) \geq_{\mathsf{sec}} (\hat{M}_2, S)$, there exists a configuration $conf_{\mathsf{H},\mathsf{L},2} = (\hat{M}_2, S, \{\mathsf{H}, \mathsf{L}, \mathsf{T}_1, \ldots, \mathsf{T}_n\}, \mathsf{A}')$ such that $view_{conf^{n\text{-}in}_{\mathsf{H},\mathsf{L},1}}(\{\mathsf{H}, \mathsf{L}, \mathsf{T}_1, \ldots, \mathsf{T}_n\}) \approx view_{conf_{\mathsf{H},\mathsf{L},2}}(\{\mathsf{H}, \mathsf{L}, \mathsf{T}_1, \ldots, \mathsf{T}_n\})$. Recall that the set $U := \{\mathsf{H}, \mathsf{L}, \mathsf{T}_1, \ldots, \mathsf{T}_n\}$ of users remains unchanged in simulatability. It is easy to see that $conf_{\mathsf{H},\mathsf{L},2}$ is again a non-interference configuration by inspection of the precise definition in [2]. Hence, we call it $conf^{n\text{-}in}_{\mathsf{H},\mathsf{L},2}$ in the following.

For the statistical case, assume that L outputs the bit b correctly with probability at least $\frac{1}{2} + ns_1(k)$ for $ns_1 \notin SMALL$ in the configuration $conf^{n\text{-}in}_{\mathsf{H},\mathsf{L},1}$. We have to show that there exists a distinguisher D_1 that can output the bit b with a similar probability, given the view of the users in the cut $\hat{C}$ and the random tapes of $T_{\hat{C}}$ with $T_{\hat{C}}$ being the set of users that lie in the same component of the graph as L after applying the cut $\hat{C}$.

Because of $view_{conf^{n\text{-}in}_{\mathsf{H},\mathsf{L},1}}(U) \approx view_{conf^{n\text{-}in}_{\mathsf{H},\mathsf{L},2}}(U)$, there also exists $ns_2 \notin SMALL$ such that the probability of a correct guess of L in $conf^{n\text{-}in}_{\mathsf{H},\mathsf{L},2}$ is at least $\frac{1}{2} + ns_2(k)$. This follows with the definition of statistical indistinguishability, which states that the statistical distance $\delta := |\Delta(view_{conf^{n\text{-}in}_{\mathsf{H},\mathsf{L},1},k}(U), view_{conf^{n\text{-}in}_{\mathsf{H},\mathsf{L},2},k}(U))|$ is in $SMALL$. The predicate $b = b^*$ is a function of such a view of U, and by a well-known lemma the statistical distance between a function of two random variables is bounded by the statistical distance between the random variables themselves (see [2]), i.e., here by δ. Now assume for contradiction that the probability of a correct guess of L in $conf^{n\text{-}in}_{\mathsf{H},\mathsf{L},2}$ is at most $\frac{1}{2} + s_1(k)$ for $s_1 \in SMALL$. This would imply $s_1 + \delta \geq ns_2$, in contradiction to the fact that $SMALL$ is closed under addition and under making functions smaller.

As $(\hat{M}_2, S)$ fulfills $\mathcal{G}$ by precondition, this result about L's guessing ability in the second structure implies that there exists a distinguisher D_2 that, given the view of the users in the cut, $view_{conf^{n\text{-}in}_{\mathsf{H},\mathsf{L},2},k}(\hat{C})$, and the random tapes of $T_{\hat{C}}$, can output the bit b with probability at least $\frac{1}{2} + ns'_2(k)$ with $ns'_2 \notin SMALL$. Now we define the distinguisher for the first structure to be $\mathsf{D}_1 := \mathsf{D}_2$. This is possible because the input of D_2 only contains information collected by the users (their views and random tapes), whose types are unchanged in simulatability. By the same indistinguishability argument as above we see that D_1, given $view_{conf^{n\text{-}in}_{\mathsf{H},\mathsf{L},1},k}(\hat{C})$ and the random tape of $T_{\hat{C}}$, can also output the bit b with probability at least

$\frac{1}{2} + ns'_1(k)$ with $ns'_1 \notin SMALL$. This finishes the proof for the statistical case.

The proof for the perfect case follows with $SMALL := \{0\}$. For the computational case we use $SMALL := NEGL$. Here the contradiction proofs showing that if the guessing probabilities of L and $\mathsf{D} := \mathsf{D}_1 = \mathsf{D}_2$, respectively, were significantly different in the real and ideal structure, the structures would be distinguishable, is a bit different: One defines a polynomial-time distinguishing machine D^* (not to be confused with D) in a standard way: On input a view of the users, it computes what L or D would guess and whether this is correct, and accordingly guesses whether it has got a view from the real or the ideal structure. This guess is correct with probability significantly greater than $\frac{1}{2}$ by the closure properties of the class $SMALL$. ■

The omitted proofs of the corresponding preservation theorems for recognition non-interference (without trusted users) and for downgrading are completely analogous, and that of blocking non-interference is even easier, with only one usage of indistinguishability.

6 Security of Implementations with a Cryptographic Firewall

To illustrate the usefulness of the preservation theorem of the previous section, we now examine the realization of intransitive flow policies by cryptographic firewalls, which we mentioned in several examples above. A precise description of a firewall as such, i.e., a cryptographic realization, was given in [2]; only the final filtering rules were specific for transitive policies. Hence we only give an informal description here, and only for the secretary example.

The real firewall is a structure $(\hat{M}, S)$ with $\hat{M} := \{\mathsf{M}_i \mid i \in \{c, s, g\}\}$ and where S contains two ports for each machine M_i for inputs from and outputs to user I. A machine M_b for the bad customers would also belong to the intended structure, but as one cannot trust all bad customers to use correct machines, the actual structure joins that machine with the adversary machine A. This is sketched in Figure 7.

The machines M_i perform secure message transmission, i.e., on input (send, m, j) from their respective user, they sign the message with their secret signature key along with the identity of the sender and the recipient, encrypt it with the public key of j, and send it to machine M_j over an insecure network. When M_j receives a message from the network, it decrypts it and verifies the signature. If this fails, it discards the message, else it applies its filtering rules. Here M_c only accepts messages

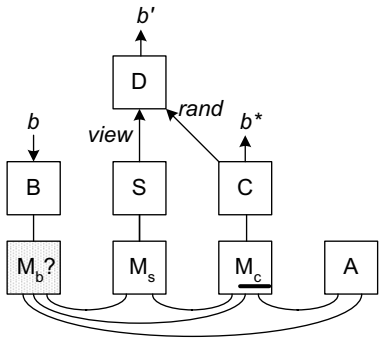

Figure 7. Sketch of the firewall system for the secretary example. The bar in machine M_c denotes the filtering function. The good customers G and their machine M_g are connected like S and M_s. A correct machine M_b may or may not be present.

from M_s and M_g, i.e., outputs them to its user C, while M_s and M_g accept messages from all machines. Note that the correctness of this test depends on the preceding signature test.

We want to prove that the structure $(\hat{M}, S)$ fulfills the policy from Figure 2 in the computational case according to Definition 3.6. As a proof for the real structure $(\hat{M}_1, S)$ would be quite complicated and error-prone, we apply the preservation theorem to get rid of cryptography-related details. In [2], an abstract specification $(\hat{M}_2, S)$ of the cryptographic firewall was presented and it was proved that the real structure is as secure as the ideal structure. Hence we can use this for our purposes. Roughly, instead of sending encrypted and signed messages over an insecure network, the abstract firewall just stores the messages in an internal array and delivers them at the appropriate scheduling requests. The internal test whether a message to C should be delivered is considerably simplified since the abstract firewall knows the sender of a message by construction: It can simply let messages from S and G through, but not from B. Thus, *no* information can flow directly from B to C in this ideal structure.

Now assume that C outputs the bit b correctly with probability at least $\frac{1}{2} + ns(k)$ for $ns \notin SMALL$. The only possible cut is $\hat{C} := \{\mathsf{S}, \mathsf{G}\}$. Thus we define a distinguisher for the abstract structure as follows, given the view of S and G in a run and the random tape of C: It simulates the user C by using C's input information, which the distinguisher knows as it must come from S or G (note that no information at all can flow from B to C directly), and the content of C's random tape. More precisely, the machine D proceeds through the joint view

of S and G. For each output that is directed to C, it calls the state transition φ_C of C with the current position on the random tape. If the function φ_C outputs a bit b^* at port $p^*_{C\_bit}!$, the distinguisher outputs this bit as its own guess, i.e., $b' := b^*$. Otherwise, it updates the position on the random tape of the simulated machine C to the first unused position, and continues proceeding through the view. It is easy to see that the distinguisher simulates the precise behavior of the machine C, since every input of C must have come from S or G.

Using the preservation theorem (Theorem 5.1), this property carries over to the concrete implementation, using cryptographic primitives and comprising error probabilities, without any further work.

Remark. The construction of D is so easy because direct information flow from B to C is completely prohibited, not just up to a negligible error probability. If we had performed this proof for the real system, we would have had to deal with certain runs where direct flow indeed occurred, namely those where B succeeded in forging a signature of S or G. These runs cannot be properly simulated by the distinguisher as it lacks the incoming information of C in this case. One possibility to deal with that is to collect these runs in so-called error sets, and to show afterwards that the aggregated probability of these runs is still negligible, or the underlying cryptography could be broken. Although this technique is common in simulatability proofs (e.g., see [21]), we do not want to bother with it when proving non-interference for particular applications. Hence our preservation theorem is indeed very useful.

7 Related Literature

Initiated by the deterministic definition of Goguen and Meseguer [5], the notion of non-interference for transitive policies was extended by many articles, with definitions for possibilistic and non-deterministic systems [27, 15, 9, 29, 19, 16, 31, 4, 13].

Although the notion of probabilistic non-interference was already introduced in 1992 by Gray [7], probabilistic information flow has rather been overlooked for quite some time. However, because of the tremendous development of cryptography, new interest arose in capturing what information flow means in the context of cryptography. Laud [11] defined real computational secrecy for a sequential language, and presented a calculus. However, only encryption is covered so far, i.e., other important concepts like authentication, pseudo-number generators, etc. are not considered. Moreover, the definition is non-reactive, i.e., it does not comprise continuous interaction between the user, the adversary, and the system, which is a severe restriction to the set of considered cryptographic systems. Volpano [28] investigated which conditions are needed so that one-way functions can be used safely in a programming language, but he did not express non-interference, but the secrecy of a specific secret. Abadi and Blanchet [1] introduced type systems where asymmetric communication primitives, especially public-key encryption can be expressed, but these primitives are only relative to a Dolev-Yao abstraction [3], i.e., the primitives are idealized so that no computational non-interference definition is needed. For a discussion why the Dolev-Yao abstraction is not justified by current cryptography, see [20]. Recently, we introduced the notion of computational probabilistic non-interference [2]. This definition is based on a fully reactive setting, comprises error probabilities and computational restrictions, and hence allows for analyzing information flow for general cryptographic primitives. However, this definition, as all the previous ones, was designed specifically for transitive information flow policies.

For intransitive policies, no prior work for a probabilistic definition of flow exists to the best of our knowledge. For deterministic systems, Goguen and Meseguer proposed a definition of intransitive information flow based on an *unless* construct [6]. This definition serves as an extension of the original definition of Goguen and Meseguer [5] for transitive flow policies. However, this *unless* construct did not meet the intuitive requirements of intransitive flow, as it accepted many intuitively insecure systems as being secure. The first satisfactory formal definition of intransitive flow was proposed by Rushby [24], followed by definitions by Pinsky [22] and recently by Roscoe and Goldsmith [23]. Today, Rushby's approach to information flow for intransitive policies seems to be the most popular one for deterministic systems as case studies have shown its feasibility for real applications [25]. Besides these definitions for deterministic systems, Mantel presented a new approach for intransitive flow that is suited for non-deterministic systems [14]. However, since all these definitions are based on non-probabilistic systems, they are not suited to capture probabilistic behaviors, and in particular cryptographic applications. Our definitions solve this problem as they can be used to capture intransitive non-interference for systems involving arbitrary cryptographic primitives.

Finally, we briefly address downgrading of information as the most common application of intransitive flow. The problem of many of the above definitions is that they are overly restrictive, preventing many useful systems from being built. This led to the approach of downgrading certain information so that it may subse-

quently leak from the system [18, 32]. The amount of leaked information can in some cases be rigorously defined using information-theoretic techniques [17, 10]. As prior approaches of downgrading are specific to non-probabilistic systems, our definition is the first that captures downgrading for probabilistic information flow.

8 Conclusion

Despite the recent interest in linking information flow and probabilism, no probabilistic formalism existed to capture intransitive flow policies. In particular this created a major gap between information flow and applications based on cryptographic primitives. In this article, we tried to bridge this gap by proposing definitions for intransitive flow for probabilistic systems, in particular blocking non-interference, recognition non-interference, and the weaker recognition non-interference for trusted recipients. The first definition is incomparable with the other two, and typically one of each type should be fulfilled. Further, we have defined downgrading, in other words the correct handling of mixed inputs by a system.

We took care that our definitions can indeed be fulfilled by cryptographic primitives by capturing error probabilities, computational restrictions, and fully reactive systems. In situations where several third parties might be involved in an intransitive flow, we had to consider how cuts of the flow graph block the flow or join their knowledge to detect information flow. This illustrates a major difference between the probabilistic and the non-probabilistic approach, pointing out why the notion of probabilistic information flow is much more fine-grained.

We have finally shown that intransitive non-interference properties proved for abstract specifications carry over to concrete implementations without any further work, provided that the implementations are correct according to the simulatability definitions of modern cryptography.

Acknowledgments

We thank Heiko Mantel and Michael Waidner for helpful discussions.

References

[1] M. Abadi and B. Blanchet. Secrecy types for asymmetric communication. In *Proc. 4th International Conference on Foundations of Software Science and Computation Structures (FOSSACS)*, volume 2030 of *Lecture Notes in Computer Science*, pages 25–41. Springer, 2001.

[2] M. Backes and B. Pfitzmann. Computational probabilistic non-interference. In *Proc. 7th European Symposium on Research in Computer Security (ESORICS)*, volume 2502 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2002.

[3] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[4] R. Focardi and F. Martinelli. A uniform approach to the definition of security properties. In *Proc. 8th Symposium on Formal Methods Europe (FME 1999)*, volume 1708 of *Lecture Notes in Computer Science*, pages 794–813. Springer, 1999.

[5] J. A. Goguen and J. Meseguer. Security policies and security models. In *Proc. 3rd IEEE Symposium on Security & Privacy*, pages 11–20, 1982.

[6] J. A. Goguen and J. Meseguer. Unwinding and inference control. In *Proc. 5th IEEE Symposium on Security & Privacy*, pages 75–86, 1984.

[7] J. W. Gray III. Toward a mathematical foundation for information flow security. *Journal of Computer Security*, 1(3):255–295, 1992.

[8] C. A. R. Hoare. *Communicating Sequential Processes*. International Series in Computer Science, Prentice Hall, Hemel Hempstead, 1985.

[9] D. M. Johnson and F. Javier Thayer. Security and the composition of machines. In *Proc. 1st IEEE Computer Security Foundations Workshop (CSFW)*, pages 72–89, 1988.

[10] M. H. Kang, I. S. Moskowitz, and D. C. Lee. A network version of the pump. In *Proc. 16th IEEE Symposium on Security & Privacy*, pages 144–154, 1995.

[11] P. Laud. Semantics and program analysis of computationally secure information flow. In *Proc. 10th European Symposium on Programming (ESOP)*, pages 77–91, 2001.

[12] N. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, San Francisco, 1996.

[13] H. Mantel. Unwinding possibilistic security properties. In *Proc. 6th European Symposium on Research in Computer Security (ESORICS)*, volume 1895 of *Lecture Notes in Computer Science*, pages 238–254. Springer, 2000.

[14] H. Mantel. Information flow control and applications – bridging a gap. In *Proc. 10th Symposium on Formal Methods Europe (FME 2001)*, volume 2021 of *Lecture Notes in Computer Science*, pages 153–172. Springer, 2001.

[15] D. McCullough. Specifications for multi-level security and a hook-up property. In *Proc. 8th IEEE Symposium on Security & Privacy*, pages 161–166, 1987.

[16] J. McLean. Security models. Chapter in *Encyclopedia of Software Engineering*, 1994.

[17] J. K. Millen. Covert channel capacity. In *Proc. 8th IEEE Symposium on Security & Privacy*, pages 60–66, 1987.

[18] A. Myers and B. Liskov. Protecting privacy using the decentralized label model. *ACM Transactions on Software Engineering and Methodology*, pages 410–442, 2000.

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

[20] B. Pfitzmann and M. Waidner. Composition and integrity preservation of secure reactive systems. In *Proc. 7th ACM Conference on Computer and Communications Security*, pages 245–254, 2000.

[21] B. Pfitzmann and M. Waidner. A model for asynchronous reactive systems and its application to secure message transmission. In *Proc. 22nd IEEE Symposium on Security & Privacy*, pages 184–200, 2001.

[22] S. Pinsky. Absorbing covers and intransitive noninterference. In *Proc. 16th IEEE Symposium on Security & Privacy*, pages 102–113, 1995.

[23] A. Roscoe and M. Goldsmith. What is intransitive noninterference? In *Proc. 12th IEEE Computer Security Foundations Workshop (CSFW)*, pages 226–238, 1999.

[24] J. Rushby. Noninterference, transitivity, and channel-control security. Technical report, Computer Science Laboratory, SRI International, 1992.

[25] G. Schellhorn, W. Reif, A. Schairer, P. Karger, V. Austel, and D. Toll. Verification of a formal security model for multiapplicative smart cards. In *Proc. 6th European Symposium on Research in Computer Security (ESORICS)*, volume 1895 of *Lecture Notes in Computer Science*, pages 17–36. Springer, 2000.

[26] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, Nov. 1979.

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

[28] D. Volpano. Secure introduction of one-way functions. In *Proc. 13th IEEE Computer Security Foundations Workshop (CSFW)*, pages 246–254, 2000.

[29] J. T. Wittbold and D. M. Johnson. Information flow in nondeterministic systems. In *Proc. 11th IEEE Symposium on Security & Privacy*, pages 144–161, 1990.

[30] A. C. Yao. Protocols for secure computations. In *Proc. 23rd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 160–164, 1982.

[31] A. Zakinthinos and E. S. Lee. A general theory of security properties. In *Proc. 18th IEEE Symposium on Security & Privacy*, pages 94–102, 1997.

[32] S. Zdancewic and A. C. Myers. Robust declassification. In *Proc. 14th IEEE Computer Security Foundations Workshop (CSFW)*, pages 15–23, 2001.