Information and Computation 207 (2009)699–725

Contents lists available at ScienceDirect

Information and Computation

journal homepage: www.elsevier.com/locate/ic

Quantifying information leakage in process calculi[☆]

Michele Boreale

Dipartimento di Sistemi e Informatica, Università di Firenze, Viale Morgagni 65, I-50134 Firenze, Italy

A R T I C L E I N F O

Article history:
Received 10 January 2007
Revised 31 March 2008
Available online 23 January 2009

Keywords:
Process calculi
Secrecy
Information leakage
Information theory

A B S T R A C T

Building on simple information-theoretic concepts, we study two quantitative models of information leakage in the pi-calculus. The first model presupposes an attacker with an essentially unlimited computational power. The resulting notion of *absolute leakage*, measured in bits, is in agreement with secrecy as defined by Abadi and Gordon: a process has an absolute leakage of zero precisely when it satisfies secrecy. The second model assumes a restricted observation scenario, inspired by the testing equivalence framework, where the attacker can only conduct repeated success-or-failure experiments on processes. Moreover, each experiment has a cost in terms of communication effort. The resulting notion of leakage *rate*, measured in bits per action, is in agreement with the first model: the maximum amount of information that can be extracted by repeated experiments coincides with the absolute leakage A of the process. Moreover, the overall extraction cost is at least A/R, where R is the rate of the process. The compositionality properties of the two models are also investigated.

1. Introduction

In the field of language-based security, properties like non-interference [17] have traditionally been studied in a functional, all-or-nothing formulation. Only in recent years have models been proposed that enable forms of quantitative reasoning on such properties. Our interest here is in measuring leakage of sensitive information due to program execution. For a sequential program, it is natural to quantify this leakage by measuring the flow of information between secret ("high") and public ("low") variables induced by the computed function. An elegant theory of quantitative non-interference in this vein has been proposed by Clark et al. [10,12]. A comparison with this and other proposals in the literature is deferred to the concluding section.

In this paper, we study quantitative models of information leakage in concurrent programs, that is processes described in a process calculus. Processes come with no natural notion of computed function. Indeed, given a process, one is typically interested in quantifying the leakage arising from its interaction with the environment, hence in its *observable behaviour*. The difference in intent with respect to sequential programs can be illustrated by the following analogy. A smart-card implements a function that takes documents as input and releases documents signed with a secret key as output. However, typical attacks targeting the secret key do not focus on the function itself, but rather on the behaviour of the card, in terms e.g. of observable time variance of basic operations [20], or observable power consumption [21].

The starting point of our study is the notion of *secrecy* as formalized by Abadi and Gordon [1]. We will subsequently refer to this particular formulation as AG-secrecy. This is a fairly general concept, although in [1] it was defined in connection with the spi- and, limited to some introductory examples, the pi-calculus. In this paper, we shall stick for simplicity to the pi-calculus. Informally, AG-secrecy holds of a process P whose code mentions a parameter x representing a piece of sensitive information, if the observable behaviour of P does not depend on the actual values x may take on. In other words, an attacker

☆ Extended and revised version of [4].
E-mail address: boreale@dsi.unifi.it

0890-5401/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.ic.2008.12.007

cannot infer anything about x by interacting with P. The notion of "observable behaviour" is formalized in terms of a suitable behavioural equivalence, such as may testing equivalence [14,5].

Although elegant and intuitive, AG-secrecy is in practice too strict. The behaviour of virtually any useful program that protects a sensitive piece of information depends nontrivially on this information. Nevertheless, many such programs are considered secure, on the grounds that the amount of leaked information is, *on average*, negligible. The average is taken here over all possible values the sensitive information may take on. Consider a PIN-checking process $P(x)$ that receives a code from a user and checks it against a 4-digits secret PIN x, in order to authorize, or deny, a given operation. An attacker could easily submit a specific code of its choice to $P(x)$, say 4811, receive a deny and hence acquire negative information about x, i.e. "x is *not* 4811". However, assuming x has been chosen at random, such a small leak of information should be of no concern. In another scenario, an attacker could be allowed to query repeatedly $P(x)$, so that, given enough time, he/she could determine x with certainty. In this case, one is interested in quantifying the overall effort, in terms of interaction units (actions), necessary for the attacker to do so. Or, in other words, one's interest is in determining at which rate $P(x)$ leaks sensitive information.

In the present paper, we propose two quantitative models of leakage for processes that address the issues outlined above. The first model is designed for measuring *absolute* leakage of P, while the second model is designed for measuring the *rate* at which information is leaked by P. As explained below, the two models correspond to different assumptions on the control an attacker can exercise over P. The connections between these two models will also be clarified. We will take an *unconditional security* approach. Roughly, a "small leak" implies absence of attacks, while a "large leak" points to existence of attacks, without implying that such attacks can be mounted in practice. A more precise account of our work follows.

After quickly reviewing a few notions from Information Theory that will be used in the paper (Section 2), we introduce our reference language, a pi-calculus with data values (Section 3). In the first model (absolute leakage), we presuppose an attacker with full control over the process. Implicitly, we assume the attacker: (a) knows the process code $P(x)$, and (b), can, at no cost, produce as many copies as desired of the instance of P under consideration and run them. The role of these two strong assumptions is twofold: on the one hand, they set up a worst-case scenario, providing security guarantees independent from the computational power of actual attackers; on the other hand, they help to simplify the treatment of nondeterminism in processes. In particular, a consequence of assumption (b) is that all possible ways in which P's nondeterminism can be resolved should be experienced by the attacker. Idealizing this, one can say that the attacker can tell the equivalence class (of behaviours) the observed instance of P belongs to. A third assumption, commonly found when reasoning about protection of confidential data, is that: (c) the probability distribution of the data x is known to the attacker.

We are interested in the average amount of information about x that is leaked to the attacker by P under these assumptions. The average is taken over all values x may take on. In the language of unconditional security, this scenario can be formalized as follows. A piece of sensitive information is modeled as a random variable, say X. The a priori *uncertainty* of an adversary about X is measured by the Shannon *entropy* $H(X)$, expressed in bits [13]. For full generality, it is assumed that some "side-information" Y, possibly related to X, is publicly available: the conditional entropy $H(X \mid Y)$ measures the uncertainty about X given that Y is known. To illustrate these notions in a concrete case, consider the PIN-checking example. There, X represents a randomly chosen 4-digits secret code, hence $H(X) = \log(10^4) \approx 13.29$ bits. Y might represent whether $X = 4811$ or not, a piece of information the attacker might easily learn. Note that observing the event $(X = 4811)$ reduces the uncertainty about X to 0, while observing $(X \neq 4811)$ rules out one possibility reducing the uncertainty to $\log(10^4 - 1)$: on the average, observing Y reduces the uncertainty of the attacker to $H(X \mid Y) = 0 \cdot \Pr(X = 4811) + \log(10^4 - 1) \cdot (1 - \frac{1}{10^4}) \approx 13.28$.

Any process $P(x, y)$ and any two r.v.'s X and Y induce a new a random variable $Z = P(X, Y)$: following the discussion above, it is reasonable to stipulate that Z takes as values "observable behaviours", that is, equivalence classes of a fixed behavioral equivalence (Section 4). Now, the conditional entropy $H(X \mid Y, Z)$ quantifies the uncertainty on X left after observing both Y and Z. Hence the difference $I = H(X \mid Y) - H(X \mid Y, Z)$ is the amount of uncertainty about X removed by observing $Z = P(X, Y)$, that we take as the absolute leakage of P relative to X, Y (Section 5). We prove that this notion is in full agreement with the functional notion of AG-secrecy. In the special case when there is no side-information, this means that $P(x)$ respects AG-secrecy if and only if $P(X)$ has an absolute leakage of 0 for every random variable X. We also offer two alternative characterizations of zero-leakage, hopefully more amenable to automatic checking.

Next, we discuss the significance of absolute leakage in relation to certain security measures well-known from the literature (Section 6). Specifically, we show how to relate absolute leakage to the attacker's *error probability* of guessing the secret X given Z, and to the *guesswork* of X given Z, which measures the average number of attempts before correctly guessing X, in a scenario similar to that of dictionary attacks against password systems.

The second model we consider (rate of leakage, Section 7), refines the previous scenario by introducing a notion of *cost*. Adapting the testing equivalence framework from [14], we stipulate that an attacker can only conduct upon P repeated tests T_1, T_2,... each yielding a binary answer, success or failure. The attacker has full control – in the sense of the first model – over the compound systems $P||T_i$, but not over P itself. The security measure we are interested in is the overall number of synchronizations with P necessary for the adversary to extract one bit of information about X. Hence we define the *rate* at which P leaks information in terms of the maximal number of bits of information per visible action conveyed by an experiments $P||T_i$. We then give evidence that this is indeed a reasonable notion. First, we establish a relationship with the first model, showing that the absolute leakage A coincides with the maximum amount of information about X that can be extracted by repeated experiments on P, and that this costs the adversary at least A/R, where R is the rate of P. Second, in the vein of testing equivalence, we give an experiment-independent characterization of rate that only depends on the visible traces of the observed process. Extending the discussion of the absolute leakage model, we also clarify the relation of rate of leakage to the attacker's error probability and guesswork after an effort of N synchronizations.

The search for principles of compositional reasoning is a major motivation for studying information leakage in a process calculus setting. In both models, we show that the leakage (rate) attributable to a global system cannot exceed the sum of those imputable to individual sub-systems (with the exception of the parallel composition operator, in the case of rate of leakage). We prove that under suitable conditions iteration does preserve rate, in the sense that the rate of $*P$ equals that of P, which is expected from a sensible notion of rate.

We will illustrate the application of the proposed models to a non-trivial example, a message-routing system inspired by anonymity protocols in the style of Crowds [28] (Section 8).

Some discussion on the limitations of the present approach, remarks on further research and a discussion on related works conclude the paper (Section 9). A table summarizing the main notations used throughout the paper and a few technical definitions and proofs are reported in separate appendices (Appendices 9, 9, 9).

2. Preliminary notions

We briefly recall a few concepts from probability and elementary Information Theory; see e.g. [13,33] for full definitions and underlying motivations. Recall that a random variable (r.v.) is a function $X : \Omega \to U$ where Ω is a probability space, U (called the *state space*) is the carrier of a σ-algebra $\mathcal{F}$ and for each element $F \in \mathcal{F}$, $X^{-1}(F)$ is an event of Ω (otherwise said, X is measurable). In this paper, we shall confine ourselves to *discrete* random variables, that is, random variable in which U is an at most countable set and $\mathcal{F}$ is the power-set σ-algebra over U: this amounts to requiring that for each $u \in U$, $X^{-1}(u)$ is an event of Ω. We let $X, Y, \ldots$ range over discrete random variables. We say that a r.v. X is of *type* U, and write $X : U$, if U is the state space of X (i.e. $X(\Omega) \subseteq U$); we call elements of U *outcomes* of X. Unless otherwise stated, we shall assume the U is finite. We define $|X|$ as the number of possible outcomes of X, that is $|X| \stackrel{\text{def}}{=} |\{u \in U | \Pr(X = u) > 0\}|$. We shall make use of the concepts of independent and uniformly distributed (u.d.) random variable, defined as usual. As a function, every random variable induces a partition into events of its domain Ω, which is $\{X^{-1}(u) \,|\, u \in X(\Omega)\}$: we say that two random variables X and Y are *equivalent* if they induce on Ω the same partition (this does not imply that X and Y coincide). A vector of random variables $\tilde{X} = (X_1, \ldots, X_n)$, where the $X_i : U_i$ for $1 \le i \le n$ are r.v.'s defined on the same probability space Ω, is just a r.v. of type $U_1 \times \cdots \times U_n$.

Let $X : U$ and $Y : V$ be r.v. The *entropy* of X and the *conditional entropy of X given Y* are defined, respectively, as:

$$H(X) \stackrel{\text{def}}{=} -\sum_{u \in U} \Pr(X = u) \cdot \log(\Pr(X = u))$$
$$H(X \mid Y) \stackrel{\text{def}}{=} \sum_{v \in V} H(X \mid Y = v) \cdot \Pr(Y = v)$$

where all logarithms are taken to the base of 2, by convention $0 \cdot \log 0 = 0$ and for any event e of Ω, $H(X \mid e)$ is the conditional entropy of X given e, defined as

$$H(X \mid e) \stackrel{\text{def}}{=} -\sum_{u \in U} \Pr(X = u \mid e) \cdot \log(\,\Pr(X = u \mid e)\,)\,.$$

Example 2.1. Let X represent the random choice of a PIN-code between 1 and N. Our a priori uncertainty about X is measured by its entropy

$$H(X) = -\sum_{i=1}^{N} \frac{1}{N} \log\left(\frac{1}{N}\right) = \log N\,.$$

Assume that, although ignoring the value of X, we get to know its parity, odd or even. Let Y be the r.v. that yields 1 if X is odd, 0 otherwise. Then, assuming N is even, our uncertainty about X after observing $Y = 0$ is measured by

$$\begin{aligned} H(X \mid Y = 0) &= -\sum_{i \in 1..N} \Pr(X = i \mid Y = 0) \log\Big(\Pr(X = i | Y = 0)\Big) \\ &= -\sum_{i \in 1..N,\, i \text{ odd}} \frac{1}{N/2} \log\left(\frac{1}{N/2}\right) = \log\left(\frac{N}{2}\right) = \log N - 1 \end{aligned}$$

that is, observing $Y = 0$ reduces our uncertainty by 1 bit. Similarly, $H(X \mid Y = 1) = \log N - 1$. Hence, the average uncertainty after observing Y is

$$H(X \mid Y) = \frac{1}{2} H(X \mid Y = 0) + \frac{1}{2} H(X \mid Y = 1) = \log N - 1\,.$$

Note that two equivalent random variables exhibit the same entropy and conditional entropies. For a vector $(X_1,\ldots,X_n)$ of random variables, we shall abbreviate $H\Big((X_1,\ldots,X_n)\Big)$ as $H(X_1,\ldots,X_n)$. The following fundamental (in)equalities hold:

$$0 \le H(X) \le \log |X| \tag{1}$$
$$H(X,Y) = H(X|Y) + H(Y) \quad \text{(chain rule)} \tag{2}$$
$$H(X_1,\ldots,X_n) \le H(X_1) + \cdots + H(X_n) \tag{3}$$

where: in (1), equality on the left holds iff X is a constant, and equality on the right holds iff X is u.d. on $\{u \in U \mid \Pr(X = u) > 0\}$; in (3), equality holds iff the X_i's are pairwise independent. Note that by (2) and (3), $H(X|Y) = H(X)$ iff X and Y are independent. If $Y = F(X)$ for some function F then $H(Y|X) = 0$. *Information on X conveyed by Y* (aka, *mutual information between X and Y*) is

$$I(X;Y) \stackrel{\text{def}}{=} H(X) - H(X \mid Y)\,.$$

By the chain rule, $I(X;Y) = I(Y;X)$, and $I(X;Y) = 0$ iff X and Y are independent. Mutual information can be generalized by conditioning on another r.v. Z: $I(X;Y \mid Z) \stackrel{\text{def}}{=} H(X \mid Z) - H(X \mid Z, Y)$. Conditioning on Z may in general either increase or decrease mutual information between X and Y. Note that entropy of a r.v. only depends on the underlying probability distribution; thus any probability vector $\tilde{p} = (p_1,\ldots,p_n)$ ($p_i \ge 0$, $\sum_i p_i = 1$) determines a single entropy value denoted $H(\tilde{p})$. We shall often abbreviate the binary entropy $H(p, 1-p)$ as $\mathcal{B}(p)$.

3. A process calculus

3.1. Syntax

We assume a countable set of *variables* $\mathcal{V} = \{x, y, \ldots\}$, a family of non-empty, finite *value-sets* $\mathfrak{U} = \{U, V, \ldots\}$, and a countable set of *names* $\mathcal{N} = \{a, b, \ldots\}$, partitioned into a family of *sorts* $\mathcal{S}, \mathcal{S}', \ldots$. We let u, v be generic elements of a finite value-set. We assume a fixed function that maps each variable x to some $T \in \mathfrak{U} \cup \{\mathcal{S}, \mathcal{S}', \ldots\}$, written $x : T$, and say that x has *type* T; we assume the inverse image of each T is infinite. These notations are extended to tuples as expected, e.g. for $\tilde{x} = (x_1,\ldots,x_n)$ and $\tilde{T} = (T_1,\ldots,T_n)$, $\tilde{x} : \tilde{T}$ means $x_1 : T_1, \ldots, x_n : T_n$. By slight abuse of notation, we sometimes denote by $\tilde{T}$ the cartesian product $T_1 \times \cdots \times T_n$.

An *evaluation* σ is a partial map from $\mathcal{V}$ to $\bigcup_{U \in \mathfrak{U}} U \cup \mathcal{N}$ that respects typing, that is, for each $x \in \mathrm{dom}(\sigma)$, $x : T$ implies $\sigma(x) \in T$. We denote by $[^{\tilde{d}}/_{\tilde{x}}]$ the evaluation mapping $\tilde{x}$ to $\tilde{d}$ component-wise. By $t\sigma$, where t is a term over an arbitrary signature with free variables $\mathrm{fv}(t) \subseteq \mathcal{V}$, we denote the result of replacing each free variable $x \in \mathrm{dom}(\sigma) \cap \mathrm{fv}(t)$ with $\sigma(x)$.

We assume a language of logical *formulae* $\phi, \psi, \ldots$. We leave the language unspecified, but assume it includes a first order calculus with variables $\mathcal{V}$, that function symbols include all values in $\mathfrak{U}$ and names as constants, and that the set of predicates includes equality $[x = y]$. For ϕ and σ s.t. $\mathrm{dom}(\sigma) \supseteq \mathrm{fv}(\phi)$, we write $\sigma \models \phi$ if $\phi\sigma$ is valid (i.e. a tautology). If $\sigma \models \phi$ for all evaluations σ s.t. $\mathrm{dom}(\sigma) \supseteq \mathrm{fv}(\phi)$, then we write $\models \phi$. As usual, $\phi \Rightarrow \psi$ means $\models \phi \to \psi$. We will often write $\phi(\tilde{x})$ to indicate that the free variables of ϕ are included in $\tilde{x}$, and, in that case, abbreviate $\phi[^{\tilde{u}}/_{\tilde{x}}]$ as $\phi(\tilde{u})$.

The process language is a standard pi-calculus with variables and data values. We assume a countable set of *identifiers* $A, B, \ldots$ and use $e, e' \ldots$ to range over an unspecified set of *expressions*, that can be formed starting from variables, values and names. The syntax of processes $P, Q, \ldots$ is given by the constructors of *inaction, silent prefix, input prefix, output prefix, boolean guard, nondeterministic choice, restriction, parallel composition* and *process identifier*, according to the grammar below.

$$m ::= x \;\big|\; a$$
$$P, Q ::= \mathbf{0} \;\big|\; \tau.P \;\big|\; m(\tilde{x}).P \;\big|\; \overline{m}\tilde{e}.P \;\big|\; \phi P \;\big|\; P + P \;\big|\; (\nu b)P \;\big|\; P|P \;\big|\; A(\tilde{e})\,.$$

Each identifier A has an associated defining equation of the form $A(\tilde{x}) \stackrel{\text{def}}{=} P$. Input prefix $m(\tilde{x})$. and restriction (νb) are binders for $\tilde{x}$ and b, respectively, thus, notions of free variables (fv) and free names (fn) arise as expected. We identify processes up to alpha-equivalence. We assume a few constraints on the syntax above: $\tilde{x}$ is a tuple of distinct elements in input prefix and in $A(\tilde{x}) \stackrel{\text{def}}{=} P$, and in the latter $\mathrm{fv}(P) \subseteq \tilde{x}$; ϕ is quantifier-free. We assume a fixed sorting system *à la* Milner. In particular, each sort $\mathcal{S}$ has an associated *sort object* $ob(\mathcal{S}) = (T_1, \ldots, T_k)$ ($k \ge 0$). Here, each T_i is either a sort $\mathcal{S}$ or a value-set U from the universe $\mathfrak{U}$. Informally, a process obeys this sorting system if in every input and output prefix, a name/variable m of sort $\mathcal{S}$ carries a tuple of objects of the sort specified by $ob(\mathcal{S})$; we omit the details that are standard. We let $\mathcal{P}^{\mathrm{o}}$ the set of processes (possibly containing free variables) obeying these conditions and $\mathcal{P}^{\mathrm{c}}$ its subset of *closed* processes. Notationally, we shall often omit trailing $\mathbf{0}$'s, writing e.g. $a.b.$ instead of $a.b.\mathbf{0}$, we shall write $\sum_{i=1}^{n} P_i$ for nondeterministic choice $P_1 + \cdots + P_n$, and let replication $!P$ denote the process defined by the equation: $!P \stackrel{\text{def}}{=} P|!P$.

Table 1
Operational semantics of $\mathcal{P}^{\mathrm{c}}$.

$$(\textsc{inp})\ \frac{a:\mathcal{S}\quad \tilde{d}:ob(\mathcal{S})}{a(\tilde{x}).P \xrightarrow{a\tilde{d}} P[^{\tilde{d}}/_{\tilde{x}}]}\qquad (\textsc{out})\ \frac{\tilde{e}\downarrow\tilde{d}}{\overline{a}\tilde{e}.P \xrightarrow{\overline{a}\tilde{d}} P}\qquad (\textsc{tau})\ \frac{-}{\tau.P \xrightarrow{\tau} P}$$

$$(\textsc{sum}_1)\ \frac{P \xrightarrow{\mu} P'}{P+Q \xrightarrow{\mu} P'}\qquad (\textsc{par}_1)\ \frac{P \xrightarrow{\mu} P'\quad \mathrm{bn}(\mu)\cap\mathrm{fn}(Q)=\emptyset}{P|Q \xrightarrow{\mu} P'|Q}$$

$$(\textsc{com}_1)\ \frac{P \xrightarrow{(\nu\tilde{c})\overline{a}\tilde{d}} P'\quad \tilde{c}\cap\mathrm{fn}(Q)=\emptyset\quad Q \xrightarrow{a\tilde{d}} Q'}{P|Q \xrightarrow{\tau} (\nu\tilde{c})(P'|Q')}$$

$$(\textsc{open})\ \frac{P \xrightarrow{(\nu\tilde{c})\overline{a}\tilde{d}} P'\quad a\neq b\quad b\in\tilde{d}}{(\nu b)P \xrightarrow{(\nu\tilde{c},b)\overline{a}\tilde{d}} P'}\qquad (\textsc{res})\ \frac{P \xrightarrow{\mu} P'\quad b\notin\mathrm{n}(\mu)}{(\nu b)P \xrightarrow{\mu} (\nu b)P'}$$

$$(\textsc{phi})\ \frac{\phi\downarrow true\quad P \xrightarrow{\mu} P'}{\phi P \xrightarrow{\mu} P'}\qquad (\textsc{ide})\ \frac{A(\tilde{x}) \overset{\mathrm{def}}{=} P\quad \tilde{x}:\tilde{T}\quad \tilde{d}:\tilde{T}\quad P[^{\tilde{d}}/_{\tilde{x}}] \xrightarrow{\mu} P'}{A(\tilde{d}) \xrightarrow{\mu} P'}$$

3.2. Semantics

We assume over $\mathcal{P}^{\mathrm{c}}$ the standard *early* operational semantics of pi-calculus. Transitions are the form $P \xrightarrow{\mu} P'$, where μ is one of τ (invisible action), $a\tilde{d}$ (input action) or $(\nu\tilde{c})\overline{a}\tilde{d}$ with $\tilde{c}\subseteq\tilde{d}\setminus\{a\}$ (output action) and $d ::= a \mid u$ (name or value). We let $\tilde{d}$ range over tuples of elements of names and/or values, let $\mathrm{n}(\mu)$ denote the set of names occurring in μ and define the set of bound names of μ as: $\mathrm{bn}(\mu)=\tilde{c}$ if $\mu=(\nu\tilde{c})\overline{a}\tilde{d}$ and $\mathrm{bn}(\mu)=\emptyset$ otherwise. An evaluation function $\downarrow$ is presupposed that maps closed expressions and formulae to values/names and to $\{true, false\}$, respectively, with the proviso that each name is mapped to itself ($a\downarrow a$). This is extended to tuples of expressions componentwise.

The operational semantics of $\mathcal{P}^{\mathrm{c}}$ is given by the rules reported in Table 3.2. Symmetric versions of rules $(\textsc{sum}_1)$, $(\textsc{par}_1)$ and $(\textsc{com}_1)$ are not shown for brevity.

A few standard notations will be made use of. In particular, for each *visible* (different from τ) action α, $P \overset{\alpha}{\Longrightarrow} P'$ means $P(\xrightarrow{\tau})^* \xrightarrow{\alpha} (\xrightarrow{\tau})^* P'$. This notation is extended to any sequence of visible actions $s=\alpha_1\cdots\alpha_n$ (i.e. a *trace*), $P \overset{s}{\Longrightarrow} P'$, as expected. Finally, $P \overset{s}{\Longrightarrow}$ means that there is P' s.t. $P \overset{s}{\Longrightarrow} P'$.

In the rest of the paper, we let $\asymp$ be a fixed equivalence relation over $\mathcal{P}^{\mathrm{c}}$ and denote by $[Q]_{\asymp}$ the $\asymp$-equivalence class of process Q. For the moment, we leave $\asymp$ unspecified, but assume it is included in *trace equivalence* [5], it includes *strong bisimulation* [29], it preserves all operators of the calculus, except possibly input prefix, and it satisfies the monoid laws for $+$ and $|$ with $\mathbf{0}$ as unit. We will have more to say on the role played by specific behavioural equivalences later on (see Section 5.2).

Another concept we shall rely upon is that of *most general boolean*, borrowed from [19,6], that is, the most general condition under which two given open processes are equivalent.

Definition 3.1 (*mgb*). Let $P(\tilde{x})$ and $Q(\tilde{y})$ be two open processes, with $\tilde{x}:\tilde{U}$ and $\tilde{y}:\tilde{V}$. We denote by $\mathrm{mgb}(P(\tilde{x}), Q(\tilde{y}))$ a chosen formula $\phi(\tilde{x},\tilde{y})$ s.t. for each $\tilde{u}\in\tilde{U}$ and $\tilde{v}\in\tilde{V}$: $P(\tilde{u})\asymp Q(\tilde{v})$ if and only if $\phi(\tilde{u},\tilde{v})$ is true.

It is worthwhile to notice that, under certain assumptions, mgb's for a pair of open pi-processes can be automatically computed relying on a *symbolic* operational semantics [19,6]. Let us recall that a symbolic transition also carries a logical formula: $P \xrightarrow{\mu,\phi} P'$. Informally, ϕ represents the exact condition on the free variables of P under which the given transition is enabled. For example, one has

$$([x=y]\overline{z}v.P)\mid y(w).Q \xrightarrow{\tau,[x=y]\wedge[y=z]} P|Q[^{v}/_{w}]\,.$$

In [19], an algorithm is described to compute mgb's for a pair of processes both having *finite* symbolic transition systems, in the case of strong bisimilarity.

4. Processes as random variables

This section is devoted to presenting a technical device that allows us to transform (open) processes into random variables. Let us define an *open process* as a pair $(P,\tilde{x})$, written $P(\tilde{x})$, such that $\tilde{x}$ is a tuple of distinct variables of some type $\tilde{U}\subseteq\mathfrak{U}$ and $P\in\mathcal{P}^{\mathrm{o}}$ is such that $\mathrm{fv}(P)\subseteq\tilde{x}$. When no confusion about $\tilde{x}$ arises, we shall abbreviate $P[^{\tilde{u}}/_{\tilde{x}}]$ as $P(\tilde{u})$ and $(P[^{\tilde{y}}/_{\tilde{x}}])(\tilde{y})$ as $P(\tilde{y})$ ($\tilde{y}$ a tuple of distinct variables.)

Definition 4.1 (*Open processes as random variables*)**.** Let $P(\tilde{x})$ be an open process and $\tilde{X}$ be a vector of random variables, with $\tilde{x} : \tilde{U}$ and $\tilde{X} : \tilde{U}$, for one and the same $\tilde{U}$. Let $F : \tilde{U} \longrightarrow \mathcal{P}^c/\asymp$ be the function $\tilde{u} \mapsto [P(\tilde{u})]_{\asymp}$. We denote by $P(\tilde{X})$ the random variable $F(\tilde{X})$.

In essence, the above definition tells us how to "plug" a random variable X into an open process $P(x)$ thus obtaining a new random variable $P(X)$. Note that this definition does not involve anything like textual replacement of x by X inside $P(x)$. What we do is simply taking the function $F : U \to \mathcal{P}^c/\asymp$, defined as $F(u) = [P(u)]_{\asymp}$ for each u, and composing it with the random variable X seen as a function. Doing so, we obtain a new random variable $F \circ X$, written $P(X)$, that has $\mathcal{P}^c/\asymp$ as a state space – that is, the outcomes of $P(X)$ are $\asymp$-equivalence classes. The semantical aspects of the definition are subsumed by $\asymp$. The definition itself is parametric[1] with the actual choice of $\asymp$: different choices for $\asymp$ may correspond to different assumptions on the observational power of the attacker. We shall elaborate on this point in Section 5.2.

The next example is very simple and only serves to convey some intuition about the above definition.

Example 4.1 (PIN-*checking*)**.** A PIN-checking process can be defined as follows. Here, $x, z : 1..k$ for some integer k and x represents the secret code. The situation is modeled where an observer can freely interact with the checking process.

$$Check(x) \stackrel{\text{def}}{=} a(z).([z = x]\overline{ok}.Check(x) \,+\, [z \neq x]\overline{no}.Check(x)) \,. \tag{4}$$

In this case, the range of the function $F :\ u \mapsto [Check(u)]_{\asymp}$ mentioned in Definition 4.1 has k distinct elements, as $u \neq u'$ implies $Check(u) \not\asymp Check(u')$: for instance, $Check(u)$ has the trace $au \cdot \overline{ok}$, which $Check(u')$ has not. As a consequence, for a r.v. $X : 1..k$, the distribution of $P(X)$ mirrors exactly that of X. E.g., if X is uniformly distributed on $1..k$, then $Z = P(X)$ is u.d. over $\{[Check(1)]_{\asymp}, \ldots, [Check(k)]_{\asymp}\}$, i.e. the probability of each outcome of Z is $1/k$.

In the sequel, the following two facts will turn out to be useful. First, from Definition 4.1, it is immediate to see that the distribution of $P(\tilde{X})$ is given by the following, for each $o = [Q]_{\asymp}$:

$$\Pr\left(P(\tilde{X}) = o\right) \;=\; \sum_{\tilde{u}:\, P(\tilde{u}) \asymp Q} \Pr\left(\tilde{X} = \tilde{u}\right) \tag{5}$$

Second, note that, if $P(\tilde{u}) \asymp Q(\tilde{u})$ for each $\tilde{u}$, then, for any $\tilde{X}$, $P(\tilde{X})$ and $Q(\tilde{X})$ are the same random variable.

5. Absolute leakage

Throughout the section and unless otherwise stated, we let $P(\tilde{x}, \tilde{y})$ be an arbitrary open process, with $\tilde{x} : \tilde{U}$ and $\tilde{y} : \tilde{V}$, while $\tilde{X} : \tilde{U}$ and $\tilde{Y} : \tilde{V}$ are two vectors of random variables, and $Z = P(\tilde{X}, \tilde{Y})$.

5.1. Definitions and basic properties

Definition 5.1 (*Absolute leakage*)**.** Let $P(\tilde{x}, \tilde{y})$ be an open process, $\tilde{X}$ and $\tilde{Y}$ be r.v. and let $Z = P(\tilde{X}, \tilde{Y})$. The (absolute) *information leakage from $\tilde{X}$ to P given $\tilde{Y}$* is defined as:

$$\mathcal{A}(P; \tilde{X} \mid \tilde{Y}) \stackrel{\text{def}}{=} I(\tilde{X}; Z \mid \tilde{Y}) = H(\tilde{X}|\tilde{Y}) - H(\tilde{X} \mid \tilde{Y}, Z) \,.$$

When $\tilde{Y}$ is empty, we simply write absolute leakage as $\mathcal{A}(P; \tilde{X})$. A first useful fact on the definition above is that leakage is nothing but the uncertainty about $P(\tilde{X}, \tilde{Y})$ after observing $\tilde{Y}$.

Lemma 5.1. *Let $P(\tilde{x}, \tilde{y})$ be an open process, $\tilde{X}$ and $\tilde{Y}$ be r.v. and let $Z = P(\tilde{X}, \tilde{Y})$. Then $\mathcal{A}(P; \tilde{X} \mid \tilde{Y}) = H(Z \mid \tilde{Y})$. In particular, if $\tilde{y}$ is empty, $\mathcal{A}(P; \tilde{X}) = H(Z)$.*

Proof. This is a simple application of the chain rule (2). By symmetry of mutual information I, we have $\mathcal{A}(P;\, \tilde{X} \mid \tilde{Y}) = I(\tilde{X};\, Z \mid \tilde{Y}) = H(Z \mid \tilde{Y}) - H(Z \mid \tilde{X}, \tilde{Y})$. But $Z = P(\tilde{X}, \tilde{Y})$ is a function of $\tilde{X}$ and $\tilde{Y}$, hence $H(Z \mid \tilde{X}, \tilde{Y}) = 0$. □

Example 5.1 (PIN-*checking*)**.** The process $Check(x)$ defined in (4) leaks *all* information about x. For example, if X is u.d on $1..k$ then $Z = P(X)$ is u.d. over a set of k outcomes. Hence, using Lemma 5.1, $\mathcal{A}(Check; X) = H(Z) = \log k = H(X)$.

Suppose now the adversary cannot interact freely with *Check*, rather he can observe the outcome of a user's interacting once with *Check*. The adversary knows the code y tried by the user. We represent the user simply as $\overline{a}y$, hence the new system is

$$OneTry(x, y) \stackrel{\text{def}}{=} (\nu a)(Check(x)|\overline{a}y) \,. \tag{6}$$

[1] Strictly speaking, we should make the dependency of $P(\tilde{X})$ from $\asymp$ explicit by writing e.g. $P_{\asymp}(\tilde{X})$, but we shall omit to do so unless strictly necessary.

Clearly, for any r.v. $X, Y : 1..k$, the random variable $Z = OneTry(X,Y)$ has only two possible outcomes, that is $[\tau.\overline{ok}]_{\asymp}$ and $[\tau.\overline{no}]_{\asymp}$. These outcomes have probabilities $\Pr(X = Y)$ and $\Pr(X \neq Y)$, respectively. In the case where X and Y are uniformly distributed and independent, these probabilities are $1/k$ and $1 - 1/k$, respectively. We are interested in $\mathcal{A}(OneTry;\ X \mid Y)$. Easy calculations show that Z and Y are independent: indeed, for $o = [\tau.\overline{ok}]_{\asymp}$ and any $i \in 1..k$, $\Pr(Z = o \mid Y = i) = \Pr(X = i) = \frac{1}{k}$, while $\Pr(Z = o) = \Pr(X = Y) = \frac{1}{k}$, and similarly for $o = [\tau.\overline{no}]_{\asymp}$, that is $\Pr(Z = o \mid Y = i) = \Pr(Z = o)$. For the sake of concreteness, let us assume $k = 10$. Using Lemma 5.1 we can compute absolute leakage as

$$\mathcal{A}(OneTry; X \mid Y) = H(Z \mid Y) = H(Z) = \mathcal{B}\left(\frac{1}{10}\right) \approx 0.469\,.$$

In this case, knowledge of Y brings no advantage to the adversary.

Example 5.2 (*A mobile object*). Consider an object that can freely move within a grid of coordinates $k \times k$, starting from a secret location at coordinate (x_1, x_2) (x_1 row, x_2 column). For some reason, only the *directions* (w, e, n, s) of the object's moves are observable. Considering $x_1, x_2 : 1, \ldots, k$, we have (please note that "+" denotes nondeterministic choice below):

$$\begin{aligned} Mobile(x_1,x_2) \stackrel{\text{def}}{=} & [x_1 > 1]w.Mobile(x_1 - 1, x_2) \ + \ [x_1 < k]e.Mobile(x_1 + 1, x_2) \\ & + \ [x_2 < k]n.Mobile(x_1, x_2 + 1) \ + \ [x_2 > 1]s.Mobile(x_1, x_2 - 1)\,. \end{aligned} \tag{7}$$

The "game" here is the adversary's guessing the secret location by only observing the sequence of movement directions. Note that $(u_1, u_2) \neq (u'_1, u'_2)$ implies $Mobile(u_1, u_2) \not\asymp Mobile(u'_1, u'_2)$. Hence, for any two random variables $X_1, X_2 : 1..k$, $Z = Mobile(X_1, X_2)$ is a random variable whose distribution mirrors that of (X_1, X_2). Hence in this case there is a total leakage of information. E.g., if X_1 and X_2 are uniformly distributed and independent, then, by (3), $\mathcal{A}(Mobile\,;\ X_1, X_2) = H(Z) = H(X, Y) = 2 \cdot \log k$. Suppose a "confounder" process $C \stackrel{\text{def}}{=} w.C + e.C$ is inserted into the system, that is, consider the system $Mobile|C$ (note that no synchronization can take place between $Mobile$ and C, their actions are merely interleaved). The presence of confounder makes two objects lying in the same row indistinguishable: $Mobile(u,v)|C \asymp Mobile(u',v')|C$ iff $u = u'$. As a consequence, the information conveyed by the new system is halved: $\mathcal{A}(Mobile|C\ ;\ X_1, X_2) = \log k$.

The next result asserts that absolute leakage is compositional, in the following sense: the amount of information leaked by a global system cannot exceed the overall information leaked by individual sub-systems observed in isolation. There is a technical condition on the the side information, as $\tilde{Y}$ must be decomposable into independent pieces, each of which is related only to a single sub-system (at the moment, we do not know whether this condition can be relaxed). The proof of the result is a consequence of inequality (3) and of (an instance of) the so called "data processing" inequality [13]. The latter implies that for any r.v. W and function F of appropriate domain, $H(F(W)) \leq H(W)$.

Fix a sequence of distinct process variables (placeholders for processes) $\mathfrak{X}_1, \mathfrak{X}_2, \ldots$. Recall that a *(n-holes) context* is a process term containing at least one occurrence of process variable $\mathfrak{X}_i$, for each $1 \leq i \leq n$ (the $\mathfrak{X}_i$ represents the "holes"). We write $C[\cdot, \ldots, \cdot]$ for a generic context and $C[P_1, \ldots, P_n]$ for the process obtained by replacing $\mathfrak{X}_1, \mathfrak{X}_2, \ldots$ with $P_1, P_2, \ldots$. We say that $C[\cdot, \ldots, \cdot]$ *preserves* $\asymp$ if whenever $P_i \asymp P'_i$ for $1 \leq i \leq n$ then $C[P_1, \ldots, P_n] \asymp C[P'_1, \ldots, P'_n]$.

Proposition 5.1 (Compositionality). *Let $C[\cdot, \ldots, \cdot]$ be a n-holes context that preserves $\asymp$, and let $Q_i(\tilde{x}, \tilde{y}_i)$ be open processes, $1 \leq i \leq n$, where $\tilde{y} = (\tilde{y}_1, \ldots, \tilde{y}_n)$. Let $P(\tilde{x}, \tilde{y}) = C[Q_1(\tilde{x}, \tilde{y}_1), \ldots, Q_n(\tilde{x}, \tilde{y}_n)]$. Let $\tilde{Y} = (\tilde{Y}_1, \ldots, \tilde{Y}_n)$, with the $\tilde{Y}_i$'s pairwise independent. Let $L = \mathcal{A}(P; \tilde{X} \mid \tilde{Y})$ and $L_i = \mathcal{A}(Q_i; \tilde{X} \mid \tilde{Y}_i)$ for $1 \leq i \leq n$. Then*

$$L \leq \sum_{i=1}^{n} L_i\,. \tag{8}$$

Proof. Let $G : (\mathcal{P}^c / \asymp)^n \to \mathcal{P}^c / \asymp$ be the function defined by $([P_1]_{\asymp}, \ldots, [P_n]_{\asymp}) \mapsto \Big[C[P_1, \ldots, P_n] \Big]_{\asymp}$ (note that this is well-defined since $C[\cdot]$ is $\asymp$-preserving). For $1 \leq i \leq n$, let $Z_i = Q_i(\tilde{X}, \tilde{Y}_i)$. Then $P(\tilde{X}, \tilde{Y}) = G(Z_1, \ldots, Z_n)$. Therefore $\mathcal{A}(P; \tilde{X} \mid \tilde{Y})$ can be written as $H\Big(G(Z_1, \ldots, Z_n), \tilde{Y} \Big) - H(\tilde{Y})$. By the data-processing inequality and independence of the $\tilde{Y}_i$, the last term is $\leq H(Z_1, \ldots, Z_n, \tilde{Y}_1, \ldots, \tilde{Y}_n) - \sum_{i=1,\ldots,n} H(\tilde{Y}_i)$. By inequality (3), the last term is in turn $\leq \sum_{i=1,\ldots,n} H(Z_i, \tilde{Y}_i) - H(\tilde{Y}_i) = \sum_{i=1,\ldots,n} H(Z_i \mid \tilde{Y}_i) = \sum_{i=1,\ldots,n} \mathcal{A}(Q_i;\ \tilde{X} \mid \tilde{Y}_i)$. □

In the case of parallel composition, the inequality (8) specializes to $\mathcal{A}(P|Q\ ;\ \tilde{X} \mid \tilde{Y}) \leq \mathcal{A}(P\ ;\ \tilde{X} \mid \tilde{Y}_1) + \mathcal{A}(Q\ ;\ \tilde{X} \mid \tilde{Y}_2)$. Moreover, (8) implies that leakage is never increased by unary operators preserving $\asymp$. In the case of replication ! this leads to the somewhat unexpected conclusion, which holds provided $\asymp$ is preserved by !

$\mathcal{A}(!P\,;\,\tilde{X}\ \mid\ \tilde{Y}) \leq \mathcal{A}(P\,;\,\tilde{X}\ \mid\ \tilde{Y})\,.$

The intuition underlying the above inequality can be explained under the assumptions informally discussed the Introduction: once the attacker is given P, he/she can produce as many copies of P as desired and possibly run them in parallel, thus simulating $!P$ if necessary, while the converse is not in general possible (i.e., given $!P$ it is not possible in general to simulate P; see also Example 5.3 below). In general, instances of inequality (8) may hold strict or not, as shown by the following example.

Example 5.3. Consider $P(x) = ([x = 0]a)|a$, where $x : \{0, 1\}$, and X is u.d. on the same set. Then $1 = \mathcal{A}(P; X) > \mathcal{A}(!P; X) = 0$. The reason for the last equality is that for $v \in \{0, 1\}$, $!P(v) \asymp !a$, that is, the behaviour of $!P(x)$ does not depend on x, so $H(P(X)) = 0$.
On the other hand, consider $P_1(x) = [x = 2]a + [x = 4]a$ and $P_2(x) = [x = 1]b + [x = 2]b$, where this time $x : 1..4$, and X is u.d. on the same set. Then $\mathcal{A}(P_1|P_2\,;\,X) = \mathcal{A}(P_1\,;\,X) + \mathcal{A}(P_2\,;\,X) = \mathcal{B}(\frac{1}{2}) + \mathcal{B}(\frac{1}{2}) = 2$.

Our next task is to investigate the situation of zero leakage. We start from Abadi and Gordon' definition of Secrecy [1]. According to this definition, a process $P(\tilde{x})$ keeps $\tilde{x}$ secret if the observable behaviour of $P(\tilde{x})$ does not depend on the actual values $\tilde{x}$ may take on. Partly motivated by the non-interference scenario [17,16,34], where variables are classified into "low" and "high", we find it natural to generalize the definition of [1] to the case where the behaviour of P may also depend on further parameters $\tilde{y}$ known to the adversary.

Definition 5.2 (*Generalized secrecy*). We say that $P(\tilde{x}, \tilde{y})$ *keeps* $\tilde{x}$ *secret given* $\tilde{y}$ if, for each $\tilde{v} \in \tilde{V}$, and for each $\tilde{u} \in \tilde{U}$ and $\tilde{u}' \in \tilde{U}$, it holds that $P(\tilde{u}, \tilde{v}) \asymp P(\tilde{u}', \tilde{v})$.

The main result of the section states agreement of diverse notions of secrecy: functional (definition above), quantitative (zero leakage) and logical (independence of mgb's from $\tilde{x}$). The last definition appears to be more amenable to automatic checking, because, as mentioned, a mgb can be effectively computed in many cases. We also offer an "optimized" version of the quantitative notion, by which it is sufficient to check zero-leakage relatively to uniformly distributed and independent $\tilde{X}$ and $\tilde{Y}$.

Theorem 5.1 (Secrecy). *Let $P(\tilde{x}, \tilde{y})$ be an open process. The following assertions are equivalent*:

1. *$P(\tilde{x}, \tilde{y})$ keeps $\tilde{x}$ secret given $\tilde{y}$.*
2. *For some $\tilde{X}^* : \tilde{U}$ and $\tilde{Y}^* : \tilde{V}$ uniformly distributed and independent $\mathcal{A}(P\,;\,\tilde{X}^*\ \mid\ \tilde{Y}^*) = 0$.*
3. $\max_{\tilde{X}:\tilde{U},\,\tilde{Y}:\tilde{V}}\ \mathcal{A}(P\,;\,\tilde{X}\ \mid\ \tilde{Y})\ =\ 0$.
4. *$\phi \Leftrightarrow \exists\tilde{x}\tilde{x}'.\phi$, where $\phi = \mathrm{mgb}\Big(P(\tilde{x}, \tilde{y}),\ P(\tilde{x}', \tilde{y}')\Big)$, for $\tilde{x}'$ and $\tilde{y}'$ tuples of distinct variables disjoint from $\tilde{x}$ and $\tilde{y}$, but of the same type.*

Proof. We show that (4) ⇒ (3) ⇒ (2) ⇒ (1) ⇒ (4). In what follows, for ease of notation, we will denote by $\tilde{u}, \tilde{u}', \ldots$ generic outcomes of $\tilde{X}$, by $\tilde{v}, \tilde{v}', \ldots$ generic outcomes of $\tilde{Y}$ and by $w, w', \ldots$ generic outcomes of $Z = P(\tilde{X}, \tilde{Y})$. Moreover we shall use such shorthands as $p(\tilde{u})$ for $\Pr(\tilde{X} = \tilde{u})$, $p(w \mid \tilde{u})$ for $\Pr(Z = w \mid \tilde{X} = \tilde{u})$, and so on.

- (4) ⇒ (3). By contradiction, assume for some $\tilde{X}$ and $\tilde{Y}$ it holds $\mathcal{A}(P;\ \tilde{X}\ \mid\ \tilde{Y}) = H(Z\ \mid\ \tilde{Y}) > 0$. By definition of $H(Z\ \mid\ \tilde{Y})$, this implies that there is some $\tilde{v}$ such that $H(Z\ \mid\ \tilde{Y} = \tilde{v}) > 0$. The latter, by definition of conditional entropy, implies that there are at least *two distinct* outcomes of Z corresponding to $\tilde{v}$, say $w_1 = [P(\tilde{u}_1, \tilde{v})]_{\asymp}$ and $w_2 = [P(\tilde{u}_2, \tilde{v})]_{\asymp}$ (and it also implies that $1 > p(w_i \mid \tilde{v}) > 0$, for $i = 1, 2$). That is, $P(\tilde{u}_1, \tilde{v}) \not\asymp P(\tilde{u}_2, \tilde{v})$. Now, consider the substitution $\sigma = [^{\tilde{u}_1}\!/\!_{\tilde{x}},\ ^{\tilde{u}_2}\!/\!_{\tilde{x}'},\ ^{\tilde{v}}\!/\!_{\tilde{y}},\ ^{\tilde{v}}\!/\!_{\tilde{y}'}]$. By definition of the mgb ϕ, we have that $\sigma \models \exists\tilde{x}\tilde{x}'.\phi$ (that is, $\phi\sigma$ is a tautology), while $\sigma \not\models \phi$: this contradicts $\phi \Leftrightarrow \exists\tilde{x}\tilde{x}'.\phi$.
- (3) ⇒ (2). Obvious.
- (2) ⇒ (1). By contradiction, assume $P(\tilde{x}, \tilde{y})$ does not keep $\tilde{x}$ secret given $\tilde{y}$. In other words, assume there are $\tilde{v}$ and distinct $\tilde{u}_1, \tilde{u}_2$ such that $P(\tilde{u}_1, \tilde{v}) \not\asymp P(\tilde{u}_2, \tilde{v})$. Let $w_i = [P(\tilde{u}_i, \tilde{v})]_{\asymp}$, for $i = 1, 2$. By independence and uniform distribution of $\tilde{X}^*$ and $\tilde{Y}^*$, we have that, for both $i = 1, 2$:

$$p(w_i \mid \tilde{v}) = \sum_{\tilde{u}:\tilde{U}} p(w_i \mid \tilde{u}, \tilde{v}) \cdot p(\tilde{u}) \geq p(w_i \mid \tilde{u}_i, \tilde{v}) \cdot p(\tilde{u}_i) = p(\tilde{u}_i) > 0$$

  (in the rightmost equality above we have used the fact that $p(w_i \mid \tilde{u}_i, \tilde{v}) = 1$). This inequality also implies that $p(w_i \mid \tilde{v}) < 1$, for $i = 1, 2$. Thus we have shown that $0 < p(w_i \mid \tilde{v}) < 1$, for $i = 1, 2$. By definition of conditional entropy, this implies that $H(Z \mid \tilde{Y} = \tilde{v}) > 0$, hence $H(Z \mid \tilde{Y}) > 0$, as $p(\tilde{v}) > 0$. This fact contradicts the assumption that $\mathcal{A}(P;\ \tilde{X}\ \mid\ \tilde{Y}) = 0$.
- (1) ⇒ (4). By contradiction, assume (4) does not hold. Hence it must be $\exists\tilde{x}\tilde{x}'.\phi \not\Rightarrow \phi$, as the opposite logical implication always holds. This means that there is a substitution σ with $\mathrm{dom}(\sigma) \supseteq \mathrm{fv}(\phi)$ s.t. $\sigma \models \exists\tilde{x}\tilde{x}'.\phi$ and $\sigma \not\models \phi$. Let $\tilde{u} = \sigma(\tilde{x})$ and $\tilde{u}' = \sigma(\tilde{x}')$, and $\tilde{v} = \sigma(\tilde{y})$. Then, by definition of mgb, $P(\tilde{u}, \tilde{v}) \not\asymp P(\tilde{u}', \tilde{v})$, which contradicts the assumption. □

Example 5.4. Consider the following process, where $x, y : 1..4$.

$$Q(x,y) \overset{\text{def}}{=} (\nu c)\Big(\bar{c} \mid [y=1]c.\bar{a}\Big) \,+\, [x=2]\tau.\bar{a}\,.$$

It is immediate to see that Q does not keep x secret, given y. E.g., if the adversary gets to know that $y \neq 1$ and observes the behaviour $[\tau.\bar{a}]_{\asymp}$ then he/she can infer that $x = 2$. In fact, the mgb given by the theorem above is in this case

$$\phi = \Big([y=1] \vee [x=2]\Big) \leftrightarrow \Big([y'=1] \vee [x'=2]\Big)$$

and clearly, $\phi \not\Leftrightarrow \exists xx'.\phi$. As an example, for X, Y independent and u.d on $1..4$, the leakage from X to Q given Y can be computed as follows. Let $Z = P(X, Y)$.

- If $Y = 1$ then Z does not depend on X, as for all i and j: $Q(i,1) \asymp Q(j,1) \asymp \tau.a$. Hence $H(Z \mid Y = 1) = 0$;
- If $Y = i \neq 1$ then if $X = 2$ (which happens with probability $\frac{1}{4}$) then $Z = [\tau.a]_{\asymp}$, otherwise $Z = [\mathbf{0}]_{\asymp}$. Hence $H(Z \mid Y = i) = \mathcal{B}(\frac{1}{4})$, for $i \neq 1$.

As a consequence

$$H(Z \mid Y) = \sum_{i=1}^{4} H(Z \mid Y = i) \cdot \Pr(Y = i) = \mathcal{B}\left(\frac{1}{4}\right) \cdot \frac{3}{4} \approx 0.608.$$

The process $Q'(x,y) = Q(x,y) + [y \neq 1]\tau.\bar{a}$ keeps x secret given y.

The next example shows a simple form of timing-dependent leakage.

Example 5.5 (*Modular exponentiation*)**.** The modular exponentiation algorithm, used in implementations of public-key cryptographic schemes for computing powers $a^x \bmod n$, can be described as follows. Let $\tilde{x} = (x_{k-1}, \ldots, x_0)$ be the binary representation of a (secret) k-bit exponent x and `A` an integer variable initially storing 1. The final value of `A` is that returned by the algorithm:

$E(\tilde{x}) \overset{\text{def}}{=}$ `for` $i = k-1$ `downto 0 do {A=A`^{2} `mod` n `; if` x_i `=1 then A=`a`*A mod` n `}.`

We consider two different abstract versions of E, where just the elapse of time can be observed. The basic operations of E are squaring `A=A`^{2} `mod` n and multiplication `A=`a`*A mod` n. Assume that an attacker can observe the duration of individual executions of such operations (admittedly, a strong assumption). Assume further that there is a discrete range of durations, hence it is possible to represent each duration as a distinct visible action. In the first abstract version of E, we suppose that the time taken by each operation is a constant t (below, `for` is just used as syntactic sugar):

$E_1(\tilde{x}) \overset{\text{def}}{=}$ `for` $i = k-1$ `downto 0 do` $(t.[x_i = 1]t)$.

In the second version, each squaring operation takes t_1, and each multiplication t_2:

$E_2(\tilde{x}) \overset{\text{def}}{=}$ `for` $i = k-1$ `downto 0 do` $(t_1.[x_i = 1]t_2)$.

It is easy to see that $E_1(\tilde{u}) \asymp E_1(\tilde{v})$ if and only if $\tilde{u}$ and $\tilde{v}$ have the same number of 1 digits (the same Hamming weight), which makes entropy easy to determine analytically if $\tilde{X}$ is u.d[2]. E.g. assuming $k = 4$, we get $H(E_1(\tilde{X})) \approx 2.03$. Not surprisingly, E_2 leaks all information about $\tilde{X}$, as a t_2 action at iteration number i is observed if and only if $X_i = 1$: hence $H(E_2(\tilde{X})) = H(\tilde{X})$. Under the assumptions above ($k = 4$, $\tilde{X}$ u.d), this value is 4.

5.2. Behavioural equivalences and attacker's observational power

To a large extent, our results on absolute leakage do not depend on the choice of the behavioural equivalence $\asymp$ – contrary to the case of rate of leakage, which we study in Section 7, where we will have to commit to trace equivalence. Even the numeric values in the examples we have considered so far do not depend on the choice of $\asymp$, as trace equivalence and strong bisimilarity, the two extremes in between which $\asymp$ is supposed to lie, coincide in those cases.

In general, however, choosing a specific equivalence amounts to assigning a specific observational power to the attacker: the finer (more discriminating) the equivalence, the stronger the observational power of the attacker, that is, his/her ability

[2] More precisely, $E_1(\tilde{X})$ has $k+1$ possible outcomes; the outcome corresponding to an exponent X with i "1" digits has probability $p_i = \frac{\binom{k}{i}}{2^k}$, for $i \in 0..k$.

to tell apart different behaviours induced by different outcomes of X. This is the content of the next proposition. In what follows, for notational simplicity, we consider individual r.v.'s X and Y rather than vectors of them. We shall indicate by $\mathcal{A}_{\asymp_i}$ the leakage function computed when $\asymp$ is set to the equivalence $\asymp_i$. Similarly, we indicate by $P_{\asymp_i}(X; Y)$ the r.v. induced by P, X and Y when setting $\asymp$ to $\asymp_i$.

Proposition 5.2. *Let $\asymp_1$ and $\asymp_2$ be two behavioural equivalences over closed processes and suppose $\asymp_1 \subseteq \asymp_2$. Let $P(x,y)$ be an open process and X, Y be r.v. Then $\mathcal{A}_{\asymp_1}(P\,;\,X \mid Y) \geq \mathcal{A}_{\asymp_2}(P\,;\,X \mid Y)$.*

Proof. Let U be the set of outcomes of X. For generic total functions f and g defined over U, let us write

$$f \leq g \quad \text{iff} \quad \text{for each } u \text{ and } v,\, g(u) = g(v) \text{ implies } f(u) = f(v).$$

Equivalently, $f \leq g$ iff for each $u \in U$ there is a set $V \subseteq U$ s.t. $f^{-1}(u) = \cup_{v \in V}\, g^{-1}(v)$. Note that if $f \leq g$ then $g(u)$ determines $f(u)$: indeed, f maps any element of $g^{-1}(g(u))$ to one and the same $f(u)$. Now, for any two random variables R_1 and R_2, we have the following equality, which is a consequence of the chain rule

$$H(R_1) = H(R_2) + H(R_1 \mid R_2) - H(R_2 \mid R_1)\,.$$

Applying the above equality to $R_1 = g(X)$ and $R_2 = f(X)$, we get that $H(g(X)) = H(f(X)) + H(g(X) \mid f(X))$, as $H(f(X) \mid g(X)) = 0$: indeed, the value of $g(X)$ determines that of $f(X)$. Hence, we have obtained that

$$f \leq g \quad \text{implies} \quad H(g(X)) \geq H(f(X))\,.$$

Now, fix any outcome v of Y and consider the functions $g : u \mapsto [P(u,v)]_{\asymp_1}$ and $f : u \mapsto [P(u,v)]_{\asymp_2}$. Clearly $f \leq g$. Applying the inequality above, we get $H(P_{\asymp_1}(X,v)) \geq H(P_{\asymp_2}(X,v))$. But $H(P_{\asymp_i}(X,v)) = H(P_{\asymp_i}(X,Y) \mid Y = v)$, for $i = 1, 2$, so we have actually shown that

$$H(P_{\asymp_1}(X,Y) \mid Y = v) \geq H(P_{\asymp_2}(X,Y) \mid Y = v)\,.$$

Averaging on all v's, we get $H(P_{\asymp_1}(X,Y) \mid Y) \geq H(P_{\asymp_2}(X,Y) \mid Y)$, that is, by Lemma 5.1, $\mathcal{A}_{\asymp_1}(P\,;\,X \mid Y) \geq \mathcal{A}_{\asymp_2}(P\,;\,X \mid Y)$. □

We give below a simple example involving strong bisimulation $\sim$ and trace equivalence $\simeq$.

Example 5.6. Recall that $\simeq$ takes into account only sequences of (weak) traces: indeed, $P \simeq Q$ holds true if and only if for each trace s, $P \overset{s}{\Longrightarrow}$ iff $Q \overset{s}{\Longrightarrow}$. On the other hand, strong bisimilarity $\sim$ takes into account the branching structure arising from nondeterminism, and is more discriminating than trace equivalence. Specifically, $\sim$ is defined as the largest equivalence relation over closed processes such that whenever $P \sim Q$ and $P \xrightarrow{\mu} P'$ then there is a transition $Q \xrightarrow{\mu} Q'$ such that $P' \sim Q'$. (Another difference between these two semantics is that trace equivalence is τ-abstracting while strong bisimilarity is not, but this fact is not going to play a role in the example below.) Consider now

$$P(x) \overset{\text{def}}{=} a.b \;+\; [x = 0]a$$

where $x : 0..1$. Take X u.d. on $0..1$. Let us set $\asymp$ to testing equivalence $\simeq$. What is $\mathcal{A}(P\,;\,X)$? Clearly, $P(0) \simeq P(1) \simeq a.b$, thus the function $i \mapsto [P(i)]_{\simeq}$, for $i = 0, 1$, is a constant and $\mathcal{A}(P\,;\,X) = H\Big(P(X)\Big) = 0$. Let us now set $\asymp$ to $\sim$. It is immediate to check that $P(0) \not\sim P(1)$: indeed, $P(0) \xrightarrow{a} \mathbf{0}$, a move that $P(1)$ cannot simulate. As a consequence, in this case $P(X)$ takes two distinct values, $[P(0)]_{\sim}$ and $[P(1)]_{\sim}$, each with probability $\frac{1}{2}$. Hence, $\mathcal{A}(P\,;\,X) = H\Big(P(X)\Big) = 1$.

6. Absolute leakage in relation to other security measures

The use of entropy as a measure of uncertainty in Cryptography dates back to Shannon [30]. The relationship of Shannon entropy to "guessing difficulty" is also somehow folklore: the higher the entropy, the more difficult for an attacker to correctly guess, say, a secret key (see [3]). Although the *coincidence* of entropy and guessing difficulty has been questioned (see e.g. [27]), there is no doubt that these two notions are intimately connected, as witnessed by certain results in Information Theory. Below, we review these results and use them to relate absolute leakage to certain security measures that account for either the error probability or the guessing effort of an attacker that tries to infer sensitive information from P.

Before examining those results closely, though, it is important to stress one general reason why Shannon's entropy may be (and in fact is) preferred to other, more direct metrics of guessing difficulty. This reason lies in the nice additivity properties of entropy, as expressed by the chain rule. In this respect, an instance of the chain rule called *grouping law* is illuminating. The grouping law states that given any partition $U_1, \ldots, U_n$ of the state-space U of X, the uncertainty on X can be decomposed into the uncertainty as to what block of the partition X belongs to, plus the uncertainty on which element of that block X is. Formally, once we define the r.v. $Y = i$ iff $X \in U_i$, we have that[3]

$$H(X) = H(Y) + H(X \mid Y) . \tag{9}$$

In our model, laws of this kind make it possible to establish forms of compositional reasoning, as discussed in the preceding section. As a direct example of application of (9) to our model, consider the following. It is easy to show that (for any instantiation of x) $P(x)$ can be re-written modulo $\asymp$ into a head-normal form $\sum_{i=1}^{n} \phi_i P_i(x)$, with the property that the ϕ_i's form a partition of truth (that is, $\phi_i \wedge \phi_j \Rightarrow$ *false* for $i \neq j$, and $\bigvee_{i=1}^{n} \phi \Leftrightarrow$ *true*; see e.g. [6]). Assume that the partition over the data determined by the function $u \mapsto [P(u)]_{\asymp}$ is finer than the partition determined by the ϕ_i's (that is, for each $[P(u)]_{\asymp}$ there is a ϕ_j s.t. $\forall v, P(v) \asymp P(u)$ implies $\phi(v)$). Define the r.v. Y as $Y = i$ iff $\phi_i(X) =$ *true*. This way, each outcome i of Y determines a set of possible equivalence classes $[P(u)]_{\asymp}$. Then, by (9), we get (we let $p_i = \Pr(Y = i)$)

$$H\Big(P(\tilde{X})\Big) = H(Y) + H\Big(P(X) \mid Y\Big) = H(\tilde{p}) + \sum_{i=1}^{n} p_i H\Big(P_i(X \mid Y = i)\Big)$$

by which the problem of computing the entropy of $P(X)$ is reduced to the problem of computing the probabilities p_i's (which are easy to estimate accurately) and the conditional entropy of the subterms $P_i(X \mid Y = i)$. It is worth to stress that similar additivity properties are not found in connection to other, reasonable security measures, such as those considered below (see [27] for a discussion).

In what follows, for notational simplicity, we shall consider a single r.v. X, rather than a vector, and assume that no side-information Y is available.

6.1. Error probability

Generally speaking, given a r.v. X with outcomes in U and an r.v. Z with outcomes in V, one can define the error probability of inferring X from Z under an optimal "guessing function" g, thus

$$\varepsilon_{X,Z} \stackrel{\text{def}}{=} \inf_{g:V \to U} \Pr(g(Z) \neq X) .$$

It can be shown that the above inf is in fact a minimum, attained when g fulfills the *Maximum a Posteriori Probability (MAP)* rule. This rule dictates that, for each possible outcome v of Z, $u = g(v)$ should maximize $\Pr(X = u \mid Z = v)$. *Fano's inequality* [13] sets a lower bound on $\varepsilon_{X,Z}$ in terms of the uncertainty on X after observing Z, that is $H(X \mid Z)$:

$$\varepsilon_{X,Z} \geq \frac{H(X \mid Z) - 1}{\log |X|} . \tag{10}$$

As expected, the higher the uncertainty, the higher the error probability. It is then immediate to convert upper bounds on absolute leakage into lower bounds on the attacker's error probability of guessing X after observing Z. Let $Z = P(X)$. By definition of leakage as mutual information between X and Z, we have that $H(X \mid Z) = H(X) - \mathcal{A}(P\,;\,X) = H(X) - H(Z)$. This expression can be plugged into formula (10), which can then be used to lower-bound $\varepsilon_{X,Z}$ in terms of absolute leakage.

For instance, in the case of the modular exponentiation algorithm with an exponent of $k = 4$ bits chosen at random (Example 5.5), which exhibits an absolute leakage of 2.03 bits, one gets $\varepsilon_{X,Z} \geq (4 - 2.03 - 1)/2 \approx 0.485$.

It is worth to notice that inequalities are also known that give tight upper bounds on error probability as a function of the conditional entropy (see e.g. [9] for a survey and recent results on upper bounds).

6.2. Guesswork

Consider now a slightly different situation. The attacker is given an oracle that answers (multiple) queries of the form "$X = u$?". In the absence of any extra information on X, the most effective strategy for the attacker is to submit to the oracle guesses for X, from the most likely down to the least likely, stopping as soon as a "yes" answer is received (this is what is called a *dictionary attack* in password security). Let $\tilde{p} = p_1, \ldots, p_n$ be the distribution of X, with the probabilities p_i's ordered from the greatest to the smallest. The average number of queries before correctly guessing X, the *guesswork* of X, is defined by

$$G(X) \stackrel{\text{def}}{=} \sum_{i=1}^{n} i p_i$$

[3] To see that this equation is a consequence of the chain rule, note that, by the chain rule, for any X and Y, $H(X) = H(Y) + H(X|Y) - H(Y|X)$. If Y is a function of X, like in the case considered here, $H(Y|X) = 0$.

and can be taken as a security measure relative to X. When X is u.d., there is a clear relationship between Shannon and guesswork, given by $G(X) = \frac{n+1}{2} = \frac{2^{H(X)}+1}{2}$. More generally, Massey [25] has proven that, if X has at least 2 bits of entropy:

$$G(X) \geq \frac{2^{H(X)}+1}{e}\,. \tag{11}$$

Consider now equipping the adversary of our model with the oracle described above. Assume in full generality that $Z = F(X)$ for some function F (in our model, $F : u \mapsto [P(u)]_{\asymp}$). The attacker can take advantage of both the oracle and Z: rather than querying the oracle blindly, he/she can restrict his/her search to those values of X that are consistent with the observed value of Z. If $Z = v$, only those $u \in F^{-1}(v)$ are worth to be submitted to the oracle. Measuring the security of this system calls then for a conditional definition of guesswork, $G(X \mid Z)$. The guesswork of X given $Z = v$, written $G(X \mid v)$, is just the guesswork of the conditional distribution (using a concise notation) $p_{X|Z}(u_1 \mid v), \ldots, p_{X|Z}(u_m \mid v)$, while the guesswork of X given Z can be defined as the average

$$G(X \mid Z) \overset{\text{def}}{=} \sum_v G(X \mid v) p_Z(v)$$

which we can take as a security measure. We can express a lower bound on this quantity in terms of the absolute leakage $H(Z) = \mathcal{A}(P\,;\,X)$, as follows:

$$\begin{aligned}
G(X \mid Z) &= \textstyle\sum_v G(X|v) p_Z(v) \\
&\geq \textstyle\sum_v p_Z(v)(2^{H(X|v)}+1)/e \qquad \text{by (11)} \\
&= (1 + \textstyle\sum_v 2^{H(X|v)} p_Z(v))/e \\
&\geq (1 + 2^{\sum_v H(X|v) p_Z(v)})/e \qquad \text{by Jensen's inequality [13]} \\
&= (1 + 2^{H(X|Z)})/e \\
&= (1 + 2^{H(X)-H(Z)})/e\,.
\end{aligned}$$

(when applying Jensen's inequality above we have exploited the convexity of the function 2^x). The above inequality can be used to convert upper bounds on absolute leakage into lower bounds on conditional guesswork. For instance, in the case of the modular exponentiation algorithm with an exponent X of $k = 4$ bits chosen at random (Example 5.5), which exhibits an absolute leakage of 2.03, one gets $G(X \mid Z) \geq \frac{2^{4-2.03}+1}{e} \approx 1.81$. This value should be compared with the value of a priori guesswork for X, $G(X) = 8.5$.

7. Rate of leakage

In the scenario considered in the previous section, the attacker is granted unrestricted interaction capability to the observed process P. The PIN-checking example suggests a refinement of this situation, that we study in this section. We will assume the attacker can only conduct upon P repeated experiments, each yielding a binary[4] answer, say success or failure. We are interested in the number of communications with the observed process that are *necessary* for the adversary to extract one bit of information about $\tilde{X}$ in this way. In other words, we are interested in the maximal number of bits per visible action conveyed by P: the *rate* at which P leaks information.

In the rest of the section, we fix $\asymp$ to be *trace equivalence* (aka *may testing* equivalence [14,5]), whose definition we recall below for the reader's convenience.

Definition 7.1 (*Trace equivalence*)**.** $P \simeq Q$ iff for each trace s, $P \overset{s}{\Longrightarrow}$ iff $Q \overset{s}{\Longrightarrow}$.

For the sake of presentation, we shall only consider processes where channels transport tuples of values, i.e. we ban name-passing in the rest of the section. The extension to the name-passing case is dealt with in Appendix 9. For simplicity, we shall also assume that no side-information is available to the attacker, i.e. that $\tilde{y}$ is empty. Hence, throughout the section and unless otherwise stated, $P(\tilde{x})$, where $\tilde{x} : \tilde{U}$, denotes an arbitrary open pi-process, $\tilde{X}$ an arbitrary vector of random variables of type $\tilde{U}$ and Z the r.v. $P(\tilde{X})$. Recall that $\mathcal{A}(P\,;\,\tilde{X}) = H(Z)$.

[4] We expect no significant change in the theory if k-ary answers, with $k > 2$ fixed, were instead considered.

7.1. Definitions and basic properties

Consistently with the testing equivalence framework [14,5], we view an experiment as a *test process* T that, when run in parallel with P, may succeed or not. Input on a distinct name ω, carrying no objects, is used to signal *success* to the adversary. It is convenient here to adjust the notion of (test-process) composition ($\|$) so as to ensure that, in case of success, exactly one success action is reported to the adversary.

Definition 7.2 (*Test processes and composition*)**.** A *test* T is a closed process formed without using process identifiers and possibly using a distinct success action ω. For each test T and closed process Q, define

$$Q\|T \overset{\text{def}}{=} (\nu\tilde{c},\omega')(Q|T[^{\omega'}/_{\omega}]|\overline{\omega'}.\omega)$$

where $\tilde{c} = \text{fn}(Q,T) \setminus \{\omega\}$ and $\omega' \notin \text{fn}(Q,\omega)$.

Note that, for each T and closed Q, it must be either $Q\|T \simeq \mathbf{0}$ – meaning that T fails on Q – or $Q\|T \simeq \omega.\mathbf{0}$ – meaning that T succeeds on Q. Hence, each $P(\tilde{x})$, T and $\tilde{X}$ determine a binary random variable

$$(P\|T)(\tilde{X})$$

also written $P(\tilde{X})\|T$, with possible outcomes $\mathbf{F} \overset{\text{def}}{=} [\mathbf{0}]_{\simeq}$ and $\mathbf{S} \overset{\text{def}}{=} [\omega.\mathbf{0}]_{\simeq}$, to be interpreted as failure and success. Information on $\tilde{X}$ conveyed by $P(\tilde{X})||T$ is given by

$$I\Big(\tilde{X}\,;\; P(\tilde{X})\|T\Big) = H\Big(P(\tilde{X})\|T\Big) - H\Big(P(\tilde{X})\|T \mid \tilde{X}\Big) = H\Big(P(\tilde{X})\|T\Big)$$

and is, of course, at most one bit. The notion of rate we are after should involve a ratio between this quantity of information and the *cost* of performing the test T. The following example provides some indications as to what it should be intended by cost, and shows the role played by non-determinism in extracting information out of P.

Example 7.1. Consider again $Check(X)$, where this time X is u.d. over $1..k$, for some fixed even integer $k \geq 2$. A test T that extracts one bit out of $Check(X)$ is

$$T \overset{\text{def}}{=} \sum_{d=1}^{k/2} \overline{a}d.ok.\omega\,.$$

An attacker can only observe the outcome of the interaction between $Check$ and T, i.e. an outcome of the r.v. $Check(X)\|T$. If action ω is observed, then it must be $X \leq k/2$; if action ω is not observed, then it must be $X > k/2$. The information provided by T is $I(X\,;\, P(X)||T) = H(P(X)||T) = \mathcal{B}(\frac{1}{2}) = 1$.

The above example suggests that the test's traces that may lead to success, that is traces ending with a ω, should be counted as the different "trials" attempted by the attacker. The success or failure of each trial gives the attacker some amount of information. The cost of each trial can be assumed to be proportional to its length as a trace. These considerations motivate the following definition.

Definition 7.3 (*Cost of T*)**.** Let T be a test. We say that a nonempty trace of visible actions $s = \alpha_1 \cdots \alpha_n$ ($\alpha_i \neq \omega$) *may lead T to success* if $T \overset{s\cdot\omega}{\Longrightarrow}$. Denote by $|s|$ the length of a trace s. The *cost* of T is

$$|T| \overset{\text{def}}{=} \sum_{s\,:\, s \text{ may lead } T \text{ to success}} |s|\,.$$

According to the definition above, all traces that have a *chance* of leading T to success, when T is run with *some P*, must be counted – only "dummy" traces, never leading to ω, are discarded. Consider e.g. $T = a.b.\omega + c.d + e.\omega$: we have $|T| = 3$, as the dummy trace $c \cdot d$ is discarded, while e and $a.b$ are counted. When composing T with $P(x) = [x = 0]\overline{a}.\overline{b}$, e cannot be used to lead $P\|T$ to success, while $a \cdot b$ leads to success only in case $x = 0$. In other words, while excluding dummy traces, our definition of cost does include traces corresponding to failed attempts, as they give some information to the attacker. We arrive now at the definition of leakage rate.

Definition 7.4 (*Rate of leakage*)**.** Let $P(\tilde{x})$ be an open process and $\tilde{X}$ be a tuple of r.v. The *leakage rate of P relative to* $\tilde{X}$ is

$$\mathcal{R}(P\,;\,\tilde{X}) \overset{\text{def}}{=} \sup_{|T|>0} \frac{H\Big(P(\tilde{X})\|T\Big)}{|T|}\,. \tag{12}$$

Our first result is a test-independent characterization of rate. In accordance with the may-testing approach, this characterization is obtained in terms of observations of individual traces of processes. The practical significance of this result is that, when computing rate of P, we are relieved from checking P against *every* T, and just have to look at P's traces (which are finitely many, if P is finite). In what follows, given a trace of visible actions s, we consider the r.v. $P(\tilde{X}) \overset{s}{\Longrightarrow}$, with possible outcomes *true* or *false*. As an example, if $P(x) = a.[x = 0]a.\mathbf{0}$ and X is u.d. on $\{0, 1\}$, then $\Pr\left(P(X) \overset{aa}{\Longrightarrow} = \mathit{true}\right) = \Pr(X = 0) = \frac{1}{2}$. We shall make extensive use of the fact that both $|$ and $\|$ distribute over nondeterministic choice, that is, for any closed Q, R_1 and R_2 we have

$$Q \mid (R_1 + R_2) \ \simeq\ (Q|R_1) \ +\ (Q|R_2)$$

and similarly for $\|$. This property does not hold, in general, for behavioural equivalences different from $\simeq$ (e.g., it does not hold for bisimulation).

Proposition 7.1. *Let $P(\tilde{x})$ be an open process and $\tilde{X}$ be a tuple of r.v. It holds that*

$$\mathcal{R}(P\,;\,\tilde{X}) \ =\ \sup_{|s|>0} \frac{H\left(P(\tilde{X}) \overset{s}{\Longrightarrow}\right)}{|s|}. \tag{13}$$

Proof. For any trace s, let $\hat{s}$ be a test of cost $|s|$ such that for each closed Q, $Q\|\hat{s} \overset{\omega}{\Longrightarrow}$ iff $Q \overset{s}{\Longrightarrow}$ (see Appendix 9, where $\hat{s}$ is defined in the case of the pi-calculus). Hence, $P(\tilde{X}) \overset{s}{\Longrightarrow}$ is equivalent to $P(\tilde{X})\|\hat{s}$ as a r.v. This fact implies that the RHS of (13) is not greater than the LHS. For the opposite inequality, fix any T with $|T| > 0$ and let S be the set of traces that may lead T to success. For any s, denote by $\bar{s}$ the complementary trace obtained by inverting input and output (e.g., if $s = ad \cdot \overline{b}d'$ then $\bar{s} = \overline{a}d \cdot bd'$). By simple $\simeq$-preserving transformations (see also Appendix 9), we can show that, for any closed Q:

$$Q\|T \simeq Q\| \sum_{s\in S} \hat{\bar{s}} \simeq \sum_{s\in S} (Q\|\hat{\bar{s}})\,.$$

Hence $P(\tilde{X})\|T$ is equivalent to $\sum_{s\in S}(P(\tilde{X})\|\hat{\bar{s}})$ as a r.v. Using this fact and the inequality of Proposition 5.1 with $C[\cdot] = \sum_{s\in S}([\cdot])$, we have that

$$\frac{H\left(P(\tilde{X})||T\right)}{|T|} = \frac{H\left(\sum_{s\in S} P(\tilde{X})\|\hat{\bar{s}}\right)}{|T|} \le \frac{\sum_{s\in S} H(P(\tilde{X})\|\hat{\bar{s}})}{|T|}\,.$$

In the last term, we can replace $P(\tilde{X})\|\hat{\bar{s}}$ by $P(\tilde{X}) \overset{\bar{s}}{\Longrightarrow}$, hence $H(P(\tilde{X})\|\hat{\bar{s}})$ by $H(P(\tilde{X}) \overset{\bar{s}}{\Longrightarrow})$. Moreover, by definition of cost, we can replace $|T|$ by $\sum_{s\in S} |s| = \sum_{s\in S} |\bar{s}|$. The thesis then follows by the inequality below (which holds for generic $M_i > 0, N_i \ge 0$, for $i = 1, ..., k$):

$$\frac{N_1 + \cdots + N_k}{M_1 + \cdots + M_k} \ \le\ \max_{i=1,\ldots,k} \frac{N_i}{M_i}\,. \qquad \square \tag{14}$$

Example 7.2. Consider the process $\mathit{CheckOnce}(x) \overset{\mathrm{def}}{=} a(z).([z = x]\overline{ok} + [z \ne x]\overline{no})$, where $x, z : 1..10$, and X u.d. on the same interval. It is immediate to check that the ratio in (13) is maximized by any of $s = ad \cdot \overline{ok}$ or $s = ad \cdot \overline{no}$, for $d \in 1..10$. This yields $\mathcal{R}(\mathit{CheckOnce}\,;\,X) = \mathcal{B}(\frac{1}{10})/2 \approx 0.234$.

Remark 7.1. The proposition above allows one, at least in principle, to compute the rate of any process having a finite symbolic transition system. This can be seen as follows.

Let $P(\tilde{x})$ be one such process. Relying on P's symbolic transition system, it is possible to compute, for any given trace s, a logical formula $\phi_s(\tilde{x})$ expressing the exact condition on $\tilde{x}$ under which $P(\tilde{x})$ can perform s (we will not discuss the details here – see [19,6]).

By considering all possible traces in this way, one gets a finite set of formulae $\{\phi_1(\tilde{x}), ..., \phi_k(\tilde{x})\}$. The reason why this set is finite is that any ϕ_s can be written as a disjunctive normal form using as atoms the formulae appearing in the symbolic transition system of $P(\tilde{x})$, which is finite.

For each $i = 1, ..., k$, let s_i be the shortest trace associated with ϕ_i, and assume empty traces and tautologies are discarded away. Note that for each trace s_i, the probability that $P(\tilde{X}) \overset{s_i}{\Longrightarrow}$ holds true is simply $\Pr(\phi_i(\tilde{X}) = \mathit{true})$. Thus, $\mathcal{R}(P\,;\,\tilde{X})$ is the greatest among the ratios $\frac{H(\phi_1(\tilde{X}))}{|s_1|}, ..., \frac{H(\phi_k(\tilde{X}))}{|s_k|}$. By statistical methods it is feasible to estimate the probabilities $\Pr(\phi_i(\tilde{X}) = \mathit{true})$, even when it is difficult to obtain explicit expressions for them (e.g., by estimating the fraction of *true* values obtained when repeatedly evaluating $\phi(\tilde{x})$ for values of $\tilde{x}$ drawn according to $\tilde{X}$).

The next result explains the relationship between the notion of rate and absolute leakage. In particular, (a) establishes that $\mathcal{A}$ is the maximal amount of information that can be extracted by *repeated* binary tests; and (b) provides a lower bound on the cost necessary to extract this information, in terms of $\mathcal{R}$ – thus providing a justification for the name "rate". Given a finite sequence of tests $\tilde{T} = T^{(1)}, T^{(2)}, \ldots, T^{(n)}$, we write $|\tilde{T}|$ for its cost $|T^{(1)}| + \cdots + |T^{(n)}|$, and $P(\tilde{X})\|\tilde{T}$ for the sequence of r.v. $P(\tilde{X})\|T^{(1)}, \ldots, P(\tilde{X})\|T^{(n)}$.

Proposition 7.2. *Let $P(\tilde{x})$ be an open process and $\tilde{X}$ be a tuple of r.v. It holds that*

$$(a) \qquad \mathcal{A}(P\,;\,\tilde{X}) \;=\; \max_{\tilde{T}} I(\tilde{X}\,;\,P(\tilde{X})\|\tilde{T})$$

$$(b) \qquad \textit{for each } \tilde{T},\; I(\tilde{X}\,;\,P(\tilde{X})\|\tilde{T}) \;\le\; |\tilde{T}| \cdot \mathcal{R}(P;\tilde{X}).$$

Proof.

(a) Let $\tilde{T} = T^{(1)}, \ldots, T^{(n)}$ be a generic sequence of tests, and let $P(\tilde{X})||\tilde{T} = P(\tilde{X})||T^{(1)}, \ldots, P(\tilde{X})||T^{(n)}$ be the corresponding sequence of r.v. Let $Z = P(\tilde{X})$, hence $\mathcal{A}(P;\tilde{X}) = H(Z)$. By symmetry of I: $I\left(\tilde{X}; P(\tilde{X})||\tilde{T}\right) = H\left(P(\tilde{X})||\tilde{T}\right) \le H(Z)$, where the last inequality stems from the data-processing inequality applied to the function $G : [Q]_{\asymp} \mapsto (\,[Q||T^{(1)}]_{\asymp}, \ldots, [Q||T^{(n)}]_{\asymp}\,)$.
We show that the max can be attained for a suitable choice of $\tilde{T}$. By definition of $\simeq$, an outcome $[Q]_{\asymp}$ of Z can be identified with the set of traces $L = \{s \mid Q \overset{s}{\Longrightarrow}\}$, that is, the *language* generated by Q. For any two distinct outcomes of Z, say L and L', choose a trace $s \in (L \setminus L') \cup (L' \setminus L)$, and let D be the set of all such traces. Clearly, for any two outcomes of Z, say L and L', it holds that $L = L'$ if and only if $D \cap L = D \cap L'$. In other words Z and $Z \cap D$ are equivalent as r.v. We will define a sequence of tests $\tilde{T}$ s.t. $P(\tilde{X})||\tilde{T}$ is equivalent to $Z \cap D$, which will prove the thesis, as $H(P(\tilde{X})||\tilde{T}) = H(Z) = \mathcal{A}(P;\tilde{X})$.
Let us arbitrarily order the elements of D as: $s_1, \ldots, s_n$. A vector in $\{0,1\}^n$ can be identified with a subset of D in the obvious way. For each $s_i \in D$, consider the test $T^{(i)} \overset{\text{def}}{=} \hat{s}_i$ s.t. for any Q, $Q\|\hat{s}_i \overset{\omega}{\Longrightarrow}$ iff $Q \overset{s_i}{\Longrightarrow}$. By identifying 0 with $\mathbf{F} = [\mathbf{0}]_{\asymp}$ and 1 with $\mathbf{S} = [\omega.\mathbf{0}]_{\asymp}$, we see that the r.v. $P(\tilde{X})||\tilde{T} = P(\tilde{X})||\hat{s}_1, \ldots, P(\tilde{X})||\hat{s}_n$ yields outcomes in $\{0,1\}^n$, i.e. on subsets of D. Moreover, by definition of $P(\tilde{X})||\tilde{T}$, the event $(P(\tilde{X})||\tilde{T} = L)$ is the same as the event $(Z \cap D = L)$. In other words, $P(\tilde{X})||\tilde{T}$ and $Z \cap D$ are equivalent as r.v.

(b) Fix any $\tilde{T}$, and assume without loss of generality that $|T^{(i)}| > 0$ for each i. Then we have:

$$\begin{aligned}
I(\tilde{X}; P(\tilde{X})||\tilde{T}) &= H(P(\tilde{X})||\tilde{T}) \\
&\le \sum_{i=1}^{n} H(P(\tilde{X})||T^{(i)}) && \text{(by (3))} \\
&= |\tilde{T}| \cdot \frac{\sum_{i=1}^{n} H(P(\tilde{X})||T^{(i)})}{|\tilde{T}|} \\
&\le |\tilde{T}| \cdot \max_{i=1,\ldots,n} \frac{H\left(P(\tilde{X})||T^{(i)}\right)}{|T^{(i)}|} && \text{(by (14))} \\
&\le |\tilde{T}| \cdot \mathcal{R}(P;\tilde{X})\,.
\end{aligned}$$

□

Note that the cost of extracting *all* available information, $\mathcal{A}(P\,;\,\tilde{X}) = H(Z)$, cannot be less than $\frac{H(Z)}{\mathcal{R}(P;\tilde{X})}$, for $Z = P(\tilde{X})$. It is important to remark that two processes with equal absolute leakage may well exhibit different rates. Here is a small example to illustrate this point.

Example 7.3. Let $P(x)$ and $Q(x)$, where $x : 0..3$, be defined as follows:

$$\begin{array}{lclclclcl}
P(x) & = & [x=0](a+b) & + & [x=1](b+c) & + & [x=2](c+d) & + & [x=3](d+a) \\
Q(x) & = & [x=0]a & + & [x=1]b & + & [x=2]c & + & [x=3]d\,.
\end{array}$$

Assume X is u.d. over 0..3. Both $P(X)$ and $Q(X)$ are u.d. on a domain of four elements: the four distinct equivalence classes $[P(i)]_{\asymp}$, resp. $[Q(i)]_{\asymp}$, for $i \in 0..3$. Hence their absolute leakage is $H(P(X)) = H(Q(X)) = H(X) = 2$ bits. On the other hand, each nonempty trace of P occurs with probability 1/2 (i.e., $\Pr(P(X) \overset{s}{\Longrightarrow}) = 1/2$ for $s \in \{a,b,c,d\}$), while each nonempty trace of Q occurs with probability 1/4 (i.e., $\Pr(Q(X) \overset{s}{\Longrightarrow}) = 1/4$ for $s \in \{a,b,c,d\}$). Thus, by Proposition 7.1, $\mathcal{R}(P;X) = \mathcal{B}(\frac{1}{2}) = 1$ and $\mathcal{R}(Q;X) = \mathcal{B}(\frac{1}{4}) \approx 0.811$. Proposition 7.2(b) then implies that gaining all information about X costs an attacker at least $\frac{2}{1} = 2$ synchronizations in the case of P, and at least $\lceil \frac{2}{0.811} \rceil = 3$ synchronizations in the case of Q. Indeed, the sequence of tests $\overline{a}.\omega$, $\overline{b}.\omega$ is sufficient in the case of P for determining X. The sequence of tests $\overline{a}.\omega$, $\overline{b}.\omega$, $\overline{c}.\omega$ is sufficient in the case of Q.

7.2. Compositionality

The results below are about composing rates of processes. The first proposition gives upper bounds for the rate of a global system in terms of the individual sub-systems. These inequalities can be used for compositional reasoning on rates, although the bounds they provide are sometimes rather loose, especially in the case of restriction (νc). The proof of the proposition is based on simple use of the data-processing inequality plus inequality (14). The subsequent Theorem 7.1 establishes that, under certain conditions, iteration $*$ preserves rate, thus providing another justification for the definition of rate. We regard this as the main result of the section.

Proposition 7.3. *Let $P(\tilde{x})$ and $Q(\tilde{x})$ be open processes and $\tilde{X}$ be a vector of random variables of the same type as $\tilde{x}$. Let $\tilde{e}(\tilde{x})$ be a tuple of expressions and let $\tilde{v}$ be the tuple of values that maximizes* $\mathrm{Pr}(\tilde{e}(\tilde{X}) = \tilde{v})$*. Let $\phi(\tilde{x})$ be a logical formula. Then the following inequalities hold*:

$$\begin{array}{llcl}
\text{(i)} & \mathcal{R}(a(\tilde{z}).P;\tilde{X}) & \leq & \max_{\tilde{u}}\ \mathcal{R}(P[\tilde{u}/\tilde{z}];\tilde{X}) \\
\text{(ii)} & \mathcal{R}(\overline{a}\tilde{e}.P;\tilde{X}) & \leq & \max\{H([\tilde{e}(\tilde{X}) = \tilde{v}]),\ \mathcal{R}(P;\tilde{X})\} \\
\text{(iii)} & \mathcal{R}(\phi P;\tilde{X}) & \leq & H(\phi(\tilde{X})) + \mathcal{R}(P;\tilde{X}) \\
\text{(iv)} & \mathcal{R}((\nu a)P;\tilde{X}) & \leq & \mathcal{R}(P;\tilde{X}) \\
\text{(v)} & \mathcal{R}(P+Q;\tilde{X}) & \leq & \mathcal{R}(P;\tilde{X}) + \mathcal{R}(Q;\tilde{X})\,.
\end{array}$$

Proof. We only cover in detail the case (ii), as the other cases are routine applications of the data processing inequality and/or of inequalities (1–3).

We rely on the trace-based characterization of $\mathcal{R}$ provided by Proposition 7.1. Let s be a generic nonempty trace and $p_s = \mathrm{Pr}\left((\overline{a}\tilde{e}.P)(\tilde{X}) \overset{s}{\Longrightarrow} \right)$. Will show that $\mathcal{B}(p_s)/|s| \leq \max\{H([\tilde{e}(\tilde{X}) = \tilde{v}]),\ \mathcal{R}(P;\tilde{X})\}$. This is obvious if s is not of the form $\overline{a}\tilde{w}\cdot s'$, as in this case $p_s = 0 = \mathcal{B}(p_s)$. Thus, assume $s = \overline{a}\tilde{w}\cdot s'$, for some $\tilde{w}$ and s'. First, note that, by symmetry of the binary entropy function $\mathcal{B}(p)$ around the point $p = 1/2$ (see Fig. 1), the value $\tilde{v}$ that maximizes the probability $\mathrm{Pr}(\tilde{e}(\tilde{X}) = \tilde{v})$ is also the value that maximizes the entropy $\mathcal{B}(\tilde{e}(\tilde{X}) = \tilde{v})$. Second, note that the r.v. $(\overline{a}\tilde{e}.P)(\tilde{X}) \overset{s}{\Longrightarrow}$ is the same as $(\tilde{e}(\tilde{X}) = \tilde{w}) \wedge P(\tilde{X}) \overset{s'}{\Longrightarrow}$. From these two facts, applying the data processing inequality and (3), we have:

$$\begin{aligned}
\mathcal{B}(p_s) \,=\, H\Big((\overline{a}\tilde{e}.P)(\tilde{X}) \overset{s}{\Longrightarrow}\Big) &\leq H\Big(\tilde{e}(\tilde{X}) = \tilde{w}\,,\, P(\tilde{X}) \overset{s'}{\Longrightarrow}\Big) \\
&\leq H\Big([\tilde{e}(\tilde{X}) = \tilde{v}]\Big) + H\Big(P(\tilde{X}) \overset{s'}{\Longrightarrow}\Big)\,.
\end{aligned}$$

If $s' = \varepsilon$ then $H\Big(P(\tilde{X}) \overset{s'}{\Longrightarrow}\Big) = 0$ (as $P(\tilde{X}) \overset{\varepsilon}{\Longrightarrow}$ holds with probability 1), and the thesis follows. Assume $s' \neq \varepsilon$. Dividing by $|s| = 1 + |s'|$ the inequality obtained above and then applying inequality (14), we have

$$\frac{\mathcal{B}(p_s)}{|s|} \leq \frac{H([\tilde{e}(\tilde{X}) = \tilde{v}]) + H(P(\tilde{X}) \overset{s'}{\Longrightarrow})}{1 + |s'|} \leq \max\left\{ H([\tilde{e}(\tilde{X}) = \tilde{v}]),\ \frac{H(P(\tilde{X}) \overset{s'}{\Longrightarrow})}{|s'|} \right\}$$

but $\frac{H(P(\tilde{X}) \overset{s'}{\Longrightarrow})}{|s'|} \leq \mathcal{R}(P;\tilde{X})$ by definition of rate. □

A notable omission from the previous proposition is the case of parallel composition $P|Q$. Interaction between P and Q may give rise to short traces conveying much information on X. Indeed, synchronization may turn visible actions of P and Q into invisible τ's. This might lead $P|Q$ to exhibit a higher rate than the sum of P's and Q's alone, as illustrated below.

Example 7.4. Consider $P(x) = c.[x = 0]a$, $Q(x) = \overline{c}$ with $x : 0..1$. Take X u.d. on 0..1. Clearly, $P(X)$ has a rate of $\mathcal{B}(\frac{1}{2})/2 = \frac{1}{2}$, while the rate of $Q(X)$ is 0, as Q does not actually depend on x. When composing, however, we get for $P|Q$ a rate of 1: indeed, there is an interaction on c that makes trace a available if and only if $x = 0$. Hence $\mathrm{Pr}((P|Q)(X) \overset{a}{\Longrightarrow}) = \frac{1}{2}$, which implies the rate is 1.

In order to define iteration on processes, we have to first define sequential composition. Output on a distinct name *stop*, not carrying objects, is used to signal termination of a thread. We define sequential composition as $P; Q \overset{\text{def}}{=} (\nu\ stop')(P[stop'/stop] \,|\, stop'.Q)$ (with *stop′* fresh). This means that the first thread of P that terminates will trigger execution of Q. This is slightly different from sequential composition in the usual sense, that would require termination of all threads before activating Q. However, the two notions are equivalent in the context we are going to consider (see definition of determinate process below). For any closed process P, let iteration $*P$ be the process recursively defined by $*P \overset{\text{def}}{=} P; *P$. We show that, under a suitable condition, described below, the rate of $*P$ is the same as P's. The condition requires essentially that termination of a single thread in a process is equivalent to termination of the whole process. Its role is that of forbidding "hidden" interactions between threads belonging to different iterations of P in $*P$. We discuss its necessity below (Remark 7.2).

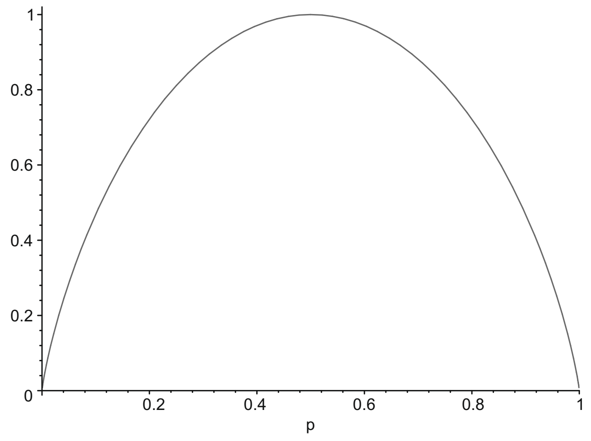


Fig. 1. Binary entropy function $\mathcal{B}(p)$ in $[0, 1]$.

Definition 7.5 (*Determinate processes*). Let Q be a closed process. We say that a trace s is *terminating for* Q if $Q \overset{s\cdot\overline{stop}}{\Longrightarrow}$. We say that Q is *determinate* if for every terminating trace s, whenever $Q \overset{s}{\Longrightarrow} Q'$ then $Q' \simeq \overline{stop}$. Finally, an open process $P(\tilde{x})$ is determinate if $\sum_{\tilde{u}\in\tilde{U}} P(\tilde{u})$ is determinate.

We need another technical condition: let us say that Q is *stable* if whenever $Q \overset{\varepsilon}{\Longrightarrow} Q'$ (ε = empty trace) then $Q' \simeq Q$.

Theorem 7.1 *(*Iteration rate*). Suppose that $P(\tilde{x})$ is determinate, and that for each $\tilde{u}$, $P(\tilde{u})$ is stable. Then $\mathcal{R}({*}P\,;\,\tilde{X}) = \mathcal{R}(P\,;\,\tilde{X})$.*

Proof. We show that for each nonempty s there exists a nonempty s' s.t. $H({*}\,P(\tilde{X}) \overset{s}{\Longrightarrow})/|s| \le H(P(\tilde{X}) \overset{s'}{\Longrightarrow})/|s'|$, that proves, by Proposition 7.1, that $\mathcal{R}({*}P\,;\,\tilde{X}) \le \mathcal{R}(P\,;\,\tilde{X})$. The proof of the opposite inequality is easier and omitted.

We proceed by induction on s. Suppose $|s| > 0$ and let T denote the set of terminating traces of $S \overset{\text{def}}{=} \sum_{\tilde{u}\in\tilde{U}} P(\tilde{u})$. There are two cases for s.

(1) No trace in T is a prefix of s. In this case, it is easy to see that by definition of determinate process, for each $\tilde{u}$, ${*}\,P(\tilde{u}) \overset{s}{\Longrightarrow}$ iff $P(\tilde{u}) \overset{s}{\Longrightarrow}$, hence $P(\tilde{X}) \overset{s}{\Longrightarrow}$ is equivalent to ${*}P(\tilde{X}) \overset{s}{\Longrightarrow}$, and we can take $s' = s$.

(2) There is $s_1 \in T$ that is a prefix of s, say $s = s_1 \cdot s_2$ for some s_2. We can assume that s_1 is not empty (otherwise, $S \simeq \overline{stop}$, and the thesis would follow trivially.) Note that, for each $\tilde{u}$, whenever ${*}\,P(\tilde{u}) \overset{s_1}{\Longrightarrow} P'$ then necessarily $P' \simeq {*}\,P(\tilde{u})$ (a consequence of determinacy and stability of $P(\tilde{u})$). Using this fact, one can prove that for each $\tilde{u}$,

$${*}\,P(\tilde{u}) \overset{s}{\Longrightarrow} \quad \text{if and only if} \quad P(\tilde{u}) \overset{s_1}{\Longrightarrow} \;\text{ and } {*}\,P(\tilde{u}) \overset{s_2}{\Longrightarrow}\;.$$

In other words, the r.v. ${*}\,P(\tilde{X}) \overset{s}{\Longrightarrow}$ is the same as $(P(\tilde{X}) \overset{s_1}{\Longrightarrow} \wedge {*}P(\tilde{X}) \overset{s_2}{\Longrightarrow})$. By virtue of the data-processing inequality and by (3), we obtain

$$H\Big({*}\,P(\tilde{X}) \overset{s}{\Longrightarrow}\Big) \le H\Big(P(\tilde{X}) \overset{s_1}{\Longrightarrow}\Big) + H\Big({*}\,P(\tilde{X}) \overset{s_2}{\Longrightarrow}\Big).$$

Now, if s_2 is empty, that is $s = s_1$, the second term of the summation above is 0, hence dividing by $|s|$ we have $H({*}\,P(\tilde{X}) \overset{s}{\Longrightarrow})/|s| \le H(P(\tilde{X}) \overset{s}{\Longrightarrow})/|s|$, and the thesis follows. Assume now that s_2 is not empty. Dividing the above inequality again by $|s| = |s_1| + |s_2|$ and using (14), we get

$$\frac{H\Big({*}\,P(\tilde{X}) \overset{s}{\Longrightarrow}\Big)}{|s|} \le \max\left\{ \frac{H\Big(P(\tilde{X}) \overset{s_1}{\Longrightarrow}\Big)}{|s_1|}, \frac{H\Big({*}\,P(\tilde{X}) \overset{s_2}{\Longrightarrow}\Big)}{|s_2|} \right\}.$$

If the max is the first of the two terms, we set $s' = s_1$ and stop; otherwise, we invoke induction hypothesis on s_2. □

Example 7.5. It is easy to check that $CheckOnceStop(x) \overset{\text{def}}{=} a(z).([z = x]\overline{ok}.\overline{stop} + [z \neq x]\overline{no}.\overline{stop})$ is determinate ($x : 1..10$). Since $Check(d) \simeq {*}\,CheckOnceStop(d)$, for every d, by Theorem 7.1 and Example 7.2 we have: $\mathcal{R}(Check\,;\,X) = \mathcal{R}(CheckOnceStop\,;\,X) = \mathcal{B}(\frac{1}{10})/2 \approx 0.234$.

Remark 7.2 (*On the necessity of the determinacy condition*)**.** In the absence of determinacy, neither (a) $\mathcal{R}(P\,;\,\tilde{X}) \le \mathcal{R}(*P\,;\,\tilde{X})$ nor (b) $\mathcal{R}(*P\,;\,\tilde{X}) \le \mathcal{R}(P\,;\,\tilde{X})$ hold in general. As a counter-example to inequality (a), consider $P(x) = [x = 0]a.\overline{stop} + [x = 1]a.a.\overline{stop}$ with $x : \{0, 1\}$, which is not determinate. For X u.d. on $\{0, 1\}$, the ratio $\mathcal{B}(p_s)/|s|$ is maximized by the trace $s = a \cdot a$, for which $p_s = 1/2$ (it occurs iff $X = 0$), hence $\mathcal{R}(P\,;\,\tilde{X}) = \frac{\mathcal{B}(1/2)}{2} = 1/2$ (note that the trace $s = a$ yields no information on x, as it can be performed regardless of the value of x). On the other hand, all traces of $*P$ are of the form $a \cdots a$ and occur with probability 1: that is $p_s = 1$ for any trace s of that form, and $p_s = 0$ for traces of a different form. Hence $H(p_s) = 0$ for all s, and $\mathcal{R}(P\,;\,\tilde{X}) = 0$.

As a counter-example to inequality (b), consider $P'(x) = [x = 0]a.b + \overline{stop}.\overline{a} + [x = 1]a$, with $x : \{0, 1\}$, which, again, is not determinate. For X u.d. on $\{0, 1\}$, the ratio $\mathcal{B}(p_s)/|s|$ is maximized by the trace $s = a \cdot b$, for which $p_s = 1/2$ (it occurs iff $X = 0$), hence $\mathcal{R}(P'\,;\,\tilde{X}) = \frac{\mathcal{B}(1/2)}{2} = 1/2$. On the other hand, $*P'$ has a shorter trace $s = b$, which arises from interaction between $\overline{a}$ in P' and $[x = 0]a.b$ in $*P'$ (recall that $*P' \overset{\text{def}}{=} P'; *P'$) and occurs with probability 1/2 (i.e. iff $X = 0$). Hence $\mathcal{R}(*P'\,;\,\tilde{X}) = 1$.

7.3. Rate of leakage and other security measures

It is easy to relate rate of leakage to error probability and guesswork, along the lines of Section 6. We show the details of the error probability case, and just state the result for the case of guesswork, as the details can be easily filled in by the reader.

We consider the attacker's error probability of guessing X after an effort of $N \ge 0$ synchronizations with P. Assuming X has outcomes in U, this probability can be defined as

$$\varepsilon_{X,P,N} \overset{\text{def}}{=} \inf_{g,\tilde{T}\,:\,|\tilde{T}|\le N} \Pr\left(g(P(X)||\tilde{T}) \ne X\right)$$

where g ranges over all functions of type $\{\mathbf{S}, \mathbf{F}\}^* \to U$. That is, g takes a sequence of test outcomes – success or failure – and yields a guess for the value of X.

Like in the case of absolute leakage, we rely on Fano's Inequality (10). For arbitrary but fixed g and $\tilde{T}$ s.t. $|\tilde{T}| \le N$, from (10) we get that:

$$\Pr\left(g(P(X)||\tilde{T}) \ne X\right) \ge \varepsilon_{X,P(X)||\tilde{T}} \ge \frac{H(X \mid P(X)||\tilde{T}) - 1}{\log|X|} = \frac{H(X) - I(X\,;\,P(X)||\tilde{T}) - 1}{\log|X|}.$$

But, by Proposition 7.2(b), $I(X\,;\,\tilde{P}(X)||\tilde{T}) \le N\mathcal{R}(P; X)$, hence we get

$$\Pr\left(g(P(X)||\tilde{T}) \ne X\right) \ge \frac{H(X) - N\mathcal{R}(P; X) - 1}{\log|X|}.$$

Since $\tilde{T}$ and g are arbitrary, we get

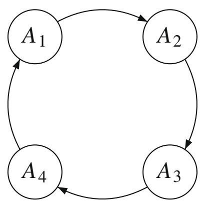
$$\varepsilon_{X,P,N} \ge \frac{H(X) - N\mathcal{R}(P; X) - 1}{\log|X|}.$$

In complete analogy, we can define guesswork for X after N synchronizations with P as $G_{P,N}(X) \overset{\text{def}}{=} \inf_{|\tilde{T}|\le N} G(X \mid P(X)||\tilde{T})$ and prove the following lower bound:

$$G_{P,N}(X) \ge \frac{2^{H(X)-N\mathcal{R}(P;X)} + 1}{e}.$$

8. An extended example

We analyze absolute leakage and rate of leakage of a non-trivial system inspired by – but much simpler than – anonymity protocols in the style of Crowds [28]. A typical goal of these protocols is allowing a group of users to exchange messages over a public network, while hiding the identities of the senders of individual messages from an external (passive) eavesdropper. An essential ingredient to achieve this goal is a routing policy of messages that aims at confounding the eavesdropper as to who is sending to whom at any given moment. Here we consider one such policy for a simple ring-shaped network and quantify the average information leaked to the eavesdropper about the sender, the receiver and the message in a run of the protocol. The average is taken with respect to the random choice of the sender, of the receiver and of the message. In the three subsections below, we give a description of the system and then discuss its absolute leakage and rate of leakage. In that discussion, for the sake of readability we have preferred not to dwell on technical details, which can be found in Appendix 9.



$$\begin{tikzcd} A_1 \arrow[r, bend left] & A_2 \arrow[d, bend left] \\ A_4 \arrow[u, bend left] & A_3 \arrow[l, bend left] \end{tikzcd}$$

Fig. 2. A ring-shaped network with $N = 4$ nodes.

8.1. Description of the system

A set of $N \geq 2$ nodes $A_1, ..., A_N$ are connected through N public, unidirectional channels so as to form a ring-shaped network. This is shown in Fig. 2 for $N = 4$. The purpose of a run of the protocol is to let a sender node A_s transmit a one-bit message $m \in \{0, 1\}$ to a receiver node A_r. The pieces of information represented by r, m and s should be concealed. Since A_s and A_r may possibly be not directly connected, the message m may have to be routed through intermediate nodes. The protocol consists of N stages. To confound the eavesdropper, at each stage, *every* node A_j sends a message to the next node in the ring, A_{j+1} (all indices here are intended mod$N$). More precisely, the node that currently holds the "genuine" message m – initially A_s – sends m, while any other node sends an arbitrarily chosen bit m'. Each A_j must receive a message from A_{j-1} before proceeding to the next stage. Note that after $(r - s)$ mod N stages, message m has actually reached the receiver A_r: the remaining stages are executed for the sole purpose of hiding the relation between s and r.

We make a few assumptions to make our analysis feasible. We analyze leakage due to a *single node* of the network, say A_j with $j \in 1..N$. In other words, we consider a situation where a local attacker eavesdrops on a single node A_j. The attacker can observe incoming and outgoing messages, but cannot tamper with them. He can, however, force the re-execution of the whole protocol with the same parameters (e.g., by fooling participants into believing that some messages sent to A_j have been lost). We furthermore assume that the secrets r, m and s, as well as any routing information needed by them, have somehow been distributed securely to the participants prior to the protocol's execution – that is, we do not model the secure distribution of the secret parameters.

The behaviour of the jth node of the network from the point of view of the attacker is modeled by the process A_j defined below. There, s, r, i are variables of type $1..N$ and m is a variable of type $0..1$. We use input actions in_0 and in_1 (resp. output actions $\overline{out}_0$ and $\overline{out}_1$) to denote the reception (resp. sending) of bits 0 and 1 from the node A_{j-1} (resp. to the node A_{j+1}). We make use of the following notational shorthand. " $\texttt{if}\ \phi\ \texttt{then}\ P\ \texttt{else}\ Q$" stands for $\phi P + \neg\phi Q$; moreover, "$in_x.Q$", where x is a variable of type 0..1, stands for $[x = 0]in_0.Q[^0/_x] + [x = 1]in_1.Q[^1/_x]$; similarly for $\overline{out}_x.Q$. We denote by path(s,j,r) the predicate that is true if and only if node j is in the path from s to r (e.g., path$(3, 4, 1)$ holds true for a configuration of $N = 4$ nodes; note that we set path(s,j,s) to true only if $j = s$). Finally, we consider the predicate holds(s,j,r,i) which tells if node j will hold the genuine message at the beginning of ith stage, counting stages from 0 to $N - 1$; formally

$$\text{holds}(s,j,r,i) \qquad \text{iff} \qquad \text{path}(s,j,r) \text{ and } i = j - s \bmod N\,.$$

E.g. holds$(3, 4, 1, 1)$ holds true in the configuration with 4 nodes.

$B_j(s, r, m, i)$ represents A_j's behaviour from the i^{th} stage onward, counting stages from 0 through $N - 1$.

$$\begin{aligned} A_j(s,r,m) &\stackrel{\text{def}}{=} B_j(s,r,m,0) \\ B_j(s,r,m,i) &\stackrel{\text{def}}{=} (i < N)\Big(\ \texttt{if}\ \text{holds}(s,j,r,i)\ \texttt{then}\ \overline{out}_m \\ &\qquad\qquad \texttt{else}\ \overline{out}_0 + \overline{out}_1 \\ &\qquad\quad \mid\ \texttt{if}\ \text{holds}(s,j-1,r,i)\ \texttt{then}\ in_m.B_j(s,r,m,i+1) \\ &\qquad\qquad \texttt{else}\ in_0.B_j(s,r,m,i+1) + in_1.B_j(s,r,m,i+1)\Big)\,. \end{aligned}$$

The two threads that compose B_j correspond to the following behaviour:

- at each stage, A_j must send a bit to its successor in the ring: if A_j currently holds the genuine message m, then it is this m that will be sent (this is the first `then` branch), otherwise 0 or 1 will be nondeterministically chosen and sent (this is the first `else` branch);
- at each stage, A_j must receive a bit from its predecessor in the ring: if A_{j-1} currently holds the genuine message m, then it is this m that will be received (this is the second `then` branch), otherwise otherwise 0 or 1 will be nondeterministically chosen and received (this is the second `else` branch).

To exemplify the functioning of the node, let us instantiate the above specification of B_j to the case $s = 1, j = 3$ and $r = 4$, for the 4-nodes network in Fig. 8.1. We get the following relations which explain the behaviour of the system (recall that $\sim$

denotes strong bisimilarity):

$$\begin{array}{llcll} B_3(1,4,m,0) & \sim & \overline{out}_0+\overline{out}_1 & \mid & in_0.B_3(1,4,m,1)+in_1.B_3(1,4,m,1)\\ B_3(1,4,m,1) & \sim & \overline{out}_0+\overline{out}_1 & \mid & in_m.B_3(1,4,m,2)\\ B_3(1,4,m,2) & \sim & \overline{out}_m & \mid & in_0.B_3(1,4,m,3)+in_1.B_3(1,4,m,3)\\ B_3(1,4,m,3) & \sim & \overline{out}_0+\overline{out}_1 & \mid & in_0+in_1\,. \end{array} \tag{15}$$

Note that this is a description of A_j's behaviour from the point of view of the attacker. This description says little about the "physical" implementation of the node. Indeed, routing information would normally be found in message headers and not "hardwired" into the processes (see also Remark 8.1 below).

In the analysis below, we assume the receiver r, the sender s and the message m are chosen according to three independent random variables S, R and M, respectively, with S and R uniformly distributed. We let $h = H(M)$ (this is 1 bit if M is chosen at random). We want to analyze absolute leakage and rate of leakage relative to the random variable

$$Z \stackrel{\text{def}}{=} A_j(S,R,M)\,.$$

8.2. Absolute leakage

We first assume the attacker already knows a piece of information Y telling him whether j is in the path from S to R ($Y = true$) or not ($Y = false$). We discuss the two cases separately.

In the case $Y = false$, at each stage both `else` branches are taken, as neither of the two instances of holds$(\cdot)$ ever evaluates to true. As easily seen, the resulting behaviour of B_j, hence of A_j, does not depend on S, R or M. Therefore, in this case the absolute leakage due to A_j is 0, that is, the attacker does not learn anything, apart from the very fact that j is not in the path:

$$H(Z \mid Y = false) = 0\,.$$

In the second case, the attacker, by observing A_j, can tell at which stage i the genuine message m is sent to the successor. Intuitively, in all (re-)executions of the protocol there is a unique stage at which it is always fired *the same* output ($\overline{out}_m$), rather than one of two possible ($\overline{out}_0 + \overline{out}_1$); see e.g. the equations (15) above, where the stage in question is $i = 2$. This way, the attacker can tell the distance between S and j, hence the identity of the sender S, since j is known. As a consequence, he can also tell the value of m, which is directly observed at stage i. He cannot tell the identity of the receiver, though. Therefore, in this case the absolute leakage due to A_j is

$$H(Z \mid Y = true) = H(S) + H(M) = \log N + h\,.$$

Now, j is found in the path from the sender to the receiver approximately in half of the cases, that is the probability that $Y = true$ is about $\frac{1}{2}$ (for large values of N; the exact value is given in the appendix). Hence $H(Y) \approx 1$. The overall leakage can hence be computed as

$$\begin{aligned} \mathcal{A}(A_j; S,R,M) &= H(Z)\\ &= H(Z,Y)\\ &= H(Y) + H(Z \mid Y)\\ &\approx 1 + \tfrac{1}{2}H(Z \mid Y = true) = 1 + \tfrac{1}{2}(\log N + h)\,. \end{aligned}$$

where in the second equality we have used the fact that $H(Z) = H(Z,Y)$, that is, if observing Z, observing also Y does not provide additional information. The above formula can be interpreted as follows: when the number of nodes is large, A_j leaks on the average approximately half of the message content and half of the bits of the sender's identity, plus one bit saying whether A_j is in the path from the sender to the receiver. Intuitively, this is the case because in half of the cases, i.e. when j is not in the sender-receiver path, the attacker cannot say anything about M, S and R – apart from the very fact that j is not in the path – while in the other half of the cases, when j is in the path, the attacker can tell precisely S and M.

8.3. Rate of leakage

For any trace t, let p_t be $\Pr(A_j(S,R,M) \xRightarrow{t})$. The rate we have to estimate is the supremum of

$$\frac{\mathcal{B}(p_t)}{|t|}$$

taken over all nonempty traces t. For reasons explained in the appendix, without loss of generality we can confine ourselves to examining traces where output actions are fired eagerly, that is traces of the form

$$t = \overline{out}_{m_0} \cdot in_{m'_1} \cdot \overline{out}_{m_1} \cdots in_{m'_k} \cdot \overline{out}_{m_k}$$

with $0 \le k \le N-1$ and $m_i, m'_i \in 0..1$. Let us now estimate the probability p_t.

In the first place, it holds that $p_t \ge \frac{1}{2}$, as shown below. Now, the binary entropy function $\mathcal{B}(p)$ attains its maximum in the point $p = 1/2$ (see Fig. 1); hence, in order to maximize $\mathcal{B}(p)/|t|$, while keeping $|t|$ fixed, it is convenient to choose t that minimizes p_t, i.e. makes p_t as close as possible to $\frac{1}{2}$.

As explained below, this is achieved if t is chosen such that, for all $1 \le i \le k$, $m'_i \ne m_i$. By inspection of the code of A_j, such a trace can be performed if and only if the following predicate depending on S, R and M is true (here and in the following we abbreviate holds(S,j,R,i) as holds(i)):

$$\mathrm{cond}_t(S,R,M) = (\neg\mathrm{holds}(0) \vee M = m_0) \wedge \bigwedge_{i=1}^{k} \neg\mathrm{holds}(i)\,.$$

This means: if the node holds the genuine message at stage 0, then the output m_0 observed at stage 0 must be the genuine message; moreover, at none of the k subsequent stages may the node hold the genuine message, as the output at stage i, m_i, can be different from the input at the preceding stage, m'_i, if and only if the node does not hold the message at stage i. Thus, p_t is the probability of $\mathrm{cond}_t(R,S,M)$ to hold true. Note that considering a trace with $m'_i = m_i$ for some $i \ge 1$ would lead to replacing some "$\neg$holds(i)" conjuncts in the above condition with the weaker "$\neg$holds$(i) \vee M = m_i$", which would make for a higher probability.

By simple logical manipulation, the condition $\mathrm{cond}_t(S,R,M)$ is seen to be equivalent to the following:

$$(\mathrm{holds}(0) \wedge M = m_0) \vee \bigwedge_{i=0}^{k} \neg\mathrm{holds}(i) \tag{16}$$

It is easy to evaluate separately the probability of the two disjuncts in the condition. Indeed, holds(0) is equivalent to $S = j$ (the sender must coincide with the node for the node to hold the message at stage 0). For simplicity, assume $\Pr(M = m_0) = \frac{1}{2}$ (this does not really affect the result of the analysis). By independence of S and M we have

$$\Pr(\mathrm{holds}(0) \wedge M = m_0) = \frac{1}{2N}\,.$$

The second disjunct in (16) has the following probability (the computation is detailed in the appendix):

$$\Pr\left(\bigwedge_{i=0}^{k} \neg\mathrm{holds}(i)\right) = 1 - \frac{k+1}{N}\left(1 - \frac{k}{2N}\right).$$

Since the two disjuncts in (16) do not intersect (as holds(0) and $\neg$holds(0) cannot be true at the same time), we can sum up their probabilities and get

$$p_t = \frac{1}{2N} + 1 - \frac{k+1}{N}\left(1 - \frac{k}{2N}\right).$$

As k goes from 0 to $N-1$, p_t decreases from $1 - \frac{1}{2N}$ to its minimal value $\frac{1}{2}$. This shows that $p_t \ge \frac{1}{2}$. The ratio $\frac{\mathcal{B}(p_t)}{|t|}$ as a function of k is plotted in Fig. 3 in the case of $N = 100$ nodes. As seen, the maximum is obtained for $k = 0$, yielding a rate of

$$\mathcal{R}(A_j\,;\; S,R,M) = \mathcal{B}\left(\frac{1}{2N}\right). \tag{17}$$

(we have used the fact that $\mathcal{B}(1-p) = \mathcal{B}(p)$). Hence the traces that maximize the ratio are $t = \overline{out}_{m_0}$ with $m_0 \in 0..1$. Each of these two traces conveys very little information to the attacker, just telling him that it *if* $S = j$ *then* $M = m_0$. As an example, in the case of $N = 100$ nodes, $\mathcal{B}(p_t) \approx 0.045$ bits. As k increases, the conveyed information $\mathcal{B}(p_t)$ grows, but much slower than the length of the trace $|t|$, so that the ratio goes down.

Remark 8.1 In a more realistic scenario, the routing information would be found in message headers rather than being hardwired in the nodes. Since routing information cannot be sent in the clear, some encryption mechanism would be also called for. Conceivably, the resulting system would be more secure than the one discussed above. In particular, the attacker would not be able to tell the stage at which the genuine message is sent (provided randomized encryption were adopted); hence also the identity of the sender and of the message would be fully protected. We cannot directly deal with cryptography and describe such a system in the present framework.

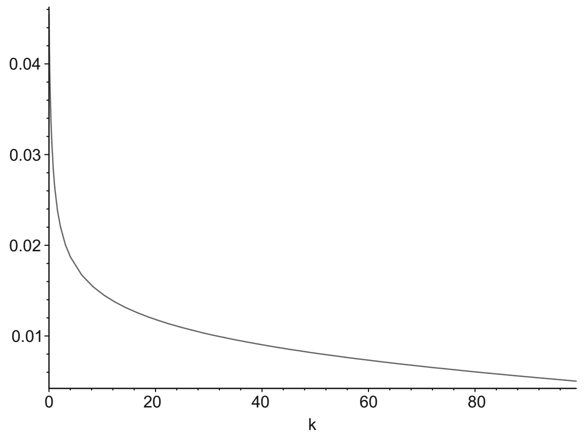


Fig. 3. Plot of $\mathcal{B}(p_t)/|t|$ as a function of k ($N = 100$).

9. Conclusion and related work

We have presented two quantitative models of information leakage for processes. Relationships existing among these two models and a functional notion of secrecy have been studied. The compositionality properties of the models have also been investigated.

The idealized, "all powerful" adversary encompassed by our models may turn out to be too strong for many practical purposes. There is much work to be done in order to go from the present theoretical treatment to a more practical one. In particular, the ability to show an absence of leakage at the language level does not imply that there will be no leakage at the implementation level, although it helps to constrain the types of attacks that can be used effectively, by forcing an attacker to require some additional knowledge relating to, e.g., timing. In a probabilistic setting where each process induces a probability distribution over the set of traces, it may be sensible to stipulate that low-probability traces are more difficult, or costly, to detect for the adversary than high-probability traces. This is impossible to describe in the present model, basically because, no matter how improbable a specific trace is, the attacker can detect that trace with a null effort. In the future, we plan to examine enhancements of the model involving probabilistic and possibly cryptographic features.

Use of conditional mutual information as a measure of leakage in computing systems can be traced back to Millen [26] and to Gray [18]. In the context of sequential, imperative programs, the significance of this measure with respect to different metrics of security has been further clarified by Clark, Hunt and Malacaria in [10,12,11]. In particular, our Theorem 5.1, stating equivalence of zero leakage and secrecy, is clearly related to Millen's result [26] that null conditional mutual information is equivalent to non-interference in the case of computing automata. In a language-based setting, essentially the same result has been proved by Clark et al. (Proposition 4.2 of [10]). Malacaria's recent work on the security of looping constructs [24] extends [10] by introducing a notion of rate for loops in imperative programs. We also mention Volpano and Smith's [34], where a quantified theory of non-interference for imperative programs is developed, including a notion of rate of leakage, albeit not based on Information Theory. Di Pierro et al. [15] propose a notion of indistinguishability for probabilistic constraint programs and study its relationship with certain security measures, such as the average number of runs necessary for an attacker to tell two systems apart.

It is worth to notice that the above mentioned works presuppose terminating computations that produce a set of "results" with a probability distribution. As such they are not appropriate to a process algebraic setting, where one just cares about the interactive behaviours of systems (and computations may be non-terminating).

Mutual information is also at the heart of the notion of *channel capacity*, which is defined as the maximum mutual information between the source and the output of a (noisy) channel. Indeed, it is perfectly sensible to view a computing system (program or process) as a channel, where the source is represented by the sensitive information one wishes to conceal and the output is whatever "observable" is appropriate for the system under consideration (state variables or behaviours). This analogy is pursued in recent works on anonymity protocols by Chatzikokolakis, Palamidessi and collaborators [8,7]. As expected, a basic result in this setting is that perfect anonymity corresponds to zero capacity. There is, however, an important difference between their anonymity-based approach and those based on secrecy/non-interference (including ours). In essence, the protocol models of [8,7] rely on *noise* to conceal sensitive information in the system, e.g. sender's identity: the noisier the channel, the lesser the capacity, the more secure is deemed the system. Noise generation is modeled by resorting to probabilistic choice. On the other hand, languages considered in secrecy/non-interference frameworks do not necessarily feature probabilistic operators (ours does not, neither do the languages of e.g. [10,24]). Indeed, in these languages, it is intended that programmers conceal a sensitive piece of information X essentially via *distortion*. In coding theory, distortion is the loss of information that occurs when a source X is coded unfaithfully, i.e. using a number of bits smaller than required. In our model of absolute leakage, X is "coded up" and made available to the attacker as the r.v. $Z = P(X)$ and proper distortion happens if $H(Z) < H(X)$. For uniformly distributed X, distortion is realized if the function $u \mapsto [P(u)]_{\asymp}$ is non-injective, with the ideal case being that this function is a constant. It would be interesting to combine the two approaches (noise-based and distortion-based) into a single framework.

In the realm of process algebras, a paper by Lowe [23] has introduced a notion of quantitative non-interference for timed CSP, defined as the number of different observable "low" behaviours that a "high" user can induce on the process. This definition is shown to be in agreement with a functional notion of lack of information flow due to Focardi and Gorrieri [16], which can be regarded as a process-algebraic version of functional non-interference (a probabilistic extension of this equivalence is in Aldini et al.'s [2]). This result is somehow related to Wittbold's [35], where various notions of lack of information flow for nondeterministic systems, like non-deducibility and forward correctability, are assessed on the basis of information-theoretic arguments. In [23], a notion of rate is also introduced that corresponds to the ratio of leaked information/elapsed time. Lowe's model is not easily comparable to ours, due to the different goals and settings (secrecy vs. process-algebraic non-interference, untimed vs. timed). For example, as noted by Lowe, a notion of rate directly based on elapsed time is to some extent unsatisfactory: a process that leaks one Gigabyte during the first second of its execution and then remains silent forever has a leakage rate of 0, and as such should be deemed as secure. A similar drawback, in a sequential/imperative setting, arises in the mentioned [24], where the leakage rate of a looping construct is obtained as the ratio of absolute leakage and number of iterations of the loop.

Finally, it is worth to mention some recent work in Information Security that addresses the issue of side-channels attacks against cryptographic hardware from an information-theoretic perspective very similar in spirit to that presented here: see e.g. [22,31,32]. As an example, the analysis of the modular exponentiation algorithm found in [22] bears some similarities to our absolute leakage model (Example 5.5). We leave for future work the task of establishing a precise connection between our models and these approaches.

Acknowledgments

Comments provided by the ICALP 2006 and I&C reviewers have been very helpful to improve on the presentation of the work.

Appendix A. Summary of notation

Information theory

$\Pr(A)$	probability	$\Pr(A \mid B)$	conditional probability
X	random variable	$\tilde{X}$	vector of random variables
$H(X)$	entropy	$H(X \mid Y)$	conditional entropy
$I(X;Y)$	mutual information	$I(X;Y \mid Z)$	conditional mutual information
$\tilde{p}$	probability distribution	$H(\tilde{p})$	entropy of distribution
$\mathcal{B}(p)$	binary entropy function		

Process theory

a, b	channel names	x, y	variables
u, v	values	σ	substitution
ϕ	formula	$\sigma \models \phi$	satisfaction
P	process	$P(\tilde{x})$	open process
s	trace	$P \overset{s}{\Longrightarrow}$	process performs trace
$\asymp$	behavioural equivalence	$[P]_{\asymp}$	behavioural equivalence class
$\simeq$	trace equivalence	$\sim$	strong bisimilarity
$C[\cdot]$	context	$C[P]$	process replaces hole
T	test	$\lvert T \rvert$	cost of test

Information leakage in processes

$P(\tilde{X})$	open process as random variable
$P(\tilde{X})\vert\vert T$	test on process as random variable
$P(\tilde{X}) \overset{s}{\Longrightarrow}$	process trace as random variable
$\mathcal{A}\left(P\,;\,\tilde{X} \mid \tilde{Y}\right)$	absolute leakage from $\tilde{X}$ to P given $\tilde{Y}$
$\mathcal{R}\left(P\,;\,\tilde{X}\right)$	leakage rate of P relative to $\tilde{X}$

Appendix B. Definition of leakage rate for name-passing processes

We give a definition of rate in the pi-calculus and then show that it enjoys a trace-based characterization analogous to that provided by Proposition 7.1 in the case without name-passing. Reformulating and extending the rest of the results of Section 7 to the pi-calculus is then a matter of routine, and is left to the interested reader. Note that the definition of rate seen in Section 7.4 does not apply "as is" to the pi-calculus, as each input prefix $a(x).P$ with $x : \mathcal{S}$ gives rise to infinitely many traces, corresponding to the infinitely many instantiations of the input parameter x with names in the sort $\mathcal{S}$. However, once a test T and a process P have been fixed, the "relevant" traces of T are only those that have a chance of giving rise to a synchronization with P. In particular, the set of possible instantiations of x can be restricted to a (super)set of the free names of P, say N. This is the intuition behind the notion of N-trace given below.

For ease of presentation, in our treatment below we stick to a monadic pi-calculus, that is, we consider only processes with action prefixes carrying one object, that take the form $a(x).P$ or $\overline{a}e.P$. We shall abbreviate $(\nu b)\overline{a}b.P$ as $\overline{a}(b).P$ when $a \neq b$. For the sake of symmetry, we shall also admit input prefixes of the form $a(b).P$, where the formal parameter is a name $b : \mathcal{S}$ for some $\mathcal{S}$.

In what follows, we let s, s' range over traces of the form $\mu_1 \cdots \mu_n$ with $n \geq 0$ and $\mu_i ::= ad \;\big|\; a(b) \;\big|\; \overline{a}d \;\big|\; \overline{a}(b) \;\big|\; \omega$ and $d ::= a|v$. These traces are taken up to alpha-equivalence, once $a(b)$ and $\overline{a}(b)$ are considered as binders for name b. Moreover, it is assumed that actions in traces respect the given sorting system. The set of names occurring free in input subject position in s will be denoted by $\mathrm{ifn}(s)$. Finally, we write $A \xrightarrow{a(b)} A'$ if $A \xrightarrow{ab} A'$ and $b \notin \mathrm{fn}(A)$: thus $A \overset{s}{\Longrightarrow}$ is well-defined for any trace s in the syntax described above.

Definition B.1 (*N-Traces*)**.** Let N be a finite set of names and A be a process or a test. We say *s is a N-trace of A* if $A \overset{s}{\Longrightarrow}$ and $\mathrm{ifn}(s) \subseteq N$. We say (N, s) *may lead A to success* if s is nonempty, ω does not occur in s and $s \cdot \omega$ is a N-trace of A.

In what follows, $P(\tilde{x})$ is an open process and $\tilde{X}$ a r.v. A test T is a finite process possibly using the distinct action ω. Clearly, a test has only a finite number of N-traces.

Definition B.2 (*Rate of leakage*)**.** For each test T and finite set of names N, the *N-cost* of T is

$$|T|_N \overset{\text{def}}{=} \sum_{s:\,(N,s)\text{ may lead } T \text{ to success}} |s|\,.$$

The *leakage rate of P relative to $\tilde{X}$* is

$$\mathcal{R}(P\,;\,\tilde{X}) \overset{\text{def}}{=} \sup_{N,T\,:\,\mathrm{fn}(P)\subseteq N,\,|T|_N>0} \frac{H\Big(P(\tilde{X})\|T\Big)}{|T|_N}\,.$$

To prove the analog of Proposition 7.1 we need some additional definitions and terminology. Given a trace s not containing ω, the test $\hat{s}_N$ that checks for the presence of a N-trace s in a process with free names $\subseteq N$ is defined by induction on s as follows:

- $\hat{\varepsilon}_N = \omega$
- $\widehat{ad \cdot s'}_N = \overline{a}d.\hat{s'}_N$
- $\widehat{a(b) \cdot s'}_N = \overline{a}(b).\hat{s'}_{N\cup\{b\}}$
- $\widehat{\overline{a}d \cdot s'}_N = a(x).[x = d]\hat{s'}_N$
- $\widehat{\overline{a}(b) \cdot s'}_N = a(b).[b \notin N]\hat{s'}_{N\cup\{b\}}$.

For instance, if $s = a(b) \cdot \overline{c}(d) \cdot \overline{d}b$ and $N = \{a, c\}$ then $\hat{s}_N = \overline{a}(b).c(d).[d \notin \{a, c, b\}].d(y).[y = b]\omega$. For a trace s, its complement $\overline{s}$ is defined by inverting polarities of its actions (i.e. by turning inputs into outputs, and vice-versa, and leaving the object part unchanged). E.g., for s as defined above we have $\overline{s} = \overline{a}(b) \cdot c(d) \cdot db$. Clearly $\overline{\overline{s}} = s$.

The following lemma summarizes what we need to know about $\hat{s}_N$. The proof is routine and omitted.

Lemma B.1. *Let N be a finite set of names.*

(a) Let Q be a closed process and s a trace with $\mathrm{ifn}(s) \cup \mathrm{fn}(Q) \subseteq N$*. Then s is a N-trace of Q if and only if $Q\|\hat{s}_N \overset{\omega}{\Longrightarrow}$.*
(b) Let T be a test and let S be the set of traces s that s.t. (N, s) may lead T to success. Then

$$T \simeq \sum_{s\in S} \hat{\overline{s}}_N + F$$

for some F that has no trace s s.t. (N, s) may lead F to success.

(c) For each trace s, we have $|\hat{s}_N|_N = |s| = |\bar{s}|$.

Proposition B.1. *It holds that*

$$\mathcal{R}(P\,;\,\tilde{X}) \;=\; \sup_{|s|>0} \frac{H\Big(P(\tilde{X}) \overset{s}{\Longrightarrow}\Big)}{|s|} \tag{18}$$

Proof. By the previous lemma, part (a), for any trace s, $P(\tilde{X}) \overset{s}{\Longrightarrow}$ is equivalent to $P(\tilde{X})\|\hat{s}_N$, for some $N \supseteq \mathrm{fn}(P,s)$. By this fact and by the previous lemma part (c) the RHS of (18) is not greater than the LHS. For the opposite inequality, fix any finite N with $N \supseteq \mathrm{fn}(P)$, fix any T with $|T|_N > 0$ and let S be the set traces s s.t. (N,s) may lead T to success. By the previous lemma, part (b), and by simple $\simeq$-preserving transformations, we can show that, for any Q with $\mathrm{fn}(Q) \subseteq N$:

$$Q\|T \simeq Q\|\left(\sum_{s\in S}\hat{\hat{s}}_N + F\right) \simeq \sum_{s\in S}(Q\|\hat{\hat{s}}_N)$$

(note in particular that $Q\|F \simeq \mathbf{0}$, as no (N,s) may lead F to success). Hence $T_{P,\tilde{X}} = P(\tilde{X})\|T$ is equivalent to $\sum_{s\in S}(P(\tilde{X})\|\hat{\hat{s}}_N)$ as a r.v. Using this fact and the inequality provided by Proposition 5.1 with $C[\cdot] = \sum_{s\in S}([\cdot])$ (or relying on the data processing inequality), we have that

$$H(T_{P,\tilde{X}})/|T|_N = H\left(\sum_{s\in S}P(\tilde{X})\|\hat{\hat{s}}_N\right)/|T|_N \le \left(\sum_{s\in S}H(P(\tilde{X})\|\hat{\hat{s}}_N)\right)/|T|_N\,.$$

In the last term, we can replace $P(\tilde{X})\|\hat{\hat{s}}_N$ by $P(\tilde{X}) \overset{\bar{s}}{\Longrightarrow}$ (by the previous lemma, part (a)), hence $H(P(\tilde{X})\|\hat{\hat{s}}_N)$ by $\mathcal{B}(p_{\bar{s}})$. Moreover, by the previous lemma, parts (b) and (c), we can replace $|T|_N$ by $\sum_{s\in S}|s| = \sum_{s\in S}|\bar{s}|$. The thesis then follows by applying inequality (14). $\square$

Appendix C. Details of the example in Section 8

C.1. Analysis of absolute leakage

It is handier to first analyze a case where the attacker knows whether the node j is or is not in the path from the sender to the receiver. That is, we consider the side-information given by the random variable

$$Y \overset{\text{def}}{=} \mathrm{path}(S,j,R)$$

and we want to first compute

$$\mathcal{A}(A_j\,;\,S,R,M \mid Y) = H(Z \mid Y)\,.$$

The random variable $D \overset{\text{def}}{=} (j - S) \bmod N$ measures the distance between the sender S and j. If j is in the path from S to R ($Y = \mathit{true}$), then the behaviour of A_j is such that at the Dth stage of the protocol, A_j can only fire a unique output $\overline{out}_m$ with $m = M$, while at any other stage it can nondeterministically choose to between $\overline{out}_0$ and $\overline{out}_1$ (intuitively, the attacker can tell these two situations apart by repeatedly executing A_j and recording at which stage it always observes the same output). In other words, A_j's behaviour in this case depends solely on D and on M, in the sense that different values for the pair (D,M) correspond to different behaviours of A_j. Noting that D can take on the values $0,1,\ldots,N-1$ with uniform probability, these considerations yield

$$H(Z \mid Y = \mathit{true}) = H(D,M) = H(D) + H(M) = \log N + h.$$

On the other hand, if j is *not* in the path from S to R ($Y = \mathit{false}$), the `then` branches are never taken and the behaviour of A_j is independent from S, R and M. Hence

$$H(Z \mid Y = \mathit{false}) = 0.$$

Now, an easy counting argument shows that the probability of $Y = \mathit{true}$ is $\approx \frac{1}{2}$, more precisely $\Pr(Y = \mathit{true}) = \frac{1}{2}(1 + \frac{1}{N})$, so that

$$H(Z \mid Y) = \Pr(Y = \mathit{true})H(Z \mid Y = \mathit{true}) = \frac{1}{2}\left(1 + \frac{1}{N}\right)(\log N + h)\,.$$

Finally, we can compute the absolute leakage $H(Z)$ using first the chain rule to derive the formula

$$H(Z) = H(Y) + H(Z \mid Y) - H(Y \mid Z)$$

and then noting that $H(Y \mid Z) = 0$: indeed, Y is determined by Z, as j is in the path from S to R if and only if at some stage of Z a unique output, rather than two possible, can be observed. To sum up

$$\mathcal{A}(A_j\,;\,S,R,M) = \mathcal{B}\left(\frac{1}{2}\left(1+\frac{1}{N}\right)\right) + \frac{1}{2}\left(1+\frac{1}{N}\right)(\log N + h) \;\approx\; 1+\frac{1}{2}(\log N + h)\,.$$

C.2. Analysis of rate of leakage

Let t be any trace with non-zero probability of $Z = A_j(S,R,M)$, that is, assume $p_t \overset{\text{def}}{=} \Pr(Z \overset{t}{\Longrightarrow}) > 0$.

Firing output actions eagerly maximizes entropy The following considerations can be justified by inspection of A_j's code. Let O be the number of output actions in t, then: (a) t must contain at least $O-1$ input actions; (b) if t contains more than O input actions, then there is a shorter trace t' obtained by erasing some input action such that $Z \overset{t}{\Longrightarrow}$ if and only if $Z \overset{t'}{\Longrightarrow}$; (c) if t contains either O or $O-1$ input and O output actions, there is a permutation t' of t where input and output actions alternate with one another such that $Z \overset{t}{\Longrightarrow}$ if and only if $Z \overset{t'}{\Longrightarrow}$. Using repeatedly (a,b,c) above, it follows that, when looking for a trace t maximizing the ratio $\mathcal{B}(p_t)/|t|$, we can restrict ourselves to traces of one of two forms

$$\begin{array}{llll}(1) & t & = & \overline{out}_{m_0} \cdot in_{m'_1} \cdot \overline{out}_{m_1} \cdots in_{m'_k} \cdot \overline{out}_{m_k}\\ (2) & t & = & \phantom{\overline{out}_{m_0} \cdot} in_{m'_1} \cdot \overline{out}_{m_1} \cdots in_{m'_k} \cdot \overline{out}_{m_k}\end{array}$$

with $0 \le k \le N-1$ and $m_i, m'_i \in 0..1$. The analysis of traces of the form (2) is similar to the one for traces of the form (1), seen in Section 8. In particular, the maximum ratio $\mathcal{B}(p_t)/|t|$ can be computed similarly and is lower than that achievable using traces of the form (1). The details are left to the interested reader.

Evaluating the probability of $\bigwedge_{i=0}^{k} \neg\text{holds}(i)$. Using De Morgan's law, the condition $\bigwedge_{i=0}^{k} \neg\text{holds}(i)$ can be written as $\neg(\bigvee_{i=0}^{k} \text{holds}(i))$. Since the holds$(i)$'s do not intersect with each other (holds$(i) \wedge$ holds(i') is false for $i \neq i'$), the probability of this event can be written as

$$1 - \sum_{i=0}^{k} \Pr(\text{holds}(i))\,.$$

A simple counting arguments shows that $\Pr(\text{holds}(i)) = \frac{1}{N}(1-\frac{i}{N})$ (indeed, the value of S is fixed and there are $N-i$ possible values for R, out of N^2 possible values for the pair (S,R)). Using the formula for the sum of the integers from 1 to k yields

$$\sum_{i=0}^{k} \Pr(\text{holds}(i)) = \frac{k+1}{N}\left(1-\frac{k}{2N}\right)$$

hence the result.

References

[1] M. Abadi, A. Gordon, A calculus for cryptographic protocols: the Spi-calculus, Information and Computation, 148 (1) (1999) 1–70.
[2] A. Aldini, M. Bravetti, R. Gorrieri, A process-algebraic approach for the analysis of probabilistic non-interference, Journal of Computer Security 12 (2) (2004) 191–245., IOS Press, March.
[3] E. Bach, et al., What's a key-guessing attack? What's entropy? in: Cryptography Frequently Asked Questions, Section 4.9. Avaliable from: <http://www.faqs.org/faqs/cryptography-faq/part04/>.
[4] M. Boreale, Quantifying information leakage in process calculi, in: ICALP 2006, LNCS, vol. 4052, Springer, 2006.
[5] M. Boreale, R. De Nicola, Testing equivalence for mobile processes, Information and Computation 120 (2) (1995) 279–303.
[6] M. Boreale, R. De Nicola, A symbolic semantics for the pi-calculus, Information and Computation 126 (1) (1996) 34–52.
[7] C. Braun, K. Chatzikokolakis, C. Palamidessi, Compositional methods for information-hiding, in: Proceedings of FOSSACS'08, LNCS, vol. 4962, Springer, 2008.
[8] K. Chatzikokolakis, C. Palamidessi, P. Panangaden, Anonymity protocols as noisy channels, in: Proceedings of the 2nd Symposium on Trustworthy Global Computing (TGC 06), Springer, LNCS, 2006, Full version in Information and Computation, 206 (2008), 378–401.
[9] K. Chatzikokolakis, C. Palamidessi, P. Panangaden, Probability of error in information-hiding protocols, in: Proceedings of the 20th IEEE CSF, IEEE Computer Society, 2007.
[10] D. Clark, S. Hunt, P. Malacaria, Quantitative analysis of the leakage of confidential data, Electronic Notes in Theoretical Computer Science 59 (3) (2001).
[11] D. Clark, S. Hunt, P. Malacaria, Quantitative information flow, relations and polymorphic types, Journal of Logic and Computation 15 (2) (2005) 181–199.

[12] D. Clark, S. Hunt, P. Malacaria, A static analysis for quantifying information flow in a simple imperative language, Journal of Computer Security 15 (3) (2007) 321–371.
[13] T.M. Cover, J.A. Thomas, Elements of Information Theory, Wiley, New York, 1991.
[14] R. De Nicola, M.C.B. Hennessy, Testing equivalences for processes, Theoretical Computer Science 34 (1984) 83–133.
[15] A. Di Pierro, C. Hankin, H. Wiklicky, Approximate non-interference, in: Computer Security Foundations Workshop, 2002, Full version in Journal of Computer Security 12 (1) (2004) 37–82.
[16] R. Focardi, R. Gorrieri, A classification of security properties, Journal of Computer Security 3 (1) (1995) 5–34.
[17] J.A. Goguen, J. Meseguer, Security policies and security models, IEEE Symposium on Security and Privacy (1982).
[18] J.W. Gray III, Towards a mathematical foundation for information flow security, in: Proceedings of 1991 IEEE Symposium on Research in Computer Security and Privacy, 1991.
[19] M.C.B. Hennessy, H. Lin, Symbolic bisimulations, Theoretical Computer Science 138 (2) (1995) 353–389.
[20] P. Kocher, Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems, in: CRYPTO 1996, 1996, pp. 104–113.
[21] P. Kocher, J. Jaffe, B. Jun, Differential power analysis, in: CRYPTO 1999, 1999, pp. 388–397.
[22] B. Köpf, D.A. Basin, An information-theoretic model for adaptive side-channel attacks, ACM Conference on Computer and Communications Security (2007) 286–296.
[23] G. Lowe, Defining information flow quantity, Journal of Computer Security 12 (3–4) (2004) 619–653.
[24] P. Malacaria, Assessing security threats of looping constructs, in: POPL 2007.
[25] J.L. Massey. Guessing and entropy, in: Proceedings of IEEE International Symposium on Information Theory, 1994, p. 204.
[26] J. Millen, Covert channel capacity, in: Proceedings of 1987 IEEE Symposium on Research in Computer Security and Privacy, 1987.
[27] J.O. Pliam, On the incomparability of entropy and marginal guesswork in Brute-force attacks, in: Proceedings of Progress in Cryptology – INDOCRYPT 2000, First International Conference in Cryptology in India, Calcutta, India, December 2000, LNCS, vol. 1977, Springer-Verlag.
[28] M. Reiter, A. Rubin, Crowds: anonymity for web transactions, ACM Transactions on Information and System Security 1 (1) (1998).
[29] D. Sangiorgi, D. Walker, The Pi-Calculus: A Theory of Mobile Processes, Cambridge University Press, 2001.
[30] C.E. Shannon, Communication theory of secrecy systems, Bell System Technical Journal 27 (1948) 379–423., 623–656.
[31] F.-X. Standaert, E. Peeters, C. Archambeau, J.-J. Quisquater, Towards security limits in side-channel attacks, in: Proceedings of CHES 2006, Lecture Notes in Computer Science, vol. 4249, Yokohama, Japan, October 2006, Springer-Verlag, pp. 30–45.
[32] F.-X. Standaert, T.G. Malkin, M. Yung, A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks, Cryptology ePrint Archive, Report 2006/139, February 2008.
[33] F. Topsøe, Basic concepts, identities and inequalities – the Toolkit of information theory, Entropy 3 (2001) 162–190. Available from: <http://www.math.ku.dk/∼topsoe/toolkitfinal.pdf/>.
[34] D. Volpano, G. Smith, Verifying secrets and relative secrecy, in: POPL 2000, 2000, pp. 268–276.
[35] J.T. Wittbold, D. Johnson, Information flow in nondeterministic systems, in: Proceedings of 1990 IEEE Symposium on Research in Computer Security and Privacy, 1990.