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 [39] and subsequently by Bell and LaPadula [23] and Denning [27]. Initiated by the work on non-interference of Goguen and Meseguer [32, 33], various definitions have subsequently been proposed that rigorously specify when information flow is considered to occur for possibilistic and non-deterministic systems [61, 47, 65, 51, 58, 31, 44, 46] and for probabilistic systems [34, 35, 49, 63, 60]. 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 [54, 38, 42, 24, 28].

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 techniques. 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 [10, 9], itself inspired from new computational probabilistic notions for fairness and liveness [15], 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 [10].

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 [10, 9] based on the model of reactive simulatability [57, 20]; 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. [28]. They defined the notion of \( \varepsilon \)-confinement that captured that information flow is still acceptable if the distance of views of specific user deviate only up to probability \( \varepsilon \). Although their definition does not address computational aspects as needed for cryptographic purposes, our work is nevertheless inspired by some of their ideas. Lowe [42] 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. [24] 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 [26], which was subsequently used in [54, 38] 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 [40, 41] 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 [62] 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. Abadi and Blanchet [1] introduced type systems where asymmetric communication primitives, especially public-key encryption, can be handled, but these primitives are only relative to a Dolev-Yao abstraction [29], i.e., the primitives are idealized so that no computational non-interference definition is needed. The computational soundness of Dolev-Yao abstraction is currently still subject to intensive research, see, e.g., [56, 2, 8, 17, 7, 11, 16, 22, 19, 3, 12, 53, 14, 13, 5]. Finally, whenever new notion of non-interference are suggested, special care must be taken concerning their composition, since obtaining security under composition is known to be a highly sophisticated task, see e.g., [47, 37, 48, 50, 52, 45, 25, 21, 4, 18, 6, 19, 12].

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 [57, 20], which is an asynchronous probabilis-
tic 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 Conf\((\hat{M}, S)\) and the subset of polynomial-time configurations by
Conf\(_{poly}(\hat{M}, S)\).\(^1\)

The general scheduling model in [57, 20] gives each connection \( c \) (from an output port \( c! \) to an input
port \( c? \)) a buffer, and the machine with the corresponding clock port \( c\triangleright \) 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. [20] for the precise definition. We call the corresponding
random variable \( \text{run}_{\text{conf}, k} \) for a configuration \( \text{conf} \) 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 \( \text{conf} \), the corresponding random variables over
the probability space of all possible runs are denoted by \( \text{view}_{\text{conf}, k}(M) \) and \( \text{view}_{\text{conf}, k}(\hat{M}) \), respectively.

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

\(^1\) Here and elsewhere we change some notation of [57, 20] 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.
an arbitrary adversary \( A \) and (\( M, S \), \( \Gamma \)). Furthermore, let \( \mathcal{I} \) be 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. \([10, 9]\) for more details. We denote the set of partition configurations of a structure \((M, S)\) and partition \( \Gamma \) by \( \text{Conf}(M, S, \Gamma) \) and the subset of polynomial-time ones by \( \text{Conf}_\text{poly}(M, S, \Gamma) \). 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 \( \varphi : \mathcal{I} \to (\mathbb{N} \to \mathbb{N} \cup \{\infty\}) \), 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 \( \text{Conf}_\text{poly}^T(M, S, \Gamma) \) and the set of polynomial ones \( \text{Conf}_\text{poly}(M, S, \Gamma) \).

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 \([28]\).

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 \((M, S)\) be a structure, let \( \mathcal{I} = \{S_i \mid i \in \mathcal{I}\} \) be a partition on the set \( S \) of service ports for a finite set \( \mathcal{I} \), and let \(|·|, ||·||\) 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 \( Q_{(M, S, \Gamma)}^{||·||}(H, L) \) that the high user \( H_H \) is allowed to transmit to \( H_L \) is defined as

\[
Q_{(M, S, \Gamma)}^{||·||}(H, L) := \max_{\text{conf}_1, \text{conf}_2 \in \text{Conf}(M, S, \Gamma)} ||\text{view}_1, \text{conf}_1, i(I(H_L), \text{view}_2, \text{conf}_2, i(H_L))||,
\]

such that \( \text{conf}_i \) is of the form \( \text{conf}_i := (M, S, U_i, A) \) with \( U_i = \{H_H^1, H_L, X\} \cup \{H_i \mid i \in \mathcal{I} \setminus \{H, L\}\} \) for an arbitrary adversary \( A \) and arbitrary users \( H_H^1, H_H^2, H_L \), and \( H_i \) for \( i \in \mathcal{I} \setminus \{H, L\}\). The polynomial-time worst-case information quantity \( Q_{(M, S, \Gamma)}^{||·||}(H, L) \) is defined similarly by taking the maximum over \( \text{Conf}_\text{poly}(M, S, \Gamma) \).

\( \diamond \)
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 \( J \subseteq I \), i.e., we parameterize the information quantity by a set \( \mathcal{M} := \{ H_j \mid j \in J \} \). 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 \( \mathcal{M} := \{ H_j \mid j \in J \} \) be given for fixed machines \( H_j \) and \( J \subseteq I \). Then the generalized information quantity \( Q_{(\mathcal{M}, S, I)}^{\varphi}(H, L) \) with respect to \( \mathcal{M} \) is defined as

\[
Q_{(\mathcal{M}, S, I)}^{\varphi}(H, L) := \max_{\text{conf}_1, \text{conf}_2 \in \text{Conf}(\mathcal{M}, S, I)} ||\text{view}_{\text{conf}_1, k}(H_L), \text{view}_{\text{conf}_2, k}(H_L)||,
\]

such that \( \text{conf}_i \) is of the form \( \text{conf}_i := (\hat{M}, S, U_i, \hat{A}) \) with \( U_i = \{ H^{(1)}_j, H^{(2)}_j, X \} \cup \{ H_i \mid i \in I \setminus \{ H, L \} \} \) for an arbitrary adversary \( \hat{A} \) and arbitrary users \( H^{(1)}_j, H^{(2)}_j, H_L, \) and \( H_i \) for \( i \in I \setminus (J \cup \{ H, L \}) \), i.e., the maximum is only taken over those configurations in which the users \( H_j \) for \( j \in J \) are fixed by the parameter \( \mathcal{M} \). The polynomial-time variant \( Q_{(\mathcal{M}, S, I)}^{\varphi}(H, L) \) is defined as usual. \( \diamond \)

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 \( \varphi : I \to (\mathbb{N} \to \mathbb{N} \cup \{ \infty \}) \) be given. Then the timed generalized information quantity \( Q_{(\mathcal{M}, S, I)}^{\varphi}(H, L) \) with respect to \( \mathcal{M}, \varphi \) is defined as in Definition 2 except that the maximum is only taken over \( \text{Conf}^\varphi(\hat{M}, S, I) \). The polynomial-time variant \( Q_{(\mathcal{M}, S, I)}^{\varphi}(H, L) \) is defined as usual. \( \diamond \)

## 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 \( A_U \), there exists a polynomial-time adversary \( A'_U \) 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 [66].

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

\[
| P(\text{Dis}(1^k, \text{var}_k) = 1) - P(\text{Dis}(1^k, \text{var}'_k) = 1) | \in \text{NEGL},
\]

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

Intuitively, given the security parameter and an element chosen according to either \( \text{var}_k \) or \( \text{var}'_k \), \( \text{Dis} \) 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 \((\hat{M}_1, S) \geq_{\text{sec}} (\hat{M}_2, S)\) if for every configuration \( \text{conf}_1 =\)
A perfect reactive simulatability  

We speak of computational distances if and only if their distance constitutes a negligible function. We call such configurations of the respective structures, and with indistinguishability replaced by equality.

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 amount of 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 [10] 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., [64, 30, 55, 63, 59, 60, 67], 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 $(\hat{M}_1, S)$ 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 $\mathcal{M} := \{ H_j \mid j \in \mathcal{J} \}$ for some $\mathcal{J} \subseteq \mathcal{I}$ and $\varphi : \mathcal{I} \to (\mathbb{N} \to \mathbb{N} \cup \{ \infty \})$ arbitrary. Then $(\hat{M}_1, S) \geq_{\text{perf}}^\text{sec} (\hat{M}_2, S)$ implies

$$ Q_\text{perf}^{\|\cdot\|_{\hat{M}, \varphi}}(H, L) \leq Q_\text{sec}^{\|\cdot\|_{\hat{M}, \varphi}}(H, L) $$

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

$$ Q_\text{sec}^{\|\cdot\|_{\hat{M}, \varphi}}(H, L) \leq Q_\text{perf}^{\|\cdot\|_{\hat{M}, \varphi}}(H, L) + \epsilon(k) $$

for some $\epsilon \in \text{NEGL}$ and every computational distance $\|\cdot\|_\epsilon$. \hfill \Box
Proof. Let \( \text{conf}_1 = (\hat{M}_1, S, U_1, A) \) and \( \text{conf}_2 = (\hat{M}_1, S, U_2, A) \) be two (polynomial-time) partition configurations in \( \text{Conf}_r^P((M_1, S, \Gamma)) \) with \( U_i := \{ \text{H}^{(i)}_H, \text{H}_L, X \} \cup \{ \text{H}_i : i \in I \setminus \{ H, L \} \} \) (for arbitrary machines \( \text{H}_i \) for \( i \in I \setminus \{ J \cup \{ H, L \} \} \)) such that \( ||(\text{view}_{\text{conf}_1}, k, \text{H}_L), (\text{view}_{\text{conf}_2}, k, \text{H}_L)|| \) is equal to \( Q^{||\cdot||, ||:\cdot\cdot\cdot||}(\hat{M}_1, S, \Gamma) \) or to \( Q_P^{||\cdot||, ||:\cdot\cdot\cdot||}(H, L) \), respectively, in the polynomial case. Owing to \((M_1, S) \geq \text{sec}(\hat{M}_2, S)\), (polynomial-time) configurations \( \text{conf}_1^2, \text{conf}_2^2 \in \text{Conf}_r^P(\hat{M}_2, S) \) exist such that \( \text{view}_{\text{conf}_1^2}, k, (U_1) \approx \text{view}_{\text{conf}_2^2}, k, (U_1) \) and \( \text{view}_{\text{conf}_1^2}, k, (U_2) \approx \text{view}_{\text{conf}_2^2}, k, (U_2) \). Moreover, we obviously have \( \text{conf}_1 \in \text{Conf}_r^P(\hat{M}_2, S, \Gamma) \) 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 \( \text{H}_L \) 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 \( \text{view}_{\text{conf}_1^2}, k, (\text{H}_L) \approx \text{view}_{\text{conf}_2^2}, k, (\text{H}_L) \) and \( \text{view}_{\text{conf}_1^2}, k, (\text{H}_L) \approx \text{view}_{\text{conf}_2^2}, k, (\text{H}_L) \). In the following, we abbreviate \( \text{view}_{\text{conf}_1^2}, k, (\text{H}_L) \) by \( v_{i, k} \) for the sake of readability. We obtain \( ||v_{i, k}^1, v_{i, k}^2|| \leq ||v_{i, k}^1, v_{i, k}^2|| \) by the triangle inequality. In the case of perfect reactive simulatability, we have \( v_{i, k}^1 = v_{i, k}^2 \) for all \( i \), hence \( ||v_{i, k}^1, v_{i, k}^2|| \leq ||v_{i, k}^1, v_{i, k}^2|| \). As \( \text{conf}_1^2, \text{conf}_2^2 \in \text{Conf}_r^P(\hat{M}_2, S, \Gamma) \), Definition 3 implies \( ||v_{i, k}^1, v_{i, k}^2|| \leq Q^{||\cdot||, ||:\cdot\cdot\cdot||}(\hat{M}_2, S, \Gamma) \) for all distances \( ||\cdot|| \), which completes the proof of the perfect case. In the computational case of reactive simulatability, we have \( ||v_{i, k}^1, v_{i, k}^2|| \in \text{NEGL} \) and \( ||v_{i, k}^1, v_{i, k}^2|| \in \text{NEGL} \) for every computational distance \( ||\cdot|| \). As the class of negligible functions is closed under addition, \( \epsilon(k) := ||v_{i, k}^1, v_{i, k}^2|| \) is a negligible function again. Now \( \text{conf}_1^2, \text{conf}_2^2 \in \text{Conf}_r^P(\hat{M}_2, S, \Gamma) \) and Definition 3 imply that \( ||v_{i, k}^1, v_{i, k}^2|| \) is upper bounded by \( Q_P^{||\cdot||, ||:\cdot\cdot\cdot||}(H, L) \) for any computational distance \( ||\cdot|| \), 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 [10] 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 computational 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 \((M, S)\) be given, and let \( \Gamma = \{ S_i : i \in I \} \) denote a partition of \( S \) for a finite index set \( I \). A flow policy \( F \) of the structure \((M, S)\) is a graph \( F = (\Gamma, \sim) \) with \( \sim \subseteq \Gamma \times \Gamma \). For \((S_i, S_j) \in \sim\), we write \( S_i \sim S_j \), and \( S_i \not\sim S_j \) otherwise. Furthermore we demand \( S_i \sim S_i \) for all \( S_i \in \Gamma \).

Here \( S_i \sim S_j \) intuitively means that information may flow from \( S_i \) to \( S_j \), whereas \( S_i \not\sim S_j \) means that it must not. The relation \( \not\sim \) is the non-interference relation of \( F \), i.e., \( S_H \not\sim S_L \) 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 [10] gives the user \( H_H \) (connected to \( S_H \)) a randomly distributed bit \( b \) at the start of the run, and \( H_L \) 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 \( \text{BiT}_H \)
and $\text{OUT}_L$ 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 $(\hat{M}, S) \models^{\text{poly}} \mathcal{F}$) iff for all $H, L$ with $S_H \not\prec S_L$ 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 $(\hat{M}, S) \models^{\text{perf}} \mathcal{F}$) 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 $\Gamma = \{S_i \mid i \in I\}$ of $S$ for a finite index set $I$, and a flow policy $\mathcal{F} = (\Gamma, \ast)$ of $(\hat{M}, S)$ be given. Then we have $(\hat{M}, S) \models^{\text{perf}} \mathcal{F}$ (resp. $(\hat{M}, S) \models^{\text{poly}} \mathcal{F}$) iff for all $H, L \in I$ with $S_H \not\prec S_L$ we have $Q_{p_{(\hat{M}, S, \Gamma)}}^{|\cdot;|;|e|}(H, L) = 0$ for all distances $|\cdot;|;|e|$. The low user is polynomial-time because both $H$ and $L$ act as the low user in $\Gamma \ast$ and outputs the bit that $\hat{M}$ it received $b = 0$, and $\hat{M}\ast$ is only negligibly greater than pure guessing. A structure fulfills the requirement in the perfect sense (written $(\hat{M}, S) \models^{\text{perf}} \mathcal{F}$) 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.

**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 $Q_{p_{(\hat{M}, S, \Gamma)}}^{|\cdot;|;|e|}(H, L) \not\in \text{NEGL}$ for some $H, L$ with $S_H \not\prec S_L$ and some computational distance $|\cdot;|;|e|$. This means that there exist two partition configurations $\text{conf}_1 = (\hat{M}, S, U_1, A)$ and $\text{conf}_2 = (\hat{M}, S, U_2, A)$ from $\text{Conf}^{\text{poly}}(\hat{M}, S, \Gamma)$ with $U_i := \{H_i \mid i \in I \setminus \{H, L\}\}$ such that $|\text{view}_{\text{conf}_1, k}(H_L), \text{view}_{\text{conf}_2, k}(H_L)|_e = Q_{p_{(\hat{M}, S, \Gamma)}}^{|\cdot;|;|e|}(H, L)$. This implies $|\text{view}_{\text{conf}_1, k}(H_L) \neq \text{view}_{\text{conf}_2, k}(H_L)|$, i.e., there exists a probabilistic polynomial-time distinguisher $\text{Dis}$ such that $|P(\text{Dis}(1^k, \text{view}_{\text{conf}_1, k}(H_L)) = 1) - P(\text{Dis}(1^k, \text{view}_{\text{conf}_2, k}(H_L)) = 1)| = n(k)$ for a non-negligible function $n$. We now define a non-interference configuration $\text{conf}$ that contradicts $(\hat{M}, S) \models^{\text{poly}} \mathcal{F}$. If the high user of $\text{conf}$ receives $b = 0$, it acts as $H^1_{(\hat{M}, S, \Gamma)}$, and as $H^2_{(\hat{M}, S, \Gamma)}$ otherwise. The low user of $\text{conf}$ act as $H_L$ but when $H_L$ would enter final state, the low user uses $\text{Dis}$ as a blackbox submachine, runs it on $H_L$’s view, and outputs the bit that $\text{Dis}$ outputs. The low user is polynomial-time because both $H_L$ and $\text{Dis}$ are polynomial-time. The remaining users of $\text{conf}$ act as in configuration $\text{conf}_1$ and $\text{conf}_2$. By the construction of $\text{conf}$, the probability of a correct guess $b = b^*$ of the low user in $\text{conf}$ is equal to $n(k)$, which yields a contradiction to $(\hat{M}, S) \models^{\text{poly}} \mathcal{F}$.

We now prove the right-to-left direction. Assume for contradiction that $(\hat{M}, S) \not\models^{\text{poly}} \mathcal{F}$. Then there exist $H, L$ with $S_H \not\prec S_L$ and a non-interference configuration $\text{conf}$ for $S_H, S_L$ such that the probability of a correct guess of the low user in $\text{conf}$ is equal to $\frac{1}{2} + n(k)$ for some non-negligible function $n$. We now define two partition configurations $\text{conf}_1, \text{conf}_2$ as follows. The user $H^1_{(\hat{M}, S, \Gamma)}$ in $\text{conf}_1$ acts as $H_H$ if it received $b = 0$, and $H^2_{(\hat{M}, S, \Gamma)}$ in $\text{conf}_2$ acts as $H_H$ if it received $b = 1$. The user $H_L$ in $\text{conf}_1$ and $\text{conf}_2$ acts as the low user in $\text{conf}$ but instead of outputting the bit $b^*$ to the now non-existing machine $\text{OUT}_L$, it simply stores $b^*$ (to keep it part of its view). The remaining users act as in $\text{conf}$. Now, as $n$
is a non-negligible function, we immediately obtain $\text{view}_{\text{conf}_1,k}(H_L) \neq \text{view}_{\text{conf}_2,k}(H_L)$ by construction of $\text{conf}_1$ and $\text{conf}_2$, and hence $||\text{view}_{\text{conf}_1,k}(H_L), \text{view}_{\text{conf}_2,k}(H_L)||_c \notin \text{NEGL}$ for every computational distance $||\cdot, \cdot||_c$. This yields $Q_P^{||\cdot, \cdot||_c}(H, L) \geq ||\text{view}_{\text{conf}_1,k}(H_L), \text{view}_{\text{conf}_2,k}(H_L)||_c \notin \text{NEGL}$ and hence the desired contradiction.

The key property proved about the notions of perfect and computation probabilistic non-interference in [10] 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 [10], Sketch) Let structures $(\hat{M}_1, S)$, $(\hat{M}_2, S)$ be given such that $(\hat{M}_1, S) \geq^x_{\text{sec}} (\hat{M}_2, S)$ for $x \in \{\text{poly}, \text{perf}\}$. 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 $(\hat{M}_2, S) \models^\text{poly} \mathcal{F}$ imply $Q_P^{||\cdot, \cdot||_c}(H, L) \in \text{NEGL}$ for all $H, L$ with $S_H \not\approx S_L$ and all computational distances $||\cdot, \cdot||_c$. Theorem 1 implies $Q_P^{||\cdot, \cdot||_c}(H, L) \leq Q_P^{||\cdot, \cdot||_c}(H, L) + \epsilon(k)$ for some $\epsilon \in \text{NEGL}$. Hence $Q_P^{||\cdot, \cdot||_c}(H, L) \in \text{NEGL}$ because the class of negligible function is closed under addition. Theorem 2 then yields $(\hat{M}_1, S) \models^\text{poly} \mathcal{F}$. The perfect case is proved by replacing $Q_P$ with $Q$ 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


A The Model of Reactive Simulatability

In this section we give a more comprehensive review of the model of reactive simulatability [57, 20] for the sake of completeness.

A.1 General System Model

Communication between different machines is done via ports. Inspired by the CSP-notation [36], we write input and output ports as p? and p!, respectively. The input and output ports in a port set P are written in(P) and out(P), respectively. Connections are defined by naming convention: port p! sends messages to p?. To achieve asynchronous timing, a message is not immediately delivered to its recipient, but is first stored in a special machine \( \tilde{p} \) called a buffer, where it waits to be scheduled. This can be done by the machine with the unique clock-out port p\( \triangleright \). To schedule the i-th message of buffer \( \tilde{p} \), it outputs i at p\( \triangleright \), 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 p! and 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 q! and q? as asynchronously connected.](image)

The precise definition of machines is a variant of probabilistic state-transition machines, similar to probabilistic I/O automata as sketched by Lynch [43]. 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 \( \Sigma \)) is a tuple

\[
M = (name_M, Ports_M, States_M, \delta_M, l_M, Init_M, Fin_M)
\]
of a machine name $\text{name}_M \in \Sigma^*$, a finite sequence $\text{Ports}_M$ of ports, a set $\text{States}_M \subseteq \Sigma^*$ of states, a probabilistic state-transition function $\delta_M$, a length function $\text{len}_M : \text{States}_M \rightarrow (\mathbb{N} \cup \{\infty\})^{\text{in}(\text{Ports}_M)}$, and sets $\text{Init}_M, \text{Fin}_M \subseteq \text{States}_M$ of initial and final states. Its input set is $\text{I}_M := (\Sigma^*)^{\text{in}(\text{Ports}_M)}$; the $i$-th element of an input tuple denotes the input at the $i$-th in-port. Its output set is $\text{O}_M := (\Sigma^*)^{\text{out}(\text{Ports}_M)}$. The empty word, $\epsilon$, denotes no in- or output at a port. $\delta_M$ maps each pair $(s, I) \in \text{States}_M \times \text{I}_M$ to a finite distribution over $\text{States}_M \times \text{O}_M$.

If $s \in \text{Fin}_M$ or $I = (\epsilon, \ldots, \epsilon)$, then $\delta_M(s, I) = (s, (\epsilon, \ldots, \epsilon))$ deterministically. Inputs are ignored beyond the length bounds, i.e., $\delta_M(s, I) = \delta_M(s, (I_{\text{in}(s)})_i := I_i |_{I_{\text{in}(s)}}$, for all $i$.

In the text, we often write “M” for $\text{name}_M$ 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 $\tilde{C}$ 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 ports($\tilde{C}$) denote the set of all ports of all machines in $\tilde{C}$. The completion $[\tilde{C}]$ of a collection $\tilde{C}$ consists of all machines of $\tilde{C}$ and the buffers needed for all the ports in $\tilde{C}$. The free ports free($\tilde{C}$) in a collection are those to which no other port in the collection connects. A collection $\tilde{C}$ is closed if its completion $[\tilde{C}]$ has no free ports except a special master clock-in port $\text{clk}^\ast$?. 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 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 $\tilde{C}$ 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 $M_{\text{CS}}$ for the current scheduler, and treats each port as a variable over $\Sigma^*$. Here, $r$ is an initially empty list, $M_{\text{CS}}$ a machine name initialized with $X$, and all port variables are initially $\epsilon$ except for $\text{clk}^\ast$? := 1. Probabilistic choices only occur in Phase 1.

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

Whenever a machine (this may be a buffer) with name $\text{name}_M$ is switched from $(s, I)$ to $(s', O)$, we append a step $(\text{name}_M, s, I', s', O)$ to the run $r$ for $I' := I |_{I_{\text{in}(s)}}$, except if $s$ is final or $I' = (\epsilon, \ldots, \epsilon)$. This gives a family of random variables

$$\text{run}_{\tilde{C}} = (\text{run}_{\tilde{C}, k})_{k \in \mathbb{N}}.$$  

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

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 \((\text{name}_M, s, I, s', O)\) where \( \text{name}_M \) is the name of a machine \( M \in \hat{M} \). The restriction \( r[S] \) 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

\[
\text{view}_C(\hat{M}) = (\text{view}_{C,k}(\hat{M}))_{k \in \mathbb{N}} \quad \text{and} \quad \text{run}_C[S] = (\text{run}_{C,k}[S])_{k \in \mathbb{N}}.
\]

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

For a one-element set \( \hat{M} = \{H\} \) we write \( \text{view}_C(H) \) for \( \text{view}_C(\{H\}) \).

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., \( q^c ! = q^c \), \( q^c ! = q^c \), \( q^c ? = q^c ? \), \( q^c ! = q^c ? \) 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 \( S \subseteq \text{free}(\hat{M}) \) is called service ports. If \( \hat{M} \) is clear from the context, let \( S := \text{free}(\hat{M}) \setminus S \). We call \( \text{forb}(\hat{M}, S) := \text{ports}(\hat{M}) \cup S^c \) 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 \( \text{conf} = (\hat{M}, S, U, A) \), where \( U \) is a set of simple machines without forbidden ports, i.e., \( \text{ports}(U) \cap \text{forb}(\hat{M}, S) = \emptyset \), and \( \hat{C} := \hat{M} \cup U \cup \{A\} \) is a closed collection. For simplicity, we often write \( \text{run}_{\text{conf}} \) and \( \text{view}_{\text{conf}}(\hat{M}) \) instead of \( \text{run}_{\hat{C}} \) and \( \text{view}_C(\hat{M}) \).

b) The set of (multi-party) configurations is written \( \text{Conf}(\hat{M}, S) \). The subset of configurations with polynomial-time users and a polynomial-time adversary is called \( \text{Conf}_{\text{poly}}(\hat{M}, S) \). The index \( \text{poly} \) 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 [10].
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 $\not\leadsto$ 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 $\text{conf} \in \text{Conf}(\hat{M}, S, \Gamma)$ for $S_H, S_L \in \Gamma$ of a structure $(\hat{M}, S)$, the guessing probability $P_{\text{guess}, \text{conf}}$ is defined as

$$P_{\text{guess}, \text{conf}} := P(b = b^* | r \leftarrow \text{run}_{\text{conf}, k}; b := r[p_{\text{bit}}]; b^* := r[p^*_{\text{bit}}]),$$

with the ports $p_{\text{bit}}$ and $p^*_\text{bit}$ 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 $(\hat{M}, S) \in \text{Sys}$, and a flow policy $\mathcal{F} = (\Gamma, \leadsto)$, $\Gamma = \{S_i | i \in I\}$ be given. We say that $(\hat{M}, S)$ fulfills the non-interference requirement defined by the flow policy $\mathcal{F}$

a) **perfectly**, written $(\hat{M}, S) \models^\text{perf} \mathcal{F}$, iff for every $H, L$ with $S_H \not\leadsto S_L$ and every non-interference configuration $\text{conf} \in \text{Conf}(\hat{M}, S, \Gamma)$ for $S_H$ and $S_L$, we have

$$P_{\text{guess}, \text{conf}} \leq \frac{1}{2}.$$

c) **computationally**, written $(\hat{M}, S) \models^\text{poly} \mathcal{F}$, iff for every $H, L$ with $S_H \not\leadsto S_L$ and every polynomial-time non-interference configuration $\text{conf} \in \text{Conf}^\text{poly}(\hat{M}, S, \Gamma)$ for $S_H$ and $S_L$ there exists a function $\epsilon \in \text{NEGL}$ such that

$$P_{\text{guess}, \text{conf}} \leq \frac{1}{2} + \epsilon(k).$$