Quantifying Probabilistic Information Flow in Computational Reactive Systems

Michael Backes

IBM Zurich Research Laboratory
mbc@zurich.ibm.com

Abstract. Information flow and non-interference are well-established techniques for expressing both integrity and privacy properties. Because of the enormous potential to transmit information using probabilistic methods of cryptography, interest has arisen in extending the traditional notions of information flow to fully reactive settings that allow for reasoning about arbitrary interactive systems, and in particular arbitrary cryptographic protocols. We propose definitions for quantifying the amount of information that users are able to transmit to each other in such reactive settings, and we in particular address computational restrictions and error probabilities so that our definitions are suited for complexity-theoretic reasoning about cryptographic systems. We show that our definitions are preserved under simulatability, which constitutes the cryptographic notion of a secure implementation, and we link our definitions to non-interference by showing that a zero or negligible quantity of information flow is equivalent to perfect or computational probabilistic non-interference, respectively.

1 Introduction

Information flow and non-interference have become powerful possibilities for expressing both privacy and integrity requirements. The concept of information flow was first investigated for secure operating systems by Lampson [17] and subsequently by Bell and LaPadula [4] and Denning [7]. Initiated by the work on non-interference of Goguen and Meseguer [11,12], various definitions have subsequently been proposed that rigorously specify when information flow is considered to occur for possibilistic and non-deterministic systems [33,24,37,26,30,10,22,23] and for probabilistic systems [13,14,25,35,32]. Whereas these lines of work concentrated on the absence of information flow in various settings, they were accompanied by work that gave quantitative measurements of the information that might flow between certain users, motivated by use cases where some flow of information might be inevitable or acceptable [27,16,20,5,8].

Recently, interest has arisen in generalizing definitions of information flow so that they allow for reasoning about real cryptographic protocols in order to capture the variety of cryptographic techniques that can be used to transmit information in a secret or undetectable way, e.g., encryption or steganographic tech-

S. De Capitani di Vimercati et al. (Eds.): ESORICS 2005, LNCS 3679, pp. 336–354, 2005.

niques. The incorporation of cryptographic reasoning into information flow definitions posed major challenges because a faithful analysis of cryptography requires not only probabilistic behaviors but also error probabilities and polynomial-time restrictions in terms of computational complexity. Moreover, a suitable definition has to capture a reactive environment, i.e., continuous interaction between users, an adversary, and the system. These problems recently led to the notion of computational probabilistic non-interference [1,2], which was the first definition that allowed for reasoning about information flow in a reactive setting and the presence of cryptography. However, quantitative measurements of information flow in reactive settings and particularly in the presence of arbitrary cryptographic protocols have not been addressed yet.

We present the first definitions for quantifying the amount of information that one user is able to transmit to another user within a reactive setting. We present definitions for unconditional security that are suitable for reasoning about informational-theoretically secure or non-cryptographic systems, as well as computational definitions that comprise complexity-theoretic reasoning such as polynomially bounded adversaries, allow error probabilities, and are tightly related to well-established cryptographic notions such as computational indistinguishability. Roughly, our approach to quantify an information flow from a high user to a low user is to consider different behaviors of the high user that result in different views of the low users (different probability distributions), to then measure the distance of these distributions, and to finally maximize the resulting measurement for different behaviors of the high user. Both the unconditional and the computational definitions comprise malicious or predefined behaviors of third parties as well as timing aspects.

We show that our definitions are preserved under simulatability, which constitutes the cryptographic notion of a secure implementation, i.e., securely implementing a specification in the sense of simulatability may not increase the transmitted information in the unconditional case, and only by a negligible quantity in the computational case. This significantly simplifies the determination of the information flow quantity permitted within a cryptographic system, since simulatability helps to eliminate cryptography-related details such as error probabilities and computational restrictions. Moreover, we show that a zero or negligible quantity of information flow is equivalent to perfect and computational probabilistic non-interference. With our simulatability preservation theorem, this in particular allows for a short, alternative proof that non-interference properties are preserved under simulatability [1].

Further Related Literature. The only definitions of information flow that reside in a reactive scenario and that allow for complexity-theoretic reasoning have been presented by Backes and Pfitzmann in [1,2] based on the model of reactive simulatability [29,3]; quantitative aspects of information flow are, however, not considered there.

The work that comes closest to ours in terms of quantifying information flow is the one on *approximate non-interference* of Di Pierro et al. [8]. They defined the notion of ε-confinement that captured that information flow is still

acceptable if the distance of views of specific user deviate only up to probability ε. Although their definition does not address computational aspects as needed for cryptographic purposes, our work is nevertheless inspired by some of their ideas. Lowe [20] measured the amount of information in a non-probabilistic setting by counting the number of different behaviors of the high user that yield different views for the low user. Clark et al. [5] proposed syntax-directed inference rules for computing estimates on information flow in an imperative language. Both works do not aim to deal with computational aspects.

Early ideas of quantitative security based on Shannon's information theory go back to Denning's work [6], which was subsequently used in [27,16] to measure the quantity of covert channels. The investigated settings, however, were simplistic in that the channels were memoryless, there was no input feedback in the channel, and only uncorrelated inputs; moreover, no computational restrictions were taken into account there. This stands in blatant contrast to reactive scenarios that allow for expressing and analyzing arbitrary (cryptographic) primitives and protocols, where inputs and user behaviors are typically highly correlated and protocols are highly stateful. We consider it interesting future work to extend the information-theoretic line of work to our unconditional definitions, and we have some basic ideas on this subject that we intend to pursue.

Recent research has also investigated non-interference properties involving real cryptographic primitives, but without investigating quantitative aspects. Laud [18,19] presented a sequential language for which he expressed real computational secrecy. The definition is non-reactive and specific to encryption as the only cryptographic primitive. Volpano [34] investigated conditions for safely using one-way functions in a programming language, but his underlying definition does not express non-interference, but the secrecy of a specific secret.

Outline of the Paper. In Section 2 we briefly review the underlying model of reactive simulatability, which is an asynchronous probabilistic execution model with distributed scheduling, including computational aspects as needed for cryptography. We give our definitions for capturing the quantity of information transmitted between two users in a fully reactive scenario—including the presence of cryptographic techniques—in Section 3. In Section 4 we show that our definitions are preserved under simulatability, and we finally show in Section 5 that a zero or negligible quantity of information flow is equivalent to existing notions of perfect and computational probabilistic non-interference. We conclude with a summary of our results in Section 6.

2 The Model of Reactive Simulatability

Our work is based on the model of reactive simulatability [29,3], which is an asynchronous probabilistic execution model with distributed scheduling that provides universal composability properties while including computational aspects as needed for cryptography. The model is automata based, i.e., protocols are executed by interacting machines, and event-based, i.e., machines react on

certain inputs. All details of the model that are not necessary for understanding are omitted here; for completeness, we give rigorous definitions of the relevant notions in Appendix A.

2.1 General System Model

A *machine* is a probabilistic IO automaton (extended finite-state machine) in a slightly refined model to allow complexity considerations. For these automata Turing-machine realizations are defined, and the complexity thereof is measured in terms of a common security parameter k, given as the initial work-tape content of every machine. A *structure* consists of a set $\hat{M}$ of connected machines and a subset S of free *ports*, called *service ports*. Each structure is complemented to a *configuration* by a set of *user* machines U and an *adversary* machine A. The machines in U connect only to ports in S, whereas A connects to the remaining free ports $\bar{S}$ of the structure and may interact with the users. We denote the set of configurations of a structure $(\hat{M}, S)$ by $\mathsf{Conf}(\hat{M}, S)$ and the subset of polynomial-time configurations by $\mathsf{Conf}_{\mathsf{poly}}(\hat{M}, S)$.[1]

The general scheduling model in [29,3] gives each connection c (from an output port $\mathsf{c}!$ to an input port $\mathsf{c}?$) a buffer, and the machine with the corresponding clock port $\mathsf{c}^{\triangleleft}!$ can schedule a message there when it makes a transition. In real cryptographic systems, network connections are typically scheduled by A, which usually serves as a master scheduler, but the model allows for specifying other designated master schedulers as well as local schedulers for specific connections. 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. 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, one can define a probability space on the possible runs by a canonical construction as for Markov chains, cf. [3] for the precise definition. We call the corresponding random variable $run_{conf,k}$ 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. For a configuration *conf*, the corresponding random variables over the probability space of all possible runs are denoted by $view_{conf,k}(\mathsf{M})$ and $view_{conf,k}(\hat{M})$, respectively.

[1] Here and elsewhere we change some notation of [29,3] 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.

2.2 Partition Configurations for Defining Information Flow

Structures and configurations in their general form impose no restrictions regarding which user can connect to which service ports, e.g., users in different configurations might connect to a different subset of the service ports. For quantifying information flow in a reactive environment, as well as for the mere detection of information flow as defined in [1,2], we need security domains between which we can analyze the flow of information. It is intuitive to regard the possible protocol participants as security domains. However, to be independent of the details of the actual user and the adversary machines, we represent users by the ports they connect to in the considered structure $(\hat{M}, S)$, and the adversary by the remaining free ports of the structure. This means that we consider a partition $\Gamma = \{S_i \mid i \in \mathcal{I}\}$ of the set S of service ports, where $\mathcal{I}$ is an arbitrary finite index set. We can now designate each user H_i with $i \in \mathcal{I}$ by the subset of service ports S_i it connects to. For a given structure and a partition of its service ports, those configurations where each user only connects to its ports of the partition, and where the adversary connects to the remaining free ports of the structure are called *partition configurations*. A characteristic of partition configurations is that the different user machines and the adversary have no direct connections, because otherwise they could trivially transmit information without relying on the possibilities granted by the structure. Moreover, a specific fair master scheduler X is added to the configuration because if the adversary were allowed to schedule the connections to and from the users, it could always achieve probabilistic information flow, cf. [1,2] for more details. We denote the set of partition configurations of a structure $(\hat{M}, S)$ and partition Γ by $\mathsf{Conf}(\hat{M}, S, \Gamma)$ and the subset of polynomial-time ones by $\mathsf{Conf}_{\mathsf{poly}}(\hat{M}, S, \Gamma)$. Finally, we consider the subset of partition configurations where users are only allowed to perform a certain number of steps. This will allow us to reason about timing aspects of information flow. We call a partition configuration with index set $\mathcal{I}$ a *timed* partition configuration *for a function* $\varphi\colon \mathcal{I} \to (\mathbb{N} \to \mathbb{N} \cup \{\infty\})$, if the user H_i in this configuration only makes $\varphi(i)$ outputs (as a function of k, the security parameter). We call the set of these configurations $\mathsf{Conf}^{\varphi}(\hat{M}, S, \Gamma)$ and the set of polynomial ones $\mathsf{Conf}^{\varphi}_{\mathsf{poly}}(\hat{M}, S, \Gamma)$.

3 Measuring Probabilistic Information Flow in Reactive Settings

We now define the amount of information that one user is able to transmit to another user via a particular structure. Using standard terminology, we call these two users the *high user* and the *low user*. The remaining users are referred to as *third parties*.

Roughly speaking, the idea of quantifying information flow is that we consider different behaviors of the high user that result in different views of the low users, and measure the distance between these different views. Finally, we maximize this distance for all possible behaviors of the high user, which gives the desired

measure of the information quantity. The notion hence provides information on how much two behaviors of the high user might differ in the worst case, given only the view of the low user, and it hence resembles the similarity relation of [8].

It remains to decide to what extent the third parties might contribute to the information flow. The most stringent choice is to regard every third party as malicious, i.e., it fully exploits its possibilities to help the high user to transmit information to the low user. Formally, this means that we quantify over the behavior of all third parties to maximize the distance, and we speak of the *worst-case information quantity* in this scenario. This approach is the one commonly taken in the literature, as it gives an upper bound on the amount of information flow under worst-case assumptions. Moreover, it is naturally linked to the notion of non-interference, i.e., absence of information flow, as we will see in Section 5. Based on this core definition, we introduce several variants and extensions, including more benign behaviors of the third parties as well as timing aspects.

Definition 1. *(Worst-Case Information Quantity)* Let $(\hat{M}, S)$ be a structure, let $\Gamma = \{S_i \mid i \in \mathcal{I}\}$ be a partition on the set S of service ports for a finite set $\mathcal{I}$, and let $||\cdot,\cdot||$ be a distance of user views, i.e., of probability distributions. Furthermore, let $H, L \in \mathcal{I}$ be given. Then the *worst-case information quantity* $\mathcal{Q}^{||\cdot,\cdot||}_{(\hat{M},S,\Gamma)}(H,L)$ that the high user H_H is allowed to transmit to H_L is defined as

$$\mathcal{Q}^{||\cdot,\cdot||}_{(\hat{M},S,\Gamma)}(H,L) := \max_{conf_1, conf_2 \in \mathsf{Conf}(\hat{M},S,\Gamma)} ||view_{conf_1,k}(\mathsf{H}_L), view_{conf_2,k}(\mathsf{H}_L)||,$$

such that $conf_l$ is of the form $conf_l := (\hat{M}, S, U_l, \mathsf{A})$ with $U_l = \{\mathsf{H}^{(l)}_H, \mathsf{H}_L, \mathsf{X}\} \cup \{\mathsf{H}_i \mid i \in \mathcal{I} \setminus \{H, L\}\}$ for an arbitrary adversary A and arbitrary users $\mathsf{H}^{(1)}_H, \mathsf{H}^{(2)}_H, \mathsf{H}_L$, and H_i for $i \in \mathcal{I} \setminus \{H, L\}$. The *polynomial-time worst-case information quantity* $\mathcal{Q}_{\mathcal{P}}{}^{||\cdot,\cdot||}_{(\hat{M},S,\Gamma)}(H,L)$ is defined similarly by taking the maximum over $\mathsf{Conf}_{\mathsf{poly}}(\hat{M}, S, \Gamma)$. ◇

Several extensions of this definition are useful. First, it is often more natural to consider fixed behaviors for some of the third parties because in real world examples, e.g., when a spy attempts to transmit information out of a company, it is unlikely that every employee will help the spy to do so. Formally, this means that we consider fixed user behaviors for a subset $\mathcal{J} \subseteq \mathcal{I}$, i.e., we parameterize the information quantity by a set $\mathcal{M} := \{\mathsf{H}_j \mid j \in \mathcal{J}\}$. The remaining users are considered malicious as in the previous definition. We speak of the *generalized information quantity* here because we obtain the worst-case definition as the special case $\mathcal{M} = \emptyset$.

Definition 2. *(Generalized Information Quantity)* Consider the preconditions as in Definition 1 and let $\mathcal{M} := \{\mathsf{H}_j \mid j \in \mathcal{J}\}$ be given for fixed machines H_j and $\mathcal{J} \subseteq \mathcal{I}$. Then the *generalized information quantity* $\mathcal{Q}^{||\cdot,\cdot||,\mathcal{M}}_{(\hat{M},S,\Gamma)}(H,L)$ *with respect to* $\mathcal{M}$ is defined as

$$\mathcal{Q}^{||\cdot,\cdot||,\mathcal{M}}_{(\hat{M},S,\Gamma)}(H,L) := \max_{conf_1, conf_2 \in \mathsf{Conf}(\hat{M},S,\Gamma)} ||view_{conf_1,k}(\mathsf{H}_L), view_{conf_2,k}(\mathsf{H}_L)||,$$

such that $conf_l$ is of the form $conf_l := (\hat{M}, S, U_l, \mathsf{A})$ with $U_l = \{\mathsf{H}_H^{(l)}, \mathsf{H}_L, \mathsf{X}\} \cup \{\mathsf{H}_i \mid i \in \mathcal{I} \setminus \{H, L\}\}$ for an arbitrary adversary A and arbitrary users $\mathsf{H}_H^{(1)}, \mathsf{H}_H^{(2)}, \mathsf{H}_L$, and H_i for $i \in \mathcal{I} \setminus (\mathcal{J} \cup \{H, L\})$, i.e., the maximum is only taken over those configurations in which the users H_j for $j \in \mathcal{J}$ are fixed by the parameter $\mathcal{M}$. The polynomial-time variant $\mathcal{Q}_{\mathcal{P}(\hat{M},S,\Gamma)}^{||\cdot,\cdot||,\mathcal{M}}(H, L)$ is defined as usual. ◇

Timing capabilities of certain users are typically of interest, i.e., to model that a spy should not be allowed to send data all the time, or only has limited access to his machine. We use timed partition configurations for this and speak of the *timed* generalized information quantity.

Definition 3. *(Timed Generalized Information Quantity)* Consider the preconditions as in Definition 2 and let in addition a function $\varphi\colon \mathcal{I} \to (\mathbb{N} \to \mathbb{N} \cup \{\infty\})$ be given. Then the *timed generalized information quantity* $\mathcal{Q}_{(\hat{M},S,\Gamma)}^{||\cdot,\cdot||,\mathcal{M},\varphi}(H, L)$ *with respect to* $\mathcal{M}, \varphi$ is defined as in Definition 2 except that the maximum is only taken over $\mathsf{Conf}^{\varphi}(\hat{M}, S, \Gamma)$. The polynomial-time variant $\mathcal{Q}_{\mathcal{P}(\hat{M},S,\Gamma)}^{||\cdot,\cdot||,\mathcal{M},\varphi}(H, L)$ is defined as usual. ◇

4 Preservation of Information Quantities Under Simulatability

We now investigate how the information quantity behaves under simulatability, which is the cryptographic notion of secure implementation. For reactive systems, it means that whatever might happen to users in a real structure $(\hat{M}_1, S)$ can also happen to users in an ideal structure $(\hat{M}_2, S)$ (with the same set of service ports to which the same users can connect). Formally, for every set U of polynomial-time users, and every polynomial-time adversary A_1, there exists a polynomial-time adversary A_2 such that the views of the machines in U are computationally indistinguishable when run either with $(\hat{M}_1, S)$ or with $(\hat{M}_2, S)$. This is illustrated in Figure 1. Indistinguishability is a well-known cryptographic notion from [38].

Definition 4. *(Computational Indistinguishability)* Two families $(\mathsf{var}_k)_{k\in\mathbb{N}}$ and $(\mathsf{var}'_k)_{k\in\mathbb{N}}$ of random variables on common domains D_k are *computationally indistinguishable* ("$\approx$") iff for every algorithm Dis (the distinguisher) that is probabilistic polynomial-time in its first input, we have

$$|P(\mathsf{Dis}(1^k, \mathsf{var}_k) = 1) - P(\mathsf{Dis}(1^k, \mathsf{var}'_k) = 1)| \in NEGL,$$

where $NEGL$ denotes the set of all *negligible functions*, i.e., $g\colon \mathbb{N} \to \mathbb{R}_{\geq 0} \in NEGL$ iff for all positive polynomials Q, $\exists k_0 \forall k \geq k_0\colon\ g(k) \leq 1/Q(k)$. ◇

Intuitively, given the security parameter and an element chosen according to either var_k or var'_k, Dis tries to guess which distribution the element came from.

Definition 5. *(Reactive Simulatability)* Let structures $(\hat{M}_1, S)$ and $(\hat{M}_2, S)$ be given. We say that $(\hat{M}_1, S)$ is *at least as secure as* $(\hat{M}_2, S)$, written $(\hat{M}_1, S) \geq_{\mathsf{sec}}^{\mathsf{poly}}$

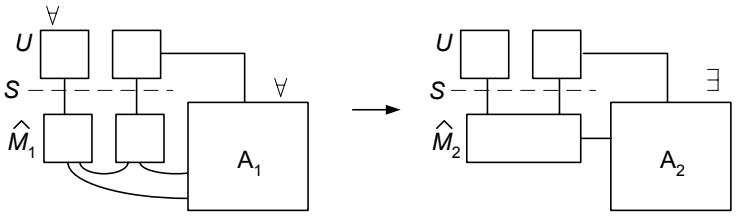

Fig. 1. Reactive simulatability: The two views of U must be indistinguishable

$(\hat{M}_2, S)$ if for every configuration $conf_1 = (\hat{M}_1, S, U, \mathsf{A}_1) \in \mathsf{Conf}_{\mathsf{poly}}(\hat{M}_1, S)$, there exists a configuration $conf_2 = (\hat{M}_2, S, U, \mathsf{A}_2) \in \mathsf{Conf}_{\mathsf{poly}}(\hat{M}_2, S)$ such that

$$view_{conf_1}(U) \approx view_{conf_2}(U).$$

We speak of *perfect reactive simulatability*, written $(\hat{M}_1, S) \geq^{\mathsf{perf}}_{\mathsf{sec}} (\hat{M}_2, S)$, if the above formula holds for all (also non-polynomially bounded) configurations of the respective structures, and with indistinguishability replaced by equality. $\diamond$

4.1 Preservation of Information Quantities

The following theorem establishes that the information quantity between two users is essentially unchanged under reactive simulatability. More precisely, the theorem states that only a negligible additional quantity of information can be transmitted to the low user when simulatability is applied, provided that the employed distance respects computational indistinguishability in a natural manner, i.e., two ensembles are indistinguishable if and only if their distance constitutes a negligible function. We call such distances *computational distances*. In the case of perfect reactive simulatability, we even show that no additional information can be sent to the low user for any distance. (Note that reactive simulatability is not symmetric, hence we cannot rule out that the user can only transmit *less* information when interacting with a real structure rather than when interacting with the ideal structure.)

These are exactly the properties that already allow modular and cryptographically sound proofs on the abstract level: For instance, an ideal specification that should prohibit the flow of information between two users has information quantity zero because it allows no communication between these two users by construction, e.g., the ideal firewall presented in [1] is of this kind. This is typically much easier to prove than for a cryptographic realization where the restriction on the information flow might be achieved by cryptographic techniques, e.g., digital signatures in the real implementation of the firewall. The theorem hence allows for conveniently analyzing the information flow properties of real cryptographic systems by means of their ideal counterparts, and we can hope that well-established techniques for enforcing the absence respectively measuring the quantity of information flow based on type checking techniques, e.g., [36,9,28,35,31,32,39], can be applied to our setting. Moreover, a negligible

amount of information is the best we can hope for in the presence of asymmetric cryptography because a negligible probability of error there always remains.

Theorem 1. *(Preservation of Information Quantity)* Let two structures $(\hat{M}_i, S)$ for $i = 1, 2$ be given, let $\Gamma := \{S_i \mid i \in \mathcal{I}\}$ be a partition of S for a finite index set $\mathcal{I}$, and let $H, L \in \mathcal{I}$. Let $\mathcal{M} := \{\mathsf{H}_j \mid j \in \mathcal{J}\}$ for some $\mathcal{J} \subseteq \mathcal{I}$ and $\varphi\colon \mathcal{I} \to (\mathbb{N} \to \mathbb{N} \cup \{\infty\})$ arbitrary. Then $(\hat{M}_1, S) \geq_{\mathsf{sec}}^{\mathsf{perf}} (\hat{M}_2, S)$ implies

$$\mathcal{Q}^{||\cdot,\cdot||,\mathcal{M},\varphi}_{(\hat{M}_1,S,\Gamma)}(H, L) \leq \mathcal{Q}^{||\cdot,\cdot||,\mathcal{M},\varphi}_{(\hat{M}_2,S,\Gamma)}(H, L)$$

for every distance $||\cdot,\cdot||$. Moreover, $(\hat{M}_1, S) \geq_{\mathsf{sec}}^{\mathsf{poly}} (\hat{M}_2, S)$ implies

$$\mathcal{Q}_{\mathcal{P}}{}^{||\cdot,\cdot||_{\mathsf{c}},\mathcal{M},\varphi}_{(\hat{M}_1,S,\Gamma)}(H, L) \leq \mathcal{Q}_{\mathcal{P}}{}^{||\cdot,\cdot||_{\mathsf{c}},\mathcal{M},\varphi}_{(\hat{M}_2,S,\Gamma)}(H, L) + \epsilon(k)$$

for some $\epsilon \in \mathit{NEGL}$ and every computational distance $||\cdot,\cdot||_{\mathsf{c}}$. □

Proof. Let $\mathit{conf}_1^1 = (\hat{M}_1, S, U_1, \mathsf{A})$ and $\mathit{conf}_2^1 = (\hat{M}_1, S, U_2, \mathsf{A})$ be two (polynomial-time) partition configurations in $\mathsf{Conf}^{\varphi}(\hat{M}_1, S, \Gamma)$ with $U_l := \{\mathsf{H}_H^{(l)}, \mathsf{H}_L, \mathsf{X}\} \cup \{\mathsf{H}_i \mid i \in \mathcal{I} \setminus \{H, L\}\}$ (for arbitrary machines H_i for $i \in \mathcal{I} \setminus (\mathcal{J} \cup \{H, L\})$) such that $||(\mathit{view}_{\mathit{conf}_1^1,k}(\mathsf{H}_L), \mathit{view}_{\mathit{conf}_2^1,k}(\mathsf{H}_L))||$ is equal to $\mathcal{Q}^{||\cdot,\cdot||,\mathcal{M},\varphi}_{(\hat{M}_1,S,\Gamma)}(H, L)$ or to $\mathcal{Q}_{\mathcal{P}}{}^{||\cdot,\cdot||_{\mathsf{c}},\mathcal{M},\varphi}_{(\hat{M}_1,S,\Gamma)}(H, L)$, respectively, in the polynomial case. Owing to $(\hat{M}_1, S) \geq_{\mathsf{sec}} (\hat{M}_2, S)$, (polynomial-time) configurations conf_1^2, $\mathit{conf}_2^2 \in \mathsf{Conf}^{\varphi}(\hat{M}_2, S)$ exist such that $\mathit{view}_{\mathit{conf}_1^1,k}(U_1) \approx \mathit{view}_{\mathit{conf}_1^2,k}(U_1)$ and $\mathit{view}_{\mathit{conf}_2^1,k}(U_2) \approx \mathit{view}_{\mathit{conf}_2^2,k}(U_2)$. Moreover, we obviously have conf_1^2, $\mathit{conf}_2^2 \in \mathsf{Conf}^{\varphi'}(\hat{M}_2, S, \Gamma)$ because the users and the set of service ports are unchanged under simulatability, and $\varphi = \varphi'$ as the users' view could otherwise be trivially distinguished in both configurations (the distinguisher waits until one user stops in one configuration but continues to send messages in the other configuration).

We now restrict the views of U_1 and U_2 to the user H_L in all configurations. This is the function on the view of both U_1 and U_2, i.e., a polynomial-time computable function applied to indistinguishable views, hence we obtain $\mathit{view}_{\mathit{conf}_1^1,k}(\mathsf{H}_L) \approx \mathit{view}_{\mathit{conf}_1^2,k}(\mathsf{H}_L)$ and $\mathit{view}_{\mathit{conf}_2^1,k}(\mathsf{H}_L) \approx \mathit{view}_{\mathit{conf}_2^2,k}(\mathsf{H}_L)$. In the following, we abbreviate $\mathit{view}_{\mathit{conf}_j^i,k}(\mathsf{H}_L)$ by $v_k^{i,j}$ for the sake of readability. We obtain $||v_k^{1,1}, v_k^{1,2}|| \leq ||v_k^{1,1}, v_k^{2,1}|| + ||v_k^{1,2}, v_k^{2,2}|| + ||v_k^{2,1}, v_k^{2,2}||$ by the triangle inequality. In the case of perfect reactive simulatability, we have $v_k^{1,1} = v_k^{2,1}$ and $v_k^{1,2} = v_k^{2,2}$ for all k, hence $||v_k^{1,1}, v_k^{1,2}|| \leq ||v_k^{2,1}, v_k^{2,2}||$. As $\mathit{conf}_1^2, \mathit{conf}_2^2 \in \mathsf{Conf}^{\varphi}(\hat{M}_2, S, \Gamma)$, Definition 3 implies $||v_k^{2,1}, v_k^{2,2}|| \leq \mathcal{Q}^{||\cdot,\cdot||,\mathcal{M},\varphi}_{(\hat{M}_2,S,\Gamma)}(H, L)$ for all distances $||\cdot,\cdot||$, which completes the proof of the perfect case. In the computational case of reactive simulatability, we have $||v_k^{1,1}, v_k^{2,1}||_{\mathsf{c}} \in \mathit{NEGL}$ and $||v_k^{1,2}, v_k^{2,2}||_{\mathsf{c}} \in \mathit{NEGL}$ for every computational distance $||\cdot,\cdot||_{\mathsf{c}}$. As the class of negligible functions is closed under addition, $\epsilon(k) := ||v_k^{1,1}, v_k^{2,1}||_{\mathsf{c}} + ||v_k^{1,2}, v_k^{2,2}||_{\mathsf{c}}$ is a negligible function again. Now $\mathit{conf}_1^2, \mathit{conf}_2^2 \in \mathsf{Conf}^{\varphi}(\hat{M}_2, S, \Gamma)$ and Definition 3 imply that $||v_k^{2,1}, v_k^{2,2}||_{\mathsf{c}}$ is upper bounded by $\mathcal{Q}_{\mathcal{P}}{}^{||\cdot,\cdot||_{\mathsf{c}},\mathcal{M},\varphi}_{(\hat{M}_2,S,\Gamma)}(H, L)$ for any computational distance $||\cdot,\cdot||_{\mathsf{c}}$, which completes the proof of the computational case. ■

5 Relationship to Probabilistic Non-interference

In this section, we show that the worst-case information quantity is related in a natural way to the notion of probabilistic non-interference, i.e., to the absence of probabilistic information flow. More precisely, we consider the recently proposed definitions of perfect and computational probabilistic non-interference [1] in reactive systems, and we show that a structure fulfills a non-interference property for particular high and low users if and only if the worst-case information quantity between these users is zero in the case of perfect non-interference or bounded by a negligible function in the case of computational non-interference.

5.1 Brief Review of Computational Probabilistic Non-interference

We first review briefly the notions of perfect and computation non-interference in reactive systems. Information flow properties such as non-interference consist of two components: a *flow policy* and a *definition of information flow*. Flow policies specify restrictions on the information flow within a system.

Definition 6. *(Flow Policy)* Let a structure $(\hat{M}, S)$ be given, and let $\Gamma = \{S_i \mid i \in \mathcal{I}\}$ denote a partition of S for a finite index set $\mathcal{I}$. A *flow policy* $\mathcal{F}$ of the structure $(\hat{M}, S)$ is a graph $\mathcal{F} = (\Gamma, \rightsquigarrow)$ with $\rightsquigarrow \subseteq \Gamma \times \Gamma$. For $(S_i, S_j) \in \rightsquigarrow$, we write $S_i \rightsquigarrow S_j$, and $S_i \not\rightsquigarrow S_j$ otherwise. Furthermore we demand $S_i \rightsquigarrow S_i$ for all $S_i \in \Gamma$. ◇

Here $S_i \rightsquigarrow S_j$ intuitively means that information may flow from S_i to S_j, whereas $S_i \not\rightsquigarrow S_j$ means that it must not. The relation $\not\rightsquigarrow$ is the non-interference relation of $\mathcal{F}$, i.e., $S_H \not\rightsquigarrow S_L$ means that no information must flow from the user connected to the ports S_H to the user connected to the ports S_L. To capture this in a way that allows for computational restrictions, error probabilities etc., the notion of probabilistic non-interference from [1] gives the user H_H (connected to S_H) a randomly distributed bit b at the start of the run, and H_H should try to transmit this bit to H_L (connected to S_L). The user H_L then outputs a bit b^*, which is its guess of the bit b. To capture this formally in the model, the specific users have special ports for receiving the initial bit and for outputting their guess, respectively, and special machines BIT_H and OUT_L are added that produce the bit b and consume the bit b^*. As for partition configurations, the same specific fair master scheduler X is added to the configuration to prevent from achieving information flow in a trivial manner. The resulting configurations are called *non-interference configurations* for S_H and S_L. Then the underlying structure $(\hat{M}, S)$ is defined to fulfill the non-interference requirement defined by flow policy $\mathcal{F}$ in the computational sense (written $(\hat{M}, S) \models^{\mathsf{poly}} \mathcal{F}$) iff for all H, L with $S_H \not\rightsquigarrow S_L$ and all polynomial-time non-interference configurations for S_H and S_L, the probability of a correct guess $b = b^*$ is only negligibly greater than pure guessing. A structure fulfills the requirement in the perfect sense (written $(\hat{M}, S) \models^{\mathsf{perf}} \mathcal{F}$) iff the same holds for all (also non-polynomially bounded) configurations and the advantage over pure guessing should be zero. We review the rigorous definitions in Appendix A.

5.2 Linking Information Quantity and Non-interference

We now show that a structure fulfills a non-interference requirement if and only if the worst-case information quantity between the respective pairs of users defined by the flow policy is zero or a negligible function in the security parameter k, respectively.

The left-to-right direction of this statement is closely related to existing results for previous definitions of non-interference, where it has been proved that non-interference implies that the information quantity exchanged between the respective users is zero. In a cryptographic scenario, the notion of a negligibly small information quantity has replaced the total absence of information flow. When considering the converse direction however, an information quantity of zero was not sufficient to establish the non-interference property for many existing definitions of non-interference, which often made these properties too strict for dealing with information flow. For the definition of perfect and computational probabilistic non-interference in the reactive setting, we can establish this converse direction. This might serve as an indication that the reactive definition of non-interference is not overly restrictive and might constitute an important tool for reasoning about absence of information flow in the presence of cryptography.

Theorem 2. *(Information Quantity and Non-Interference)* Let a structure $(\hat{M}, S)$, a partition $\Gamma = \{S_i \mid i \in \mathcal{I}\}$ of S for a finite index set $\mathcal{I}$, and a flow policy $\mathcal{F} = (\Gamma, \rightsquigarrow)$ of $(\hat{M}, S)$ be given. Then we have $(\hat{M}, S) \models^{\mathsf{perf}} \mathcal{F}$ (resp. $(\hat{M}, S) \models^{\mathsf{poly}} \mathcal{F}$) iff for all $H, L \in \mathcal{I}$ with $S_H \not\rightsquigarrow S_L$ we have $\mathcal{Q}^{||\cdot,\cdot||}_{(\hat{M},S,\Gamma)}(H, L) = 0$ for all distances $||\cdot, \cdot||$ (resp. $\mathcal{Q}_{\mathcal{P}}{}^{||\cdot,\cdot||_{\mathsf{c}}}_{(\hat{M},S,\Gamma)}(H, L) \in \mathit{NEGL}$ for all computational distances $||\cdot, \cdot||_{\mathsf{c}}$). □

Proof. We only prove the more complicated computational case here; the perfect case can be easily derived from that. We start with the left-to-right direction. Assume for contradiction that $\mathcal{Q}_{\mathcal{P}}{}^{||\cdot,\cdot||_{\mathsf{c}}}_{(\hat{M},S,\Gamma)}(H, L) \notin \mathit{NEGL}$ for some H, L with $S_H \not\rightsquigarrow S_L$ and some computational distance $||\cdot, \cdot||_{\mathsf{c}}$. This means that there exist two partition configurations $\mathit{conf}_1 = (\hat{M}, S, U_1, \mathsf{A})$ and $\mathit{conf}_2 = (\hat{M}, S, U_2, \mathsf{A})$ from $\mathsf{Conf}_{\mathsf{poly}}(\hat{M}, S, \Gamma)$ with $U_l := \{\mathsf{H}^{(l)}_H, \mathsf{H}_L, \mathsf{X}\} \cup \{\mathsf{H}_i \mid i \in \mathcal{I} \setminus \{H, L\}\}$ such that $||\mathit{view}_{\mathit{conf}_1,k}(\mathsf{H}_L), \mathit{view}_{\mathit{conf}_2,k}(\mathsf{H}_L)||_{\mathsf{c}} = \mathcal{Q}_{\mathcal{P}}{}^{||\cdot,\cdot||_{\mathsf{c}}}_{(\hat{M},S,\Gamma)}(H, L)$. This implies $\mathit{view}_{\mathit{conf}_1,k}(\mathsf{H}_L) \not\approx \mathit{view}_{\mathit{conf}_2,k}(\mathsf{H}_L)$, i.e., there exists a probabilistic polynomial-time distinguisher Dis such that $|P(\mathsf{Dis}(1^k, \mathit{view}_{\mathit{conf}_1,k}(\mathsf{H}_L)) = 1) - P(\mathsf{Dis}(1^k, \mathit{view}_{\mathit{conf}_2,k}(\mathsf{H}_L)) = 1)| = n(k)$ for a non-negligible function n. We now define a non-interference configuration conf that contradicts $(\hat{M}, S) \models^{\mathsf{poly}} \mathcal{F}$. If the high user of conf receives $b = 0$, it acts as $\mathsf{H}^{(1)}_H$, and as $\mathsf{H}^{(2)}_H$ otherwise. The low user of conf act as H_L but when H_L would enter final state, the low user uses Dis as a blackbox submachine, runs it on H_L's view, and outputs the bit that Dis outputs. The low user is polynomial-time because both H_L and Dis are polynomial-time. The remaining users of conf act as in configuration conf_1 and conf_2. By the construction of conf, the probability of a correct guess

$b = b^*$ of the low user in *conf* is equal to $n(k)$, which yields a contradiction to $(\hat{M}, S) \models^{\mathsf{poly}} \mathcal{F}$.

We now prove the right-to-left direction. Assume for contradiction that $(\hat{M}, S) \not\models^{\mathsf{poly}} \mathcal{F}$. Then there exist H, L with $S_H \not\rightsquigarrow S_L$ and a non-interference configuration *conf* for S_H, S_L such that the probability of a correct guess of the low user in *conf* is equal to $\frac{1}{2} + n(k)$ for some non-negligible function n. We now define two partition configurations $conf_1$, $conf_2$ as follows. The user $\mathsf{H}_H^{(1)}$ in $conf_1$ acts as H_H would if it received $b = 0$, and $\mathsf{H}_H^{(2)}$ in $conf_2$ acts as H_H would if it received $b = 1$. The user H_L in $conf_1$ and $conf_2$ acts as the low user in *conf* but instead of outputting the bit b^* to the now non-existing machine OUT_L, it simply stores b^* (to keep it part of its view). The remaining users act as in *conf*. Now, as n is a non-negligible function, we immediately obtain $view_{conf_1,k}(\mathsf{H}_L) \not\approx view_{conf_2,k}(\mathsf{H}_L)$ by construction of $conf_1$ and $conf_2$, and hence $||view_{conf_1,k}(\mathsf{H}_L), view_{conf_2,k}(\mathsf{H}_L)||_{\mathsf{c}} \notin NEGL$ for every computational distance $||\cdot,\cdot||_{\mathsf{c}}$. This yields $\mathcal{Q}_{\mathcal{P}}^{||\cdot,\cdot||_{\mathsf{c}}}{}_{(\hat{M},S,\Gamma)}(H, L) \geq ||view_{conf_1,k}(\mathsf{H}_L), view_{conf_2,k}(\mathsf{H}_L)||_{\mathsf{c}} \notin NEGL$ and hence the desired contradiction. ∎

The key property proved about the notions of perfect and computation probabilistic non-interference in [1] is that they are preserved under reactive simulatability, i.e., if $(\hat{M}_2, S)$ fulfills a non-interference requirement and $(\hat{M}_1, S)$ is at least as secure as $(\hat{M}_2, S)$, then $(\hat{M}_1, S)$ also fulfills this non-interference requirement. Using the results of Theorem 2 and Theorem 1, we can give a very short alternative proof.

Corollary 1. *(Preservation of Perfect/Computation Probabilistic Non-interference [1], Sketch)* Let structures $(\hat{M}_1, S)$, $(\hat{M}_2, S)$ be given such that $(\hat{M}_1, S) \geq_{\mathsf{sec}}^{x} (\hat{M}_2, S)$ for $x \in \{\mathsf{poly}, \mathsf{perf}\}$. Then $(\hat{M}_2, S) \models^{x} \mathcal{F}$ for a flow policy $\mathcal{F}$ implies $(\hat{M}_1, S) \models^{x} \mathcal{F}$. □

Proof. Theorem 2 and $(\hat{M}_2, S) \models^{\mathsf{poly}} \mathcal{F}$ imply $\mathcal{Q}_{\mathcal{P}}^{||\cdot,\cdot||_{\mathsf{c}}}{}_{\hat{M}_2,S,\Gamma}(H, L) \in NEGL$ for all H, L with $S_H \not\rightsquigarrow S_L$ and all computational distances $||\cdot,\cdot||_{\mathsf{c}}$. Theorem 1 implies $\mathcal{Q}_{\mathcal{P}}^{||\cdot,\cdot||_{\mathsf{c}}}{}_{\hat{M}_1,S,\Gamma}(H, L) \leq \mathcal{Q}_{\mathcal{P}}^{||\cdot,\cdot||_{\mathsf{c}}}{}_{\hat{M}_2,S,\Gamma}(H, L) + \epsilon(k)$ for some $\epsilon \in NEGL$. Hence $\mathcal{Q}_{\mathcal{P}}^{||\cdot,\cdot||_{\mathsf{c}}}{}_{\hat{M}_1,S,\Gamma}(H, L) \in NEGL$ because the class of negligible function is closed under addition. Theorem 2 then yields $(\hat{M}_1, S) \models^{\mathsf{poly}} \mathcal{F}$. The perfect case is proved by replacing $\mathcal{Q}_{\mathcal{P}}$ with $\mathcal{Q}$ and by considering arbitrary distances. ∎

6 Conclusion

We have presented the first definitions for quantifying information flow within a reactive setting. The definitions comprise unconditional as well as complexity-theoretic aspects of security and are hence suited for reasoning about information flow even in the presence of cryptography. We have shown that our definitions are preserved under simulatability which constitutes the cryptographic notion of a

secure implementation. This significantly simplifies to determine the information flow quantity for cryptographic system since simulatability helps to eliminate cryptography-related details such as error probabilities and computational restrictions; hence we can hope to exploit existing non-cryptographic techniques for this task. We have linked our definitions to existing non-interference definitions by showing that a zero or negligible quantity of information flow is equivalent to perfect and computational probabilistic non-interference. With our simulatability preservation theorem, this has in particular allowed for a short, alternative proof that non-interference properties are preserved under simulatability.

References

1. 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.
2. M. Backes and B. Pfitzmann. Intransitive non-interference for cryptographic purposes. In *Proc. 24th IEEE Symposium on Security & Privacy*, pages 140–152, 2003.
3. M. Backes, B. Pfitzmann, and M. Waidner. Secure asynchronous reactive systems. IACR Cryptology ePrint Archive 2004/082, Mar. 2004.
4. D. Bell and L. LaPadula. Secure computer systems: Unified exposition and multics interpretation. Computer Science Technical Report ESD-TR-75-306, The Mitre Corporation, 1976.
5. D. Clark, S. Hunt, and P. Malacaria. Quantitative analysis of the leakage of confidential data. In *Proc. Quantitative Aspects of Programming Languages*, volume 59 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 2002.
6. D. Denning. *Cryptography and Data Security*. Addison-Wesley, 1982.
7. D. E. Denning. A lattice model of secure information flow. *Communications of the ACM*, 19(5):236–243, 1976.
8. A. Di Pierro, C. Hankin, and H. Wiklicky. Approximate non-interference. In *Proc. 15th IEEE Computer Security Foundations Workshop (CSFW)*, pages 1–17, 2002.
9. R. Focardi and R. Gorrieri. The compositional security checker: A tool for the verification of information flow security properties. *IEEE Transactions on Software Engineering*, 23(9):550–571, 1997.
10. 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.
11. J. A. Goguen and J. Meseguer. Security policies and security models. In *Proc. 3rd IEEE Symposium on Security & Privacy*, pages 11–20, 1982.
12. J. A. Goguen and J. Meseguer. Unwinding and inference control. In *Proc. 5th IEEE Symposium on Security & Privacy*, pages 75–86, 1984.
13. J. W. Gray III. Probabilistic interference. In *Proc. 11th IEEE Symposium on Security & Privacy*, pages 170–179, 1990.
14. J. W. Gray III. Toward a mathematical foundation for information flow security. *Journal of Computer Security*, 1(3):255–295, 1992.
15. C. A. R. Hoare. *Communicating Sequential Processes*. International Series in Computer Science, Prentice Hall, Hemel Hempstead, 1985.
16. 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.

17. B. W. Lampson. A note on the confinement problem. *Communications of the ACM*, 16(10):613–615, 1973.
18. P. Laud. Semantics and program analysis of computationally secure information flow. In *Proc. 10th European Symposium on Programming (ESOP)*, pages 77–91, 2001.
19. P. Laud. Symmetric encryption in automatic analyses for confidentiality against active adversaries. In *Proc. 25th IEEE Symposium on Security & Privacy*, pages 71–85, 2004.
20. G. Lowe. Quantifying information flow. In *Proc. 15th IEEE Computer Security Foundations Workshop (CSFW)*, pages 18–31, 2002.
21. N. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, San Francisco, 1996.
22. 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.
23. H. Mantel and A. Sabelfeld. A generic approach to the security of multi-threaded programs. In *Proc. 14th IEEE Computer Security Foundations Workshop (CSFW)*, pages 200–214, 2001.
24. D. McCullough. Specifications for multi-level security and a hook-up property. In *Proc. 8th IEEE Symposium on Security & Privacy*, pages 161–166, 1987.
25. J. McLean. Security models and information flow. In *Proc. 11th IEEE Symposium on Security & Privacy*, pages 180–187, 1990.
26. J. McLean. Security models. Chapter in *Encyclopedia of Software Engineering*, 1994.
27. J. K. Millen. Covert channel capacity. In *Proc. 8th IEEE Symposium on Security & Privacy*, pages 60–66, 1987.
28. A. Myers and B. Liskov. A decentralized model for information flow control. In *Proc. ACM Symposium on Operating System Principles*, pages 129–142, 1997.
29. 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. Extended version of the model (with Michael Backes) IACR Cryptology ePrint Archive 2004/082, `http://eprint.iacr.org/`.
30. R. G. Riccardo Focardi, Anna Ghelli. Using non-interference for the analysis of security protocols. In *Proc. DIMACS Workshop on Design and Formal Verification of Security Protocols*, 1997.
31. A. Sabelfeld and D. Sands. A per model of secure information flow in sequential programs. In *Proc. European Symposium on Programming (ESOP)*, pages 40–58. Springer, 1999.
32. A. Sabelfeld and D. Sands. Probabilistic noninterference for multi-threaded programs. In *Proc. 13th IEEE Computer Security Foundations Workshop (CSFW)*, pages 200–214, 2000.
33. D. Sutherland. A model of information. In *Proc. 9th National Computer Security Conference*, pages 175–183, 1986.
34. D. Volpano. Secure introduction of one-way functions. In *Proc. 13th IEEE Computer Security Foundations Workshop (CSFW)*, pages 246–254, 2000.
35. D. Volpano and G. Smith. Probabilistic noninterference in a concurrent language. In *Proc. 11th IEEE Computer Security Foundations Workshop (CSFW)*, pages 34–43, 1998.
36. D. Volpano, G. Smith, and C. Irvine. A sound type system for secure flow analysis. *Journal of Computer Security*, 4(3):167–187, 1996.

37. J. T. Wittbold and D. M. Johnson. Information flow in nondeterministic systems. In *Proc. 11th IEEE Symposium on Security & Privacy*, pages 144–161, 1990.
38. A. C. Yao. Theory and applications of trapdoor functions. In *Proc. 23rd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 80–91, 1982.
39. S. Zdancewic and A. C. Myers. Secure information flow and CPS. In *Proc. 10th European Symposium on Programming (ESOP)*, volume 2028 of *Lecture Notes in Computer Science*, pages 46–61. Springer, 2001.

A The Model of Reactive Simulatability

In this section we give a more comprehensive review of the model of reactive simulatability [29,3] for the sake of completeness.

A.1 General System Model

Communication between different machines is done via ports. Inspired by the CSP-notation [15], we write input and output ports as $\mathsf{p}?$ and $\mathsf{p}!$, respectively. The input and output ports in a port set P are written $\mathsf{in}(P)$ and $\mathsf{out}(P)$, respectively. Connections are defined by naming convention: port $\mathsf{p}!$ sends messages to $\mathsf{p}?$. To achieve asynchronous timing, a message is not immediately delivered to its recipient, but is first stored in a special machine $\widetilde{\mathsf{p}}$ called a buffer, where it waits to be scheduled. This can be done by the machine with the unique clock-out port $\mathsf{p}^{\triangleleft}!$. To schedule the i-th message of buffer $\widetilde{\mathsf{p}}$, it outputs i at $\mathsf{p}^{\triangleleft}!$, see Figure 2. The buffer then delivers the i-th message and removes it from its internal list. Most buffers are scheduled either by a specific master scheduler or by the adversary, i.e., one of those has the clock-out port. Ports $\mathsf{p}!$ and $\mathsf{p}?$, in contrast to the other four port types occurring at the buffers, are called *simple*, and a *simple machine* has only simple ports and clock-out ports.

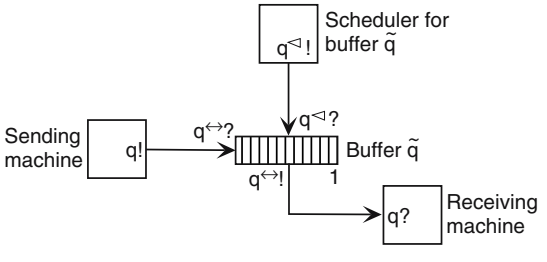

Fig. 2. Naming of ports around a buffer. Later one can often abstract from the buffer and simply regard $\mathsf{q}!$ and $\mathsf{q}?$ as asynchronously connected.

The precise definition of machines is a variant of probabilistic state-transition machines, similar to probabilistic I/O automata as sketched by Lynch [21]. If a machine is switched, it reads an input tuple at its input ports and performs its transition function. This yields a new state and an output tuple. A probabilistic

transition function actually describes a finite distribution over the pairs of a new state and an output tuple. Furthermore, each machine has bounds on the length of considered inputs. This allows time bounds independent of the environment.

Definition 7. *(Machines)* A *machine* (for an alphabet Σ) is a tuple

$$\mathsf{M} = (name_{\mathsf{M}}, Ports_{\mathsf{M}}, States_{\mathsf{M}}, \delta_{\mathsf{M}}, l_{\mathsf{M}}, Ini_{\mathsf{M}}, Fin_{\mathsf{M}})$$

of a machine name $name_{\mathsf{M}} \in \Sigma^+$, a finite sequence $Ports_{\mathsf{M}}$ of ports, a set $States_{\mathsf{M}} \subseteq \Sigma^*$ of states, a probabilistic state-transition function δ_{M}, a length function $l_{\mathsf{M}} : States_{\mathsf{M}} \to (\mathbb{N} \cup \{\infty\})^{|\mathsf{in}(Ports_{\mathsf{M}})|}$, and sets $Ini_{\mathsf{M}}, Fin_{\mathsf{M}} \subseteq States_{\mathsf{M}}$ of initial and final states. Its input set is $\mathcal{I}_{\mathsf{M}} := (\Sigma^*)^{|\mathsf{in}(Ports_{\mathsf{M}})|}$; the i-th element of an input tuple denotes the input at the i-th in-port. Its output set is $\mathcal{O}_{\mathsf{M}} := (\Sigma^*)^{|\mathsf{out}(Ports_{\mathsf{M}})|}$. The empty word, ϵ, denotes no in- or output at a port. δ_{M} maps each pair $(s, I) \in States_{\mathsf{M}} \times \mathcal{I}_{\mathsf{M}}$ to a finite distribution over $States_{\mathsf{M}} \times \mathcal{O}_{\mathsf{M}}$.

If $s \in Fin_{\mathsf{M}}$ or $I = (\epsilon, \ldots, \epsilon)$, then $\delta_{\mathsf{M}}(s, I) = (s, (\epsilon, \ldots, \epsilon))$ deterministically. Inputs are ignored beyond the length bounds, i.e., $\delta_{\mathsf{M}}(s, I) = \delta_{\mathsf{M}}(s, I\lceil_{l_{\mathsf{M}}(s)})$ for all $I \in \mathcal{I}_{\mathsf{M}}$, where $(I\lceil_{l_{\mathsf{M}}(s)})_i := I_i\lceil_{l_{\mathsf{M}}(s)_i}$ for all i. ◇

In the text, we often write "M" for $name_{\mathsf{M}}$ as well. In the following, the initial states of all machines are a security parameter $k \in \mathbb{N}$ in unary representation. In order to define the notion of polynomial runtime for these machines, Turing machine realizations of them are defined so that the runtime can be measured in the size of the initial worktape content (typically a security parameter in unary representation).

A *collection* $\hat{C}$ of machines is a finite set of machines with pairwise different machine names and disjoint sets of ports. All machines start with the same security parameter k. Let furthermore $\mathsf{ports}(\hat{C})$ denote the set of all ports of all machines in $\hat{C}$. The *completion* $[\hat{C}]$ of a collection $\hat{C}$ consists of all machines of $\hat{C}$ and the buffers needed for all the ports in $\hat{C}$. The *free* ports $\mathsf{free}(\hat{C})$ in a collection are those to which no other port in the collection connects. A collection $\hat{C}$ is *closed* if its completion $[\hat{C}]$ has no free ports except a special master clock-in port $\mathsf{clk}^{\triangleleft}?$. The machine with this port is the *master scheduler*, to which control returns as a default.

For a closed collection, a probability space of *runs* (sometimes called traces or executions) is defined. The machines switch sequentially, i.e., there is exactly one active machine M at any time, called the *current scheduler*. 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 recipient is the next active machine. If M attempts to schedule multiple messages, only one is taken. If it schedules none or the message does not exist, the master scheduler is activated. Formally, runs are sequences of *steps* defined as follows (where the state-transition function of buffers is as explained above).

Definition 8. *(Runs)* Given a closed collection $\hat{C}$ with master scheduler X and a security parameter k, the probability space of *runs* is defined inductively by the following algorithm. It has variables r for the run under construction and

$\mathsf{M_{CS}}$ for the current scheduler, and treats each port as a variable over Σ^*. Here, r is an initially empty list, $\mathsf{M_{CS}}$ a machine name initialized with X, and all port variables are initially ϵ except for $\mathsf{clk}^{\triangleleft}? := 1$. Probabilistic choices only occur in Phase 1.

1. *Switch current scheduler:* Switch machine $\mathsf{M_{CS}}$, i.e., set $(s', O) \leftarrow \delta_{\mathsf{M_{CS}}}(s, I)$ for its current state s and in-port values I. Then assign ϵ to all in-ports of $\mathsf{M_{CS}}$.
2. *Termination:* If X is in a final state, the run stops.
3. *Buffer new messages:* For each simple out-port $\mathsf{q}!$ of $\mathsf{M_{CS}}$, switch buffer $\widetilde{\mathsf{q}}$ with input $\mathsf{q}^{\leftrightarrow}? := \mathsf{q}!$, cf. Figure 2. Then assign ϵ to all these ports $\mathsf{q}!$ and $\mathsf{q}^{\leftrightarrow}?$.
4. *Clean up scheduling:* If at least one clock out-port of $\mathsf{M_{CS}}$ has a value $\neq \epsilon$, let $\mathsf{q}^{\triangleleft}!$ denote the first such port and assign ϵ to the others. Otherwise let $\mathsf{clk}^{\triangleleft}? := 1$ and $\mathsf{M_{CS}} := \mathsf{X}$ and go back to Phase 1.
5. *Deliver scheduled message:* Switch buffer $\widetilde{\mathsf{q}}$ with input $\mathsf{q}^{\triangleleft}? := \mathsf{q}^{\triangleleft}!$ (see Figure 2), set $\mathsf{q}? := \mathsf{q}^{\leftrightarrow}!$ and then assign ϵ to all ports of $\widetilde{\mathsf{q}}$ and to $\mathsf{q}^{\triangleleft}!$. Let $\mathsf{M_{CS}} := \mathsf{M}'$ for the unique machine M' with $\mathsf{q}? \in \mathsf{ports}(\mathsf{M}')$. Go back to Phase 1.

Whenever a machine (this may be a buffer) with name $name_{\mathsf{M}}$ is switched from (s, I) to (s', O), we append a *step* $(name_{\mathsf{M}}, s, I', s', O)$ to the run r for $I' := I\lceil_{l_{\mathsf{M}}(s)}$, except if s is final or $I' = (\epsilon, \ldots, \epsilon)$. This gives a family of random variables

$$run_{\hat{C}} = (run_{\hat{C},k})_{k \in \mathbb{N}}.$$

For a number $l \in \mathbb{N}$, the l-step *prefix* of a run r is the list of the first l steps. $\diamond$

Next we define what a machine sees in a run and what events happen at a set of ports, and the probabilities of such views and events.

Definition 9. *(Views and Restrictions to Ports)* The *view* of a set of machines $\hat{M}$ in a run r is the subsequence of all steps $(name_{\mathsf{M}}, s, I, s', O)$ where $name_{\mathsf{M}}$ is the name of a machine $\mathsf{M} \in \hat{M}$. The *restriction* $r\lceil_S$ of a run to a set S of ports is a sequence derived as follows: First only retain the inputs and outputs, (I, O), from each step, and further restrict I and O to the ports in S. Then delete pairs where both I and O have become empty.

The corresponding families of random variables (in the probability space over the runs) are denoted by

$$view_{\hat{C}}(\hat{M}) = (view_{\hat{C},k}(\hat{M}))_{k \in \mathbb{N}} \text{ and}$$

$$run_{\hat{C}}\lceil_S = (run_{\hat{C},k}\lceil_S)_{k \in \mathbb{N}}.$$

With an additional index l, we denote the $l(k)$-step prefixes of the views and restrictions. $\diamond$

For a one-element set $\hat{M} = \{\mathsf{H}\}$ we write $view_{\hat{C}}(\mathsf{H})$ for $view_{\hat{C}}(\{\mathsf{H}\})$.

A.2 Security-Specific System Model

For security purposes, we have to define how adversaries and honest users connect to specified machines of a collection. First, an adversary may take over parts of the initially intended machines. These machines are then absorbed into the adversary, and the remaining machines form a *structure*. Formally, a structure is a collection of machines in which one additionally distinguishes which free ports honest users can connect to and expect some reasonable service (e.g., message transport in a cryptographic firewall), and which ports are only used by adversaries. The former are the *service ports* in the following definition. Valid honest users should neither try to connect to the remaining free ports of a structure, nor, for unique naming, have ports that already occur inside the structure. This is expressed by *forbidden ports*. The ports connecting to a given port set P are expressed by the complement notation P^c, e.g., $\mathsf{q}!^c = \mathsf{q}^{\leftrightarrow}?$, $\mathsf{q}^{\triangleleft}!^c = \mathsf{q}^{\triangleleft}?$, $\mathsf{q}^{\leftrightarrow}!^c = \mathsf{q}?$ in Figure 2, and vice versa.

Definition 10. *(Structures)* A *structure* is a pair $(\hat{M}, S)$ where $\hat{M}$ is a collection of simple machines called *correct machines*, and $S \subseteq \mathsf{free}([\hat{M}])$ is called *service ports*. If $\hat{M}$ is clear from the context, let $\bar{S} := \mathsf{free}([\hat{M}]) \setminus S$. We call $\mathsf{forb}(\hat{M}, S) := \mathsf{ports}(\hat{M}) \cup \bar{S}^c$ the *forbidden ports*. ◇

A structure is completed to a (multi-party) *configuration* by a set of machines U modeling the honest users, and by a machine A modeling the adversary. As explained above, the machines in U do not have certain ports. A connects to the remaining free ports of the structure.

Definition 11. *((Multi-party) Configurations)*

a) A (multi-party) *configuration* of a structure $(\hat{M}, S)$ is a tuple $conf = (\hat{M}, S, U, \mathsf{A})$, where U is a set of simple machines without forbidden ports, i.e., $\mathsf{ports}(U) \cap \mathsf{forb}(\hat{M}, S) = \emptyset$, and $\hat{C} := \hat{M} \cup U \cup \{\mathsf{A}\}$ is a closed collection. For simplicity, we often write run_{conf} and $view_{conf}(\hat{M})$ instead of $run_{\hat{C}}$ and $view_{\hat{C}}(\hat{M})$

b) The set of (multi-party) configurations is written $\mathsf{Conf}(\hat{M}, S)$. The subset of configurations with polynomial-time users and a polynomial-time adversary is called $\mathsf{Conf}_{\mathsf{poly}}(\hat{M}, S)$. The index $_{\mathsf{poly}}$ is omitted if it is clear from the context. ◇

Partition and non-interference configurations can now be defined by considering only those users that have a specific set of ports so that they connect exactly to the ports of the structure prescribed by the considered partition. We omit the formal yet lengthy definitions and refer to [1].

A.3 Definition of Non-interference in the Model

We finally give a precise definition of perfect and computational probabilistic non-interference in the reactive model, i.e., the formal semantics of the

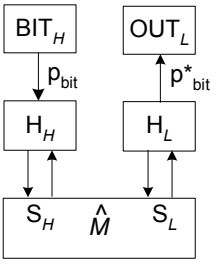

Fig. 3. Sketch of the non-interference definition: H_L attempts to guess a bit that H_H is attempting to transfer

$\not\rightsquigarrow$ relation. Usually, expressing this semantics is the most difficult part of an information-flow definition. Given our underlying model, it is somewhat easier because we already have definitions of runs, views, and indistinguishability. Based on these definitions, we can define the probability that the low user correctly guesses the bit that the high user attempts to transmit.

Definition 12. *(Guessing Probability)* For a non-interference configuration $conf \in \mathsf{Conf}(\hat{M}, S, \Gamma)$ for $S_H, S_L \in \Gamma$ of a structure $(\hat{M}, S)$, the *guessing probability* $P_{\mathsf{guess},conf}$ is defined as

$$P_{\mathsf{guess},conf} := P(b = b^* \mid r \leftarrow run_{conf,k}; b := r\lceil_{\mathsf{p_{bit}}!}; b^* := r\lceil_{\mathsf{p^*_{bit}}?}),$$

with the ports $\mathsf{p_{bit}}!$ and $\mathsf{p^*_{bit}}?$ defined as in Figure 3. This is a function of the security parameter k. ◇

Now we are ready to give the non-interference definition, i.e., the definition of the semantics of a flow policy for a reactive setting.

Definition 13. *(Non-Interference)* Let a structure $(\hat{M}, S) \in Sys$, and a flow policy $\mathcal{F} = (\Gamma, \rightsquigarrow)$, $\Gamma = \{S_i \mid i \in \mathcal{I}\}$ be given. We say that $(\hat{M}, S)$ *fulfills* the non-interference requirement defined by the flow policy $\mathcal{F}$

a) **perfectly**, written $(\hat{M}, S) \models^{\mathsf{perf}} \mathcal{F}$, iff for every H, L with $S_H \not\rightsquigarrow S_L$ and every non-interference configuration $conf \in \mathsf{Conf}(\hat{M}, S, \Gamma)$ for S_H and S_L, we have
$$P_{\mathsf{guess},conf} \leq \frac{1}{2}.$$

c) **computationally**, written $(\hat{M}, S) \models^{\mathsf{poly}} \mathcal{F}$, iff for every H, L with $S_H \not\rightsquigarrow S_L$ and every polynomial-time non-interference configuration $conf \in \mathsf{Conf}_{\mathsf{poly}}(\hat{M}, S, \Gamma)$ for S_H and S_L there exists a function $\epsilon \in NEGL$ such that
$$P_{\mathsf{guess},conf} \leq \frac{1}{2} + \epsilon(k).$$

◇