Computing the Leakage of Information-Hiding Systems

Miguel E. Andrés[1], Catuscia Palamidessi[2],
Peter van Rossum[1], Geoffrey Smith[3].

[1]Institute for Computing and Information Sciences, The Netherlands.
[2]INRIA and LIX, École Polytechnique Palaiseau, France.
[3]SCIS, Florida International University, USA.

Abstract. We address the problem of computing the information leakage of a system in an efficient way. We propose two methods: one based on reducing the problem to reachability, and the other based on techniques from quantitative counterexample generation. The second approach can be used either for exact or approximate computation, and provides feedback for debugging. These methods can be applied also in the case in which the input distribution is unknown. We then consider the interactive case and we point out that the definition of associated channel proposed in literature is not sound. We show however that the leakage can still be defined consistently, and that our methods extend smoothly.

1 Introduction

By *information hiding*, we refer generally to the problem of constructing protocols or programs that protect sensitive information from being deduced by some adversary. In *anonymity protocols* [4], for example, the concern is to design mechanisms to prevent an observer of network traffic from deducing who is communicating. In *secure information flow* [17], the concern is to prevent programs from leaking their secret input to an observer of their public output. Such leakage could be accidental or malicious.

Recently, there has been particular interest in approaching these issues *quantitatively*, using concepts of information theory. See for example [13, 5, 10, 6, 4]. The secret input S and the observable output O of an information-hiding system are modeled as random variables related by a *channel matrix*, whose (s, o) entry specifies $P(o|s)$, the conditional probability of observing output o given input s. If we define the *vulnerability* of S as the probability that the adversary could correctly guess the value of S in one try, then it is natural to measure the information leakage by comparing the *a priori* vulnerability of S with the *a posteriori* vulnerability of S after observing O. We consider two measures of leakage: *additive*, which is the difference between the *a posteriori* and *a priori* vulnerabilities; and *multiplicative*, which is their quotient [19, 3].

We thus view a protocol or program as a *noisy channel*, and we calculate the leakage from the channel matrix and the *a priori* distribution on S. But, given an operational specification of a protocol or program, how do we calculate the parameters of the noisy channel: the sets of inputs and outputs, the *a priori* distribution, the channel matrix, and the associated leakage? These are the main questions we address in this paper. We focus on *probabilistic automata*, whose transitions are labeled with probabilities and *actions*, each of which is classified as secret, observable, or internal.

We first consider the simple case in which the secret inputs take place at the beginning of runs, and their probability is fixed. The interpretation in terms of noisy channel of this kind of systems is well understood in literature. The framework of probabilistic automata, however, allows to represent more general situations. Thanks to the nondeterministic choice, indeed, we can model the case in which the input distribution is unknown, or variable. We show that the definition of channel matrix extends smoothly also to this case. Finally, we turn our attention to the interactive scenario in which inputs can occur again after outputs. This case has also been considered in literature, and there has been an attempt to define the channel matrix in terms of the probabilities of traces [11]. However it turns out that the notion of channel is unsound. Fortunately the leakage is still well defined, and it can be obtained in the same way as the simple case.

We consider two different approaches to computing the channel matrix. One uses a system of linear equations as in reachability computations. With this system of equations one can compute the *joint matrix*, the matrix of probabilities of observing both s and o; the channel matrix is trivially derived from this joint matrix. The other approach starts with a 0 channel matrix, which we call a *partial matrix* at this point. We iteratively add the contributions in conditional probabilities of complete paths to this partial matrix, obtaining, in the limit, the channel matrix itself. We then group paths with the same secret and the same observable together using ideas from quantitative counterexample generation, namely by using regular expressions and strongly connected component analysis. In this way, we can add the contribution of (infinitely) many paths at the same time to the partial matrices. This second approach also makes it possible to identify which parts of a protocol contribute most to the leakage, which is useful for debugging.

Looking ahead, after reviewing some preliminaries (Section 2) we present restrictions on probabilistic automata to ensure that they have well-defined, finite channel matrices (Section 3). This is followed by the techniques to calculate the channel matrix efficiently (Section 4 and Section 5). We then turn our attention to extensions of our information-hiding system model. We use nondeterministic choice to model the situation where the *a priori* distribution on the secret is unknown (Section 6). Finally, we consider interactive systems, in which secret actions and observable actions can be interleaved arbitrarily (Section 7).

2 Preliminaries

2.1 Probabilistic automata

This section recalls some basic notions on probabilistic automata. More details can be found in [18]. A function $\mu\colon Q \to [0,1]$ is a *discrete probability distribution* on a set Q if the support of μ is countable and $\sum_{q \in Q} \mu(q) = 1$. The set of all discrete probability distributions on Q is denoted by $\mathcal{D}(Q)$.

A *probabilistic automaton* is a quadruple $M = (Q, \Sigma, \hat{q}, \alpha)$ where Q is a countable set of *states*, Σ a finite set of *actions*, $\hat{q}$ the *initial* state, and α a *transition function* $\alpha : Q \to \wp_f(\mathcal{D}(\Sigma \times Q))$. Here $\wp_f(X)$ is the set of all finite subsets of X. If $\alpha(q) = \emptyset$ then q is a *terminal* state. We write $q \to \mu$ for $\mu \in \alpha(q),\ q \in Q$. Moreover, we write $q \xrightarrow{a} r$ for $q, r \in Q$ whenever $q \to \mu$ and $\mu(a, r) > 0$. A *fully probabilistic automaton* is a probabilistic automaton satisfying $|\alpha(q)| \leq 1$ for all states. In case $\alpha(q) \neq \emptyset$ we will overload notation and use $\alpha(q)$ to denote the distribution outgoing from q.

A *path* in a probabilistic automaton is a sequence $\sigma = q_0 \stackrel{a_1}{\rightarrow} q_1 \stackrel{a_2}{\rightarrow} \cdots$ where $q_i \in Q$, $a_i \in \Sigma$ and $q_i \stackrel{a_{i+1}}{\rightarrow} q_{i+1}$. A path can be *finite* in which case it ends with a state. A path is *complete* if it is either infinite or finite ending in a terminal state. Given a path σ, $\mathrm{first}(\sigma)$ denotes its first state, and if σ is finite then $\mathrm{last}(\sigma)$ denotes its last state. A *cycle* is a path σ such that $\mathrm{last}(\sigma) = \mathrm{first}(\sigma)$. We denote the set of actions occurring in a cycle as $\mathrm{CyclesA}(M)$. Let $\mathrm{Paths}_q(M)$ denote the set of all paths, $\mathrm{Paths}^{\star}{}_q(M)$ the set of all finite paths, and $\mathrm{CPaths}_q(M)$ the set of all complete paths of an automaton M, starting from the state q. We will omit q if $q = \hat{q}$. Paths are ordered by the prefix relation, which we denote by $\leq$. The *trace* of a path is the sequence of actions in $\Sigma^* \cup \Sigma^\infty$ obtained by removing the states, hence for the above σ we have $trace(\sigma) = a_1 a_2 \ldots$. If $\Sigma' \subseteq \Sigma$, then $trace_{\Sigma'}(\sigma)$ is the projection of $trace(\sigma)$ on the elements of Σ'. The *length* of a finite path σ, denoted by $|\sigma|$, is the number of actions in its trace.

Let $M = (Q, \Sigma, \hat{q}, \alpha)$ be a (fully) probabilistic automaton, $q \in Q$ a state, and let $\sigma \in \mathrm{Paths}^{\star}_q(M)$ be a finite path starting in q. The *cone* generated by σ is the set of complete paths $\langle\sigma\rangle = \{\sigma' \in \mathrm{CPaths}_q(M) \mid \sigma \leq \sigma'\}$. Given a fully probabilistic automaton $M = (Q, \Sigma, \hat{q}, \alpha)$ and a state q, we can calculate the *probability value*, denoted by $\mathbf{P}_q(\sigma)$, of any finite path σ starting in q as follows: $\mathbf{P}_q(q) = 1$ and $\mathbf{P}_q(\sigma \stackrel{a}{\rightarrow} q') = \mathbf{P}_q(\sigma)\ \mu(a, q'),$ where $\mathrm{last}(\sigma) \rightarrow \mu$.

Let $\Omega_q \triangleq \mathrm{CPaths}_q(M)$ be the sample space, and let $\mathcal{F}_q$ be the smallest σ-algebra generated by the cones. Then $\mathbf{P}$ induces a unique *probability measure* on $\mathcal{F}_q$ (which we will also denote by $\mathbf{P}_q$) such that $\mathbf{P}_q(\langle\sigma\rangle) = \mathbf{P}_q(\sigma)$ for every finite path σ starting in q. For $q = \hat{q}$ we write $\mathbf{P}$ instead of $\mathbf{P}_{\hat{q}}$.

Given a probability space $(\Omega, \mathcal{F}, P)$ and two events $A, B \in F$ with $P(B) > 0$, the *conditional probability* of A given B, $P(A \mid B)$, is defined as $P(A \cap B)/P(B)$.

2.2 Noisy Channels

This section briefly recalls the notion of noisy channels from Information Theory [7].

A *noisy channel* is a tuple $\mathcal{C} \triangleq (\mathcal{X}, \mathcal{Y}, P(\cdot|\cdot))$ where $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ is a finite set of *input values*, modeling the *secrets* of the channel, and $\mathcal{Y} = \{y_1, y_2, \ldots, y_m\}$ is a finite set of *output values*, the *observables* of the channel. For $x_i \in \mathcal{X}$ and $y_j \in \mathcal{Y}$, $P(y_j | x_i)$ is the conditional probability of obtaining the output y_j given that the input is x_i. These conditional probabilities constitute the so called *channel matrix*, where $P(y_j|x_i)$ is the element at the intersection of the i-th row and the j-th column. For any input distribution P_X on $\mathcal{X}$, P_X and the channel matrix determine a joint probability $P_\wedge$ on $\mathcal{X} \times \mathcal{Y}$, and the corresponding marginal probability P_Y on $\mathcal{Y}$ (and hence a random variable Y). P_X is also called *a priori distribution* and it is often denoted by π. The probability of the input given the output is called *a posteriori distribution*.

2.3 Information leakage

We recall here the definitions of *multiplicative leakage* proposed in [19], and *additive leakage* proposed in [3][1]. We assume given a noisy channel $\mathcal{C} = (\mathcal{X}, \mathcal{Y}, P(\cdot|\cdot))$ and a random variable X on $\mathcal{X}$. The *a priori vulnerability* of the secrets in $\mathcal{X}$ is the probability

[1] The notion proposed by Smith in [19] was given in a (equivalent) logarithmic form, and called simply *leakage*. For uniformity's sake we use here the terminology and formulation of [3].

of guessing the right secret, defined as $V(X) \triangleq \max_{x\in\mathcal{X}} P_X(x)$. The rationale behind this definition is that the adversary's best bet is on the secret with highest probability.

The *a posteriori vulnerability* of the secrets in $\mathcal{X}$ is the probability of guessing the right secret, after the output has been observed, averaged over the probabilities of the observables. The formal definition is $V(X \mid Y) \triangleq \sum_{y\in\mathcal{Y}} P_Y(y) \max_{x\in\mathcal{X}} P(x \mid y)$. Again, this definition is based on the principle that the adversary will choose the secret with the highest a posteriori probability.

Note that, using Bayes theorem, we can write the a posteriori vulnerability in terms of the channel matrix and the a priori distribution, or in terms of the joint probability:

$$V(X \mid Y) \;=\; \sum_{y\in\mathcal{Y}} \max_{x\in\mathcal{X}} (P(y \mid x) P_X(x)) \;=\; \sum_{y\in\mathcal{Y}} \max_{x\in\mathcal{X}} P_\wedge(x, y). \tag{1}$$

The *multiplicative* leakage is defined as $\mathcal{L}_\times(\mathcal{C}, P_X) \triangleq \frac{V(X|Y)}{V(X)}$ and the *additive* leakage as $\mathcal{L}_+(\mathcal{C}, P_X) \triangleq V(X|Y) - V(X)$.

3 Information Hiding Systems

To formally analyze the information-hiding properties of protocols and programs, we propose to model them as a particular kind of probabilistic automata, which we call *Information-Hiding Systems* (IHS). Intuitively, an IHS is a probabilistic automaton in which the actions are divided in three (disjoint) categories: those which are supposed to remain secret (to an external observer), those which are visible, and those which are internal to the protocol.

First we consider only the case in which the choice of the secret takes place entirely at the beginning, and is based on a known distribution. Furthermore we focus on fully probabilistic automata. Later in the paper we will relax these constraints.

Definition 3.1 (Information-Hiding System). An information-hiding system (IHS) is a quadruple $\mathcal{I} = (M, \Sigma_\mathcal{S}, \Sigma_\mathcal{O}, \Sigma_\tau)$ where $M = (Q, \Sigma, \hat{q}, \alpha)$ is a fully probabilistic automaton, $\Sigma = \Sigma_\mathcal{S} \cup \Sigma_\mathcal{O} \cup \Sigma_\tau$ where $\Sigma_\mathcal{S}$, $\Sigma_\mathcal{O}$, and Σ_τ are pairwise disjoint sets of secret, observable, and internal actions, and α satisfies the following restrictions:

1. $\alpha(\hat{q}) \in \mathcal{D}(\Sigma_\mathcal{S} \times Q)$,
2. $\forall s \in \Sigma_\mathcal{S}\ \exists! q\ .\ \alpha(\hat{q})(s, q) \neq 0$,
3. $\alpha(q) \in \mathcal{D}(\Sigma_\mathcal{O} \cup \Sigma_\tau \times Q)$ for $q \neq \hat{q}$,
4. $\forall a \in (\Sigma_\mathcal{S} \cup \Sigma_\mathcal{O})\ .\ a \notin \mathrm{CyclesA}(M)$,
5. $\mathbf{P}(\mathrm{CPaths}(M) \cap \mathrm{Paths}^\star(M)) = 1$.

The first two restrictions are on the initial state and mean that only secret actions can happen there (1) and each of those actions must have non null probability and occur only once (2), Restriction 3 forbids secret actions to happen in the rest of the automaton, and Restriction 4 ensures that the channel associated to the IHS has finitely many inputs and outputs. Finally, Restriction 5 means that infinite computations have probability 0 and therefore we can ignore them.

We now show how to interpret an IHS as a noisy channel. We call $trace_{\Sigma_\mathcal{S}}(\sigma)$ and $trace_{\Sigma_\mathcal{O}}(\sigma)$ the *secret* and *observable* traces of σ, respectively. For $s \in \Sigma_\mathcal{S}^*$, we define $[s] \triangleq \{\sigma \in \mathrm{CPaths}(M) \mid trace_{\Sigma_\mathcal{S}}(\sigma) = s\}$; similarly for $o \in \Sigma_\mathcal{O}^*$, we define $[o] \triangleq \{\sigma \in \mathrm{CPaths}(M) \mid trace_{\Sigma_\mathcal{O}}(\sigma) = o\}$.

Definition 3.2. Given an IHS $\mathcal{I} = (M, \Sigma_\mathcal{S}, \Sigma_\mathcal{O}, \Sigma_\tau)$, its noisy channel is $(\mathcal{S}, \mathcal{O}, P)$, where $\mathcal{S} \triangleq \Sigma_\mathcal{S}$, $\mathcal{O} \triangleq \mathit{trace}_{\Sigma_\mathcal{O}}(\mathrm{CPaths}(M))$, and $P(o \mid s) \triangleq \mathbf{P}([o] \mid [s])$. The a priori distribution $\pi \in \mathcal{D}(\mathcal{S})$ of $\mathcal{I}$ is defined by $\pi(s) \triangleq \alpha(\hat{q})(s, \cdot)$. If $\mathcal{C}$ is the noisy channel of $\mathcal{I}$, the multiplicative and additive leakage of $\mathcal{I}$ are naturally defined as

$$\mathcal{L}_\times(\mathcal{I}) \triangleq \mathcal{L}_\times(\mathcal{C}, \pi) \quad \text{and} \quad \mathcal{L}_+(\mathcal{I}) \triangleq \mathcal{L}_+(\mathcal{C}, \pi).$$

Example 3.3. Crowds [16] is a well-known anonymity protocol, in which a user (called the *initiator*) wants to send a message to a web server without revealing his identity. To achieve this, he routes the message through a crowd of users participating in the protocol. Routing is as follows. In the beginning, the initiator randomly selects a user (called a *forwarder*), possibly himself, and forwards the request to him. A forwarder performs a probabilistic choice. With probability p (a parameter of the protocol) he selects a new user and again forwards the message. With probability $1-p$ he sends the message directly to the server. One or more users can be *corrupted* and collaborate with each other to try to find the identity of the initiator.

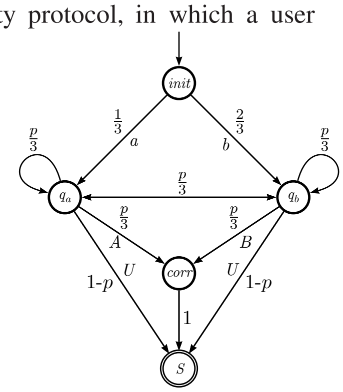

Fig. 1: Crowds Protocol

We now show how to model Crowds as an IHS for 2 honest and 1 corrupted user. We assume that the corrupted user immediately forwards messages to the server, as there is no further information to be gained for him by bouncing the message back.

Figure 1 shows the automaton[2]. Actions a and b are secret and represent who initiates the protocol; actions A, B, and U are observable; A and B represent who forwards the message to the corrupted user; U represents the fact that the message arrives at the server undetected by the corrupted user. We assume U to be observable to represent the possibility that the message is made publically available at the server's site.

The channel associated to this IHS has $\mathcal{S} = \{a, b\}$, $\mathcal{O} = \{A, B, U\}$, and a priori distribution $\pi(a) = \frac{1}{3}, \pi(b) = \frac{2}{3}$. Its channel matrix is computed in the next section.

4 Reachability analysis approach

This section presents a method to compute the matrix of joint probabilities $P_\wedge$ associated to an IHS, defined as

$$P_\wedge(s, o) \triangleq \mathbf{P}([s] \cap [o]) \text{ for all } s \in \mathcal{S} \text{ and } o \in \mathcal{O}.$$

We omit the subscript $\wedge$ when no confusion arises. From $P_\wedge$ we can derive the channel matrix by dividing $P_\wedge(s, o)$ by $\pi(s)$. The leakage can be computed directly from $P_\wedge$, using the second form of the a posteriori vulnerability in (1).

We write x_q^λ for the probability of the set of paths with trace $\lambda \in (\Sigma_\mathcal{S} \cup \Sigma_\mathcal{O})^\star$ starting from the state q of M:

$$x_q^\lambda \triangleq \mathbf{P}_q([\lambda]_q),$$

[2] For the sake of simplicity, we allow the initiator of the protocol to send the message to the server also in the first step of the protocol.

where $[\lambda]_q \triangleq \{\sigma \in \mathrm{CPaths}_q(M) \mid \mathit{trace}_{\Sigma_\mathcal{S} \cup \Sigma_\mathcal{O}}(\sigma) = \lambda\}$. The following key lemma shows the linear relation between the x_q^λ's. We assume, w.l.o.g., that the IHS has a unique final state q_f.

Lemma 4.1. *Let $\mathcal{I} = (M, \Sigma_\mathcal{S}, \Sigma_\mathcal{O}, \Sigma_\tau)$ be an* IHS. *For all $\lambda \in (\Sigma_\mathcal{S} \cup \Sigma_\mathcal{O})^\star$ and $q \in Q$ we have*

$$\begin{aligned}
x_{q_f}^\epsilon &= 1, \\
x_{q_f}^\lambda &= 0 \quad \textit{for } \lambda \neq \epsilon, \\
x_q^\epsilon &= \textstyle\sum_{h \in \Sigma_\tau} \sum_{q' \in \mathrm{succ}(q)} \alpha(q)(h, q') \cdot x_{q'}^\epsilon \quad \textit{for } q \neq q_f, \\
x_q^\lambda &= \textstyle\sum_{q' \in \mathrm{succ}(q)} \alpha(q)(\mathrm{first}(\lambda), q') \cdot x_{q'}^{\mathrm{tail}(\lambda)} \\
&\quad + \textstyle\sum_{h \in \Sigma_\tau} \alpha(q)(h, q') \cdot x_{q'}^\lambda \qquad \textit{for } \lambda \neq \epsilon \textit{ and } q \neq q_f.
\end{aligned}$$

Furthermore, for $s \in \mathcal{S}$ and $o \in \mathcal{O}$ we have $\mathbf{P}([s] \cap [o]) = x_{\hat{q}}^{so}$.

Using this lemma, one can compute joint probabilities by solving the system of linear equations in the variables x_q^λ's. It is possible that the system has multiple solutions; in that case the required solution is the minimal one.

Example 4.2. Continuing with the Crowds example, we show how to compute joint probabilities. Note that $q_f = S$. The linear equations from Lemma 4.1 are

$$\begin{array}{lll}
x_{\mathit{init}}^{aA} = \frac{1}{3} \cdot x_{q_a}^A, & x_{q_a}^A = \frac{p}{3} \cdot x_{q_a}^A + \frac{p}{3} \cdot x_{q_b}^A + \frac{p}{3} \cdot x_{\mathit{corr}}^\epsilon, & x_{\mathit{corr}}^A = x_S^A, \\
x_{\mathit{init}}^{bA} = \frac{2}{3} \cdot x_{q_b}^A, & x_{q_b}^A = \frac{p}{3} \cdot x_{q_a}^A + \frac{p}{3} \cdot x_{q_b}^A + \frac{p}{3} \cdot x_{\mathit{corr}}^A, & x_S^A = 0, \\
x_{\mathit{init}}^{aB} = \frac{1}{3} \cdot x_{q_a}^B, & x_{q_a}^B = \frac{p}{3} \cdot x_{q_a}^B + \frac{p}{3} \cdot x_{q_b}^B + \frac{p}{3} \cdot x_{\mathit{corr}}^B, & x_{\mathit{corr}}^B = x_S^B, \\
x_{\mathit{init}}^{bB} = \frac{2}{3} \cdot x_{q_b}^B, & x_{q_b}^B = \frac{p}{3} \cdot x_{q_a}^B + \frac{p}{3} \cdot x_{q_b}^B + \frac{p}{3} \cdot x_{\mathit{corr}}^\epsilon, & x_S^B = 0, \\
x_{\mathit{init}}^{aU} = \frac{1}{3} \cdot x_{q_a}^U, & x_{q_a}^U = \frac{p}{3} \cdot x_{q_a}^U + \frac{p}{3} \cdot x_{q_b}^U + (1-p) \cdot x_S^\epsilon, & x_{\mathit{corr}}^\epsilon = x_S^\epsilon, \\
x_{\mathit{init}}^{bU} = \frac{2}{3} \cdot x_{q_b}^U, & x_{q_b}^U = \frac{p}{3} \cdot x_{q_a}^U + \frac{p}{3} \cdot x_{q_b}^U + (1-p) \cdot x_S^\epsilon, & x_S^\epsilon = 1.
\end{array}$$

4.1 Complexity Analysis

We now analyze the computational complexity for the computation of the channel matrix of a simple IHS. Note that the only variables (from the system of equations in Lemma 4.1) that are relevant for the computation of the channel matrix are those x_q^λ for which it is possible to get the trace λ starting from state q. As a rough overestimate, for each state q, there are at most $|\mathcal{S}| \cdot |\mathcal{O}|$ λ's possible: in the initial state one can have every secret and every observable, in the other states no secret is possible and only a suffix of an observable can occur. This gives at most $|Q| \cdot |\mathcal{S}| \cdot |\mathcal{O}|$ variables. Therefore, we can straightforwardly obtain the desired set of values in $O((|Q| \cdot |\mathcal{S}| \cdot |\mathcal{O}|)^3)$ time (using Gaussian Elimination). Note that using Strassen's methods the exponent reduces to 2.807, this consideration applies to similar results in the rest of the paper as well.

Because secret actions can happen only at the beginning, the system of equations has a special form. The variables of the form $x_{\hat{q}}^{so}$ only depend on variables of the form x_q^o (with varying o and $q \neq \hat{q}$) and not on each other. Hence, we can first solve for all variables of the form x_q^o and then compute the remaining few of the form $x_{\hat{q}}^{so}$. Required time for the first step is $O((|\mathcal{O}| \cdot |Q|)^3)$ and the time for the second step can be ignored.

Finally, in some cases not only do the secret actions happen only at the beginning of the protocol, but the observable actions happen only at the end of the protocol, i.e., after taking a transition with an observable action, the protocol only performs internal actions (this is, for instance, the case for our model of Crowds). In this case, one might as well enter a unique terminal state q_f after an observable action happens. Then the only relevant variables are of the form $x_{\hat{q}}^{so}$, x_q^o, and $x_{q_f}^{\epsilon}$; the $x_{\hat{q}}^{so}$ only depends on the x_q^o, the x_q^o only depend on $x_{q'}^o$ (with the same o, but varying q's) and on $x_{q_f}^{\epsilon}$ and $x_{q_f}^{\epsilon} = 1$. Again ignoring the variables $x_{\hat{q}}^{so}$ for complexity purposes, the system of equations has a block form with $|\mathcal{O}|$ blocks of (at most) $|Q|$ variables each. Hence the complexity in this case decreases to $O(|\mathcal{O}| \cdot |Q|^3)$.

5 The Iterative Approach

We now propose a different approach to compute channel matrices and leakage. The idea is to iteratively construct the channel matrix of a system by adding probabilities of sets of paths containing paths with the same observable trace o and secret trace s to the $(o|s)$ entry of the matrix.

One reason for this approach is that it allows us to borrow techniques from quantitative counterexample generation. This includes the possibility of using or extending counterexample generation tools to compute channel matrices or leakage. Another reason for this approach is the relationship with debugging. If a (specification of a) system has a high leakage, the iterative approach allows us to determine which parts of the system contribute most to the high leakage, possibly pointing out flaws of the protocol. Finally, if the system under consideration is very large, the iterative approach allows us to only approximate the leakage (by not considering all paths, but only the most relevant ones) under strict guarantees about the accuracy of the approximation. We will focus on the multiplicative leakage; similar results can be obtained for the additive case.

5.1 Partial matrices

We start by defining a sequence of matrices converging to the channel matrix by adding the probability of complete paths one by one. We also define partial version of the a posteriori vulnerability and the leakage. Later, we show how to use techniques from quantitative counterexample generation to add probabilities of many (maybe infinitely many) complete paths all at once.

Definition 5.1. Let $\mathcal{I} = (M, \Sigma_{\mathcal{S}}, \Sigma_{\mathcal{O}}, \Sigma_{\tau})$ be an IHS, π its a priori distribution, and $\sigma_1, \sigma_2, \ldots$ an enumeration of the set of complete paths of M. We define the *partial matrices* $P^k : \mathcal{S} \times \mathcal{O} \to [0, 1]$ as follows

$$P^0(o|s) \triangleq 0, \qquad P^{k+1}(o|s) \triangleq \begin{cases} P^k(o|s) + \frac{\mathbf{P}(\langle \sigma_{k+1} \rangle)}{\pi(s)} & \text{if } \mathit{trace}_{\Sigma_{\mathcal{O}}}(\sigma_{k+1}) = o \\ & \text{and } \mathit{trace}_{\Sigma_{\mathcal{S}}}(\sigma_{k+1}) = s, \\ P^k(o|s) & \text{otherwise.} \end{cases}$$

We define the *partial vulnerability* $V_{S,O}^k$ as $\sum_o \max_s P^k(o|s) \cdot \pi(s)$, and the *partial multiplicative leakage* $\mathcal{L}_{\times}^k(\mathcal{I})$ as $V_{S,O}^k / \max_s \pi(s)$.

The following lemma states that partial matrices, a posteriori vulnerability, and leakage converge to the correct values.

Lemma 5.2. *Let* $\mathcal{I} = (M, \Sigma_{\mathcal{S}}, \Sigma_{\mathcal{O}}, \Sigma_\tau)$ *be an* IHS. *Then*

1. $P^k(o|s) \leq P^{k+1}(o|s)$, *and* $\lim_{k\to\infty} P^k(o|s) = P(o|s)$,
2. $V^k_{\mathrm{S,O}} \leq V^{k+1}_{\mathrm{S,O}}$, *and* $\lim_{k\to\infty} V^k_{\mathrm{S,O}} = \mathrm{V(S|O)}$,
3. $\mathcal{L}^k_\times(\mathcal{I}) \leq \mathcal{L}^{k+1}_\times(\mathcal{I})$, *and* $\lim_{k\to\infty} \mathcal{L}^k_\times(\mathcal{I}) = \mathcal{L}_\times(\mathcal{I})$.

Since rows must sum up to 1, this technique allow us to compute matrices up to given error ϵ. We now show how to estimate the error in the approximation of the multiplicative leakage.

Proposition 5.3. *Let* $(M, \Sigma_{\mathcal{S}}, \Sigma_{\mathcal{O}}, \Sigma_\tau)$ *be an* IHS. *Then we have*

$$\mathcal{L}^k_\times(\mathcal{I}) \leq \mathcal{L}_\times(\mathcal{I}) \leq \mathcal{L}^k_\times(\mathcal{I}) + \sum_{i=1}^{|\mathcal{S}|}(1 - p^k_i),$$

where p^k_i *denotes the mass probability of the* i*-th row of* P^k, *i.e.* $p^k_i \triangleq \sum_o P^k(o|s_i)$.

5.2 On the computation of partial matrices.

After showing how partial matrices can be used to approximate channel matrices and leakage we now turn our attention to accelerating the convergence. Adding most likely paths first is an obvious way to increase the convergence rate. However, since automata with cycles have infinitely many paths, this (still) gives an infinite amount of path to process. Processing many paths at once (all having the same observable and secret trace) tackles both issues at the same time: it increases the rate of convergence and can deal with infinitely many paths at the same time,

Interestingly enough, these issues also appear in *quantitative counterexample generation*. In that area, several techniques have already been provided to meet the challenges; we show how to apply those techniques in the current context. We consider two techniques: one is to group paths together using regular expression, the other is to group path together using strongly connected component analysis.

Regular expressions. In [9], regular expressions containing probability values are used to reason about traces in Markov Chains. This idea is used in [8] in the context of counterexample generation to group together paths with the same observable behaviour. The regular expression there are over pairs $\langle p, q\rangle$ with p a probability value and q a state, to be able to track both probabilities and observables. We now use the same idea to group together paths with the same secret action and the same observable actions.

We consider regular expressions over triples of the form $\langle a, p, q\rangle$ with $p \in [0, 1]$ a probability value, $a \in \Sigma$ an action label and $q \in Q$ a state. Regular expressions represent sets of paths as in [8]. We also take the probability value of such a regular expression from that paper.

Definition 5.4. The function $val : \mathcal{R}(\Sigma) \to \mathbb{R}$ evaluates regular expressions:

$$\begin{array}{llll} val(\epsilon) \triangleq 1, & val(r \cdot r') \triangleq val(r) \times val(r'), & & \\ val(\langle a, p, q\rangle) \triangleq p, & val(r^*) \triangleq 1 & \text{if } val(r) = 1, \\ val(r + r') \triangleq val(r) + val(r'), & val(r^*) \triangleq \frac{1}{1 - val(r)} & \text{if } val(r) \neq 1. \end{array}$$

The idea is to obtain regular expressions representing sets of paths of M, each regular expression will contribute in the approximation of the channel matrix and leakage. Several algorithms to translate automata into regular expressions have been proposed (see [14]). Finally, each term of the regular expression obtained can be processed separately by adding the corresponding probabilities [9] to the partial matrix.

As mentioned before, all paths represented by the regular expression should have the same observable and secret trace in order to be able to add its probability to a single element of the matrix. To ensure that condition we request the regular expression to be normal, i.e., of the form $r_1 + \cdots + r_n$ with the r_i containing no $+$'s.

For space reasons, instead of showing technical details we only show an example.

Example 5.5. We used JFLAP 7.0 [12] to obtain the regular expression $r \triangleq r_1 + r_2 + \cdots + r_{10}$ equivalent to the automaton in Figure 1.

$$\begin{aligned}
r_1 &\triangleq \langle b, \tfrac{2}{3}, q_b\rangle \cdot \hat{r}^\star \cdot \langle B, 0.3, \mathit{corr}\rangle \cdot \langle \tau, 1, S\rangle,\\
r_2 &\triangleq \langle b, \tfrac{2}{3}, q_b\rangle \cdot \hat{r}^\star \cdot \langle \tau, 0.3, q_a\rangle \cdot \langle \tau, 0.3, q_a\rangle^\star \cdot \langle A, 0.3, \mathit{corr}\rangle \cdot \langle \tau, 1, S\rangle,\\
r_3 &\triangleq \langle a, \tfrac{1}{3}, q_a\rangle \cdot \langle \tau, 0.3, q_a\rangle^\star \cdot \langle A, 0.3, \mathit{corr}\rangle \cdot \langle \tau, 1, S\rangle,\\
r_4 &\triangleq \langle b, \tfrac{2}{3}, q_b\rangle \cdot \hat{r}^\star \cdot \langle U, 0.1, S\rangle,\\
r_5 &\triangleq \langle a, \tfrac{1}{3}, q_a\rangle \cdot \langle \tau, 0.3, q_a\rangle^\star \cdot \langle \tau, 0.3, q_b\rangle \cdot \hat{r}^\star \cdot \langle B, 0.3, \mathit{corr}\rangle \cdot \langle \tau, 1, S\rangle,\\
r_6 &\triangleq \langle b, \tfrac{2}{3}, q_b\rangle \cdot \hat{r}^\star \cdot \langle \tau, 0.3, q_a\rangle \cdot \langle \tau, 0.3, q_a\rangle^\star \cdot \langle U, 0.1, S\rangle,\\
r_7 &\triangleq \langle a, \tfrac{1}{3}, q_a\rangle \cdot \langle \tau, 0.3, q_a\rangle^\star \cdot \langle U, 0.1, S\rangle,\\
r_8 &\triangleq \langle a, \tfrac{1}{3}, q_a\rangle \cdot \langle \tau, 0.3, q_a\rangle^\star \cdot \langle \tau, 0.3, q_b\rangle \cdot \hat{r}^\star \cdot \langle \tau, 0.3, q_a\rangle \cdot \langle \tau, 0.3, q_a\rangle^\star \cdot\\
&\quad \langle A, 0.3, \mathit{corr}\rangle \cdot \langle \tau, 1, S\rangle,\\
r_9 &\triangleq \langle a, \tfrac{1}{3}, q_a\rangle \cdot \langle \tau, 0.3, q_a\rangle^\star \cdot \langle \tau, 0.3, q_b\rangle \cdot \hat{r}^\star \cdot \langle U, 0.1, S\rangle,\\
r_{10} &\triangleq \langle a, \tfrac{1}{3}, q_a\rangle \cdot \langle \tau, 0.3, q_a\rangle^\star \cdot \langle \tau, 0.3, q_b\rangle \cdot \hat{r}^\star \cdot \langle \tau, 0.3, q_a\rangle \cdot \langle \tau, 0.3, q_a\rangle^\star \cdot \langle U, 0.1, S\rangle,
\end{aligned}$$

where $\hat{r} \triangleq (\langle \tau, 0.3, q_b\rangle^\star \cdot (\langle \tau, 0.3, q_a\rangle \cdot \langle \tau, 0.3, q_a\rangle^\star \cdot \langle \tau, 0.3, q_b\rangle)^\star)$. We also note

$$\begin{aligned}
&\mathit{val}(r_1) = \tfrac{7}{20}\,(b,B), \quad \mathit{val}(r_2) = \tfrac{3}{20}\,(b,A), \quad \mathit{val}(r_3) = \tfrac{1}{7}\,(a,A), \quad \mathit{val}(r_4) = \tfrac{7}{60}\,(b,U),\\
&\mathit{val}(r_5) = \tfrac{3}{40}\,(a,B), \quad \mathit{val}(r_6) = \tfrac{1}{20}\,(b,U), \quad \mathit{val}(r_7) = \tfrac{1}{21}\,(a,U), \quad \mathit{val}(r_8) = \tfrac{9}{280}\,(a,A),\\
&\mathit{val}(r_9) = \tfrac{1}{40}\,(a,U), \quad \mathit{val}(r_{10}) = \tfrac{3}{280}\,(a,U),
\end{aligned}$$

where the symbols between brackets denote the secret and observable traces of each regular expression.

Now we have all the ingredients needed to define partial matrices using regular expressions.

Definition 5.6. Let $\mathcal{I} = (M, \Sigma_{\mathcal{S}}, \Sigma_{\mathcal{O}}, \Sigma_\tau)$ be an IHS, π its a priori distribution, and $r = r_1 + r_2 + \cdots + r_n$ a regular expression equivalent to M in normal form. We define for $k = 0, 1, \ldots, n$ the matrices $P^k : \mathcal{S} \times \mathcal{O} \to [0,1]$ as follows

$$P^k(o|s) = \begin{cases} 0 & \text{if } k = 0,\\ P^{k-1}(o|s) + \frac{\mathit{val}(r_k)}{\pi(s)} & \text{if } k \neq 0 \text{ and } \mathit{trace}_{\Sigma_{\mathcal{O}}}(r_k) = o\\ & \text{and } \mathit{trace}_{\Sigma_{\mathcal{S}}}(r_k) = s,\\ P^{k-1}(o|s) & \text{otherwise.} \end{cases}$$

Note that in the context of Definition 5.6, we have $P^n = P$.

SCC **analysis approach.** In [2], paths that only differ in the way they traverse strongly connected components (SCC's) are grouped together. Note that in our case, such paths have the same secret and observable trace since secret and observable actions cannot occur on cycles. Following [2], we first abstract away the SCC's, leaving only probabilistic transitions that go immediately from an entry point of the SCC to an exit point (called input and output states in [2]). This abstraction happens in such a way that the observable behaviour of the automaton does not change.

Again, instead of going into technical details (which also involves translating the work [2] from Markov chains to fully probabilistic automata), we show an example.

Example 5.7. Figure 2 shows the automaton obtained after abstracting SCC. In the following we show the set of complete paths of the automaton, together with their corresponding probabilities and traces

$$
\begin{array}{lll}
\sigma_1 \triangleq \mathit{init} \xrightarrow{a} q_a \xrightarrow{A} \mathit{corr} \xrightarrow{\tau} S, & \mathbf{P}(\sigma_1) = \frac{7}{40}, & (a, A), \\
\sigma_2 \triangleq \mathit{init} \xrightarrow{b} q_b \xrightarrow{B} \mathit{corr} \xrightarrow{\tau} S, & \mathbf{P}(\sigma_2) = \frac{7}{20}, & (b, B), \\
\sigma_3 \triangleq \mathit{init} \xrightarrow{a} q_a \xrightarrow{U} S, & \mathbf{P}(\sigma_3) = \frac{1}{12}, & (a, U), \\
\sigma_4 \triangleq \mathit{init} \xrightarrow{b} q_b \xrightarrow{U} S, & \mathbf{P}(\sigma_4) = \frac{1}{6}, & (b, U), \\
\sigma_5 \triangleq \mathit{init} \xrightarrow{a} q_a \xrightarrow{B} \mathit{corr} \xrightarrow{\tau} S, & \mathbf{P}(\sigma_5) = \frac{3}{40}, & (a, B), \\
\sigma_6 \triangleq \mathit{init} \xrightarrow{b} q_b \xrightarrow{A} \mathit{corr} \xrightarrow{\tau} S, & \mathbf{P}(\sigma_6) = \frac{3}{20}, & (b, A).
\end{array}
$$

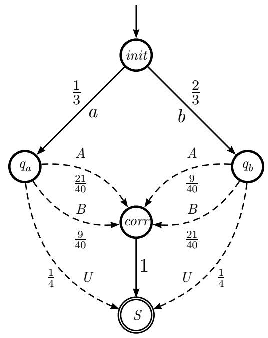

Fig. 2: Crowds after the SCC analysis

Note that the SCC analysis approach groups more paths together (for instance σ_1 group together the same paths than the regular expressions r_3 and r_8 in the examples of this section), as a result channel matrix and leakage are obtained faster. On the other hand, regular expressions are more informative providing more precise feedback.

5.3 Identifying high-leakage sources

We now describe how to use the techniques presented in this section to identify sources of high leakage of the system. Remember that the a posteriori vulnerability can be expressed in terms of joint probabilities

$$V(S \mid O) \;=\; \sum_{o} \max_{s} \mathbf{P}([s] \cap [o]).$$

This suggests that, in case we want to identify parts of the system generating high leakage, we should look at the sets of paths $[o_1] \cap [s_1], \ldots, [o_n] \cap [s_n]$ where $\{o_1, \ldots o_n\} = \mathcal{O}$ and $s_i \in \arg\left(\max_s \mathbf{P}([o_i] \cap [s])\right)$. In fact, the multiplicative leakage is given dividing $V(S \mid O)$ by $V(S)$, but since $V(S)$ is a constant value (i.e., it does not depend on the row) it does not play a role here. Similarly for the additive case.

The techniques presented in this section allow us to obtain such sets and, furthermore, to partition them in a convenient way with the purpose of identifying states/parts of the system that contribute the most to its high probability. Indeed, this is the aim of the counterexamples generation techniques previously presented. For further details on how to debug sets of paths and why these techniques meet that purpose we refer to [1, 8, 2].

Example 5.8. To illustrate these ideas, consider the path σ_1 of the previous example; this path has maximum probability for the observable A. By inspecting the path we find the transition with high probability $q_a \xrightarrow{A}$ *corr*. This suggests to the debugger that the corrupted user has an excessively high probability of intercepting a message from user a in case he is the initiator.

In case the debugger requires further information on how corrupted users can intercept messages, the regular expression approach provides further/more-detailed information. For instance, we obtain further information by looking at regular expressions r_3 and r_8 instead of path σ_1 (in particular it is possible to visualize the different ways the corrupted user can intercept the message of user a when he is the generator of the message).

6 Information Hiding Systems with Variable a Priori

In Section 3 we introduced a notion of IHS in which the distribution over secrets is fixed. However, when reasoning about security protocols this is often not the case. In general we may assume that an adversary knows the distribution over secrets in each particular instance, but the protocol should not depend on it. In such scenario we want the protocol to be secure, i.e. ensuring low enough leakage, for every possible distribution over secrets. This leads to the definition of maximum leakage.

Definition 6.1 ([19, 3]). Given a noisy channel $\mathcal{C} = (\mathcal{S}, \mathcal{O}, P)$, we define the maximum multiplicative and additive leakage (respectively) as

$$\mathcal{ML}_\times(\mathcal{C}) \triangleq \max_{\pi \in \mathcal{D}(\mathcal{S})} \mathcal{L}_\times(\mathcal{C}, \pi), \qquad \text{and} \qquad \mathcal{ML}_+(\mathcal{C}) \triangleq \max_{\pi \in \mathcal{D}(\mathcal{S})} \mathcal{L}_+(\mathcal{C}, \pi).$$

In order to model this new scenario where the distribution over secrets may change, the selection of the secret is modeled as *nondeterministic choice*. In this way such a distribution remains undefined in the protocol/automaton. We still assume that the choice of the secret happens at the beginning, and that we have only one secret per run. We call such automaton an IHS *with variable a priori*.

Definition 6.2. An IHS with variable a priori is a quadruple $\mathcal{I} = (M, \Sigma_\mathcal{S}, \Sigma_\mathcal{O}, \Sigma_\tau)$ where $M = (Q, \Sigma, \hat{q}, \alpha)$ is a probabilistic automaton, $\Sigma = \Sigma_\mathcal{S} \cup \Sigma_\mathcal{O} \cup \Sigma_\tau$ where $\Sigma_\mathcal{S}$, $\Sigma_\mathcal{O}$, and Σ_τ are pairwise disjoint sets of secret, observable, and internal actions, and α satisfies the following restrictions:

1. $\alpha(\hat{q}) \subseteq \mathcal{D}(\Sigma_\mathcal{S} \times Q)$,
2. $|\alpha(\hat{q})| = |\mathcal{S}| \wedge \forall s \in \Sigma_\mathcal{S} \,.\, \exists\, q \,.\, \pi(s, q) = 1$, for some $\pi \in \alpha(\hat{q})$,
3. $\alpha(q) \subseteq \mathcal{D}(\Sigma_\mathcal{O} \cup \Sigma_\tau \times Q)$ and $|\alpha(q)| \leq 1$, for all $q \neq \hat{q}$,
4. $\forall a \in (\Sigma_\mathcal{S} \cup \Sigma_\mathcal{O}) \,.\, a \notin \mathrm{CyclesA}(M)$,
5. $\forall q, s\ \forall \pi \in \alpha(\hat{q}) \,.\, (\pi(s, q) = 1 \Rightarrow \mathbf{P}(\mathrm{CPaths}_q(M) \cap \mathrm{Paths}_q^*(M)) = 1)$.

Restrictions 1, 2 and 3 imply that the secret choice is non deterministic and happens only at the beginning. Additionally, 3 means that all the other choices are probabilistic.

Restriction 4 ensures that the channel associated to the IHS has finitely many inputs and outputs. Finally, 5 implies that, after we have chosen a secret, every computation terminates except for a set with null probability.

Given an IHS with variable a priori, by fixing the a priori distribution we can obtain a standard IHS in the obvious way:

Definition 6.3. Let $\mathcal{I} = ((Q, \Sigma, \hat{q}, \alpha), \Sigma_{\mathcal{S}}, \Sigma_{\mathcal{O}}, \Sigma_\tau)$ be an IHS with variable a priori and π a distribution over $\mathcal{S}$. We define the IHS associated to $(\mathcal{I}, \pi)$ as $\mathcal{I}_\pi = ((Q, \Sigma, \hat{q}, \alpha'), \Sigma_{\mathcal{S}}, \Sigma_{\mathcal{O}}, \Sigma_\tau)$ with $\alpha'(q) = \alpha(q)$ for all $q \neq \hat{q}$ and $\alpha'(\hat{q})(s, \cdot) = \pi(s)$.

The following result says that the conditional probabilities associated to an IHS with variable a priori are *invariant* with respect to the a priori distribution. This is fundamental in order to interpret the IHS as a channel.

Proposition 6.4. *Let $\mathcal{I}$ be an* IHS *with variable a priori. Then for all $\pi, \pi' \in \mathcal{D}(\mathcal{S})$ such that $\pi(s) \neq 0$ and $\pi'(s) \neq 0$ for all $s \in \mathcal{S}$ we have that $P_{\mathcal{I}_\pi} = P_{\mathcal{I}_{\pi'}}$.*

Proof. The secret s appears only once in the tree and only at the beginning of paths, hence $\mathbf{P}([s] \cap [o]) = \alpha'(\hat{q})(s, \cdot)\, \mathbf{P}_{q_s}([o])$ and $\mathbf{P}([s]) = \alpha'(\hat{q})(s, \cdot)$. Therefore $\mathbf{P}([o] \mid [s]) = \mathbf{P}_{q_s}([o])$, where q_s is the state after performing s. While $\alpha'(\hat{q})(s, \cdot)$ is different in $\mathcal{I}_\pi$ and $\mathcal{I}_{\pi'}$, $\mathbf{P}_{q_s}([o])$ is the same, because it only depends on the parts of the paths after the choice of the secret. □

Note that, although in the previous proposition we exclude input distributions with zeros, the concepts of vulnerability and leakage also make sense for these distributions[3].

This result implies that we can define the channel matrix of an IHS $\mathcal{I}$ with variable a priori as the channel matrix of $\mathcal{I}_\pi$ for any π, and we can compute it, or approximate it, using the same techniques of previous sectionsSimilarly we can compute or approximate the leakage for any given π.

We now turn the attention to the computation of the maximum leakage. The following result from the literature is crucial for our purposes.

Proposition 6.5 ([3]). *Given a channel $\mathcal{C}$, $\arg\max_{\pi \in \mathcal{D}(\mathcal{S})} \mathcal{L}_\times(\mathcal{C}, \pi)$ is the uniform distribution, and $\arg\max_{\pi \in \mathcal{D}(\mathcal{S})} \mathcal{L}_+(\mathcal{C}, \pi)$ is a* corner point *distribution, i.e. a distribution π such that $\pi(s) = \frac{1}{\kappa}$ on κ elements of $\mathcal{S}$, and $\pi(s) = 0$ on all the other elements.*

As an obvious consequence, we obtain:

Corollary 6.6. *Given an* IHS *$\mathcal{I}$ with variable a priori, we have $\mathcal{ML}_\times(\mathcal{I}) = \mathcal{L}_\times(\mathcal{I}_\pi)$, where π is the uniform distribution, and $\mathcal{ML}_+(\mathcal{I}) = \mathcal{L}_+(\mathcal{I}_{\pi'})$, where π' is a corner point distribution.*

Corollary 6.6 gives us a method to compute the maxima leakages of $\mathcal{I}$. In the multiplicative case the complexity is the same as for computing the leakage[4]. In the additive case we need to find the right corner point, which can be done by computing the

[3] We assume that conditional probabilities are extended by continuity on such distributions.

[4] Actually we can compute it even faster using an observation from [19] which says that the leakage on the uniform distribution can be obtained simply by summing up the maximum elements of each column of the channel matrix.

leakages for all corner points and then comparing them. This method has exponential complexity (in $|\mathcal{S}|$) as the size of the set of corner points is $2^{|\mathcal{S}|} - 1$. We conjecture that this complexity is intrinsic, i.e. that the problem is NP-hard[5].

7 Interactive Information Hiding Systems

We now consider extending the framework to interactive systems, namely to IHS's in which the secrets and the observables can alternate in an arbitrary way. The secret part of a run is then an element of $\Sigma_{\mathcal{S}}^*$, like the observable part is an element of $\Sigma_{\mathcal{O}}^*$. The idea is that such system models an interactive play between a source of secret information, and a protocol or program that may produce, each time, some observable in response. Since each choice is associated to one player of this "game", it seems natural to impose that in a choice the actions are either secret or observable/hidden, but not both.

The main novelty and challenge of this extension is that part of the secrets come after observable events, and may depend on them.

Definition 7.1. Interactive IHS's are defined as IHS's (Definition 3.1), except that Restrictions 1 to 3 are replaced by $\alpha(q) \in \mathcal{D}(\Sigma_{\mathcal{S}} \times Q) \cup \mathcal{D}(\Sigma - \Sigma_{\mathcal{S}} \times Q)$.

Example 7.2. Consider an Ebay-like auction protocol with one seller and two possible buyers, one rich and one poor. The seller first publishes the item he wants to sell, which can be either cheap or expensive. Then the two buyers start bidding. At the end, the seller looks at the profile of the bid winner and decides whether to sell the item or cancel the transaction. Figure 7 illustrates the automaton representing the protocol, for certain given probability distributions.

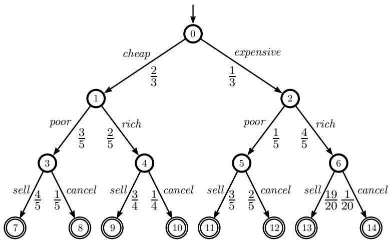

Fig. 3: Ebay Protocol

We assume that the identities of the buyers are secret, while the price of the item and the seller's decision are observable. We ignore for simplicity the hidden actions which are performed during the bidding phase. Hence $\Sigma_{\mathcal{O}} = \{cheap, expensive, sell, cancel\}$, $\Sigma_{\tau} = \emptyset$, $\mathcal{S} = \Sigma_{\mathcal{S}} = \{poor, rich\}$, and $\mathcal{O} = \{cheap, expensive\} \times \{sell, cancel\}$. The distributions on $\mathcal{S}$ and $\mathcal{O}$ are defined as usual. For instance we have $\mathbf{P}([cheap\ \ sell]) = \mathbf{P}(\{q_0 \xrightarrow{cheap} q_1 \xrightarrow{poor} q_3 \xrightarrow{sell} q_7, q_0 \xrightarrow{cheap} q_1 \xrightarrow{rich} q_3 \xrightarrow{sell} q_7\}) = \frac{2}{3} \cdot \frac{3}{5} \cdot \frac{4}{5} + \frac{2}{3} \cdot \frac{2}{5} \cdot \frac{3}{4} = \frac{13}{25}$.

Let us now consider how to model the protocol in terms of a noisy channel. It would seem natural to define the channel associated to the protocol as the triple $(\mathcal{S}, \mathcal{O}, P)$ where $P(o \mid s) = \mathbf{P}([o] \mid [s]) = \frac{\mathbf{P}([s] \cap [o])}{\mathbf{P}([s])}$. This is, indeed, the approach taken in [11]. For instance, with the protocol of Example 7.2, we would have:

$$\mathbf{P}([cheap\ \ sell] \mid [poor]) = \frac{\mathbf{P}([poor] \cap [cheap\ \ sell])}{\mathbf{P}([poor])} = \frac{\frac{2}{3} \cdot \frac{3}{5} \cdot \frac{4}{5}}{\frac{2}{3} \cdot \frac{3}{5} + \frac{1}{3} \cdot \frac{1}{5}} = \frac{24}{35}. \quad (2)$$

However, it turns out that in the interactive case (in particular when the secrets are not in the initial phase), it does not make sense to model the protocol in terms of a channel.

[5] Since submitting this paper, we have proved that our conjecture is true.

At least, not a channel with input $\mathcal{S}$. In fact, the matrix of a channel is supposed to be *invariant* with respect to the input distribution (like in the case of the IHS's with variable a priori considered in previous section), and this is not the case here. The following is a counterexample.

Example 7.3. Consider the same protocol as in Example 7.2, but assume now that the distribution over the choice of the buyer is uniform, i.e. $\alpha(q_1)(poor, q_3) = \alpha(q_1)(rich, q_4) = \alpha(q_2)(poor, q_5) = \alpha(q_2)(rich, q_6) = \frac{1}{2}$. Then the conditional probabilities are different than those for Example 7.2. In particular, in contrast to (2), we have

$$\mathbf{P}([cheap\ \ sell] \mid [poor]) = \frac{\mathbf{P}([poor] \cap [cheap\ \ sell])}{\mathbf{P}([poor])} = \frac{\frac{2}{3} \cdot \frac{1}{2} \cdot \frac{4}{5}}{\frac{2}{3} \cdot \frac{1}{2} + \frac{1}{3} \cdot \frac{1}{2}} = \frac{8}{15}.$$

The above observation, i.e. the fact that the conditional probabilities depend on the input distribution, makes it unsound to reason about certain information-theoretic concepts in the standard way. For instance, the *capacity* is defined as the maximum mutual information over all possible input distributions, and the traditional algorithms to compute it are based on the assumption that the channel matrix remains the same while the input distribution variates. This does not make sense anymore in the interactive setting.

However, when the input distribution is fixed, the matrix of the joint probabilities is well defined as $P_\wedge(s, o) = \mathbf{P}([s] \cap [o])$, and can be computed or approximated using the same methods as for simple IHS's. The a priori probability and the channel matrix can then be derived in the standard way:

$$\pi(s) = \sum_o P_\wedge(s, o), \qquad\qquad P(o \mid s) = \frac{P_\wedge(s, o)}{\pi(s)}.$$

Thanks to the formulation (1) of the a posteriori vulnerability, the leakage can be computed directly using the joint probabilities.

Example 7.4. Consider the Ebay protocol $\mathcal{I}$ presented in Example 7.2. The matrix of the joint probabilities $P_\wedge(s, o)$ is:

	$cheap\ sell$	$cheap\ cancel$	$expensive\ sell$	$expensive\ cancel$
$poor$	$\frac{8}{25}$	$\frac{2}{25}$	$\frac{1}{25}$	$\frac{2}{75}$
$rich$	$\frac{1}{5}$	$\frac{1}{15}$	$\frac{19}{75}$	$\frac{1}{75}$

Furthermore $\pi(poor) = \frac{7}{15}$ and $\pi(rich) = \frac{8}{15}$. Hence we have $\mathcal{L}_\times(\mathcal{I}) = \frac{51}{40}$ and $\mathcal{L}_+(\mathcal{I}) = \frac{11}{75}$.

We note that our techniques to compute channel matrices and leakage extend smoothly to the case where secrets are not required to happen at the beginning. However, no assumptions can be made about the occurrences of secrets (they do not need to occur at the beginning anymore). This increases the complexity of the reachability technique to $O((|\mathcal{S}| \cdot |\mathcal{O}| \cdot |Q|)^3)$. On the other hand, complexity bounds for the iterative approach remain the same.

8 Related Work

To the best of our knowledge, this is the first work dealing with the efficient computation of channel matrices and leakage. However, for the simple scenario, channel matrices can be computed using standard model checking techniques. Chatzikokolakis et al. [4] have used Prism [15] to model Crowds as a Markov Chain and compute its channel matrix. Each conditional probability $P(o|s)$ is computed as the probability of reaching a state where o holds starting from *the* state where s holds. Since for the simple version of IHS's secrets occur only once and before observables (as in Crowds), such a reachability probability equals $P(o|s)$. This procedure leads to $O(|\mathcal{S}| \cdot |\mathcal{O}| \cdot |\overline{Q}|^3)$ time complexity to compute the channel matrix, where $\overline{Q}$ is the space state of the Markov Chain.

Note that the complexity is expressed in terms of the space state of a Markov Chain instead of automaton. Since Markov Chains do not carry information in transitions they have a larger state space than an equivalent automaton. Figure 8 illustrates this: to model the automaton (left hand side) we need to encode the information in its transitions into states of the Markov Chain (right hand side). Therefore, the probability of seeing observation a and then c in the automaton can be computed as the probability of reaching the state ac. The Markov Chain used for modeling Crowds (in our two honest and one corrupted user configuration) has 27 states.

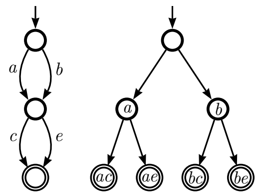

Fig. 4: Automaton vs Markov Chain

For this reason we conjecture that our complexity $O(|\mathcal{O}| \cdot |Q|^3)$ is a considerable improvement over the one on Markov Chains $O(|\mathcal{S}| \cdot |\mathcal{O}| \cdot |\overline{Q}|^3)$.

With respect to the interactive scenario, standard model checking techniques do not extend because multiple occurrences of the same secret are allowed (for instance in our Ebay example, $P(cheap\ sell|rich)$ cannot be derived from reachability probabilities from the two different states of the automaton where $rich$ holds).

9 Conclusion and Future Work

In this paper we have addressed the problem of computing the information leakage of a system in an efficient way. We have proposed two methods: one based on reachability techniques; the other based on quantitative counterexample generation.

We plan to use tools developed for counterexamples generation (in particular the Prism implementation of both techniques presented in Section 5) in order to compute/approximate leakage of large scale protocols. We also intend to investigate in more depth how the results obtained from those tools can be used to identify flaws of the protocol causing high leakage.

In Section 7 we have shown that when the automaton is interactive we cannot define its channel in the standard way. An intriguing problem is how to extend the notion of channel so to capture the dynamic nature of interaction. One idea is to use channels with history and/or feedback. Another idea is to lift the inputs from secrets to schedulers on secrets, i.e. to functions from paths to distributions over secrets.

References

1. H. Aljazzar and S. Leue. Debugging of dependability models using interactive visualization of counterexamples. In *Proc. of the Int. Conf. on Quantitative Evaluation of SysTems 2008*, pages 189–198, Los Alamitos, CA, USA, 2008. IEEE Press.
2. M. E. Andrés, P. D'Argenio, and P. van Rossum. Significant diagnostic counterexamples in probabilistic model checking. In H. Chockler and A. J. Hu, editors, *Proc. of HVC'08*, volume 5394 of *LNCS*, pages 129–148. Springer, 2009.
3. C. Braun, K. Chatzikokolakis, and C. Palamidessi. Quantitative notions of leakage for one-try attacks. In *Proc. of the 25th Conf. on Mathematical Foundations of Programming Semantics*, volume 249 of *ENTCS*, pages 75–91. Elsevier B.V., 2009.
4. K. Chatzikokolakis, C. Palamidessi, and P. Panangaden. Anonymity protocols as noisy channels. *Inf. and Comp.*, 206(2–4):378–401, 2008.
5. D. Clark, S. Hunt, and P. Malacaria. Quantitative information flow, relations and polymorphic types. *J. of Logic and Computation*, 18(2):181–199, 2005.
6. M. R. Clarkson, A. C. Myers, and F. B. Schneider. Belief in information flow. *Journal of Computer Security*, 2008. To appear.
7. T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., second edition, 2006.
8. B. Damman, T. Han, and J.-P. Katoen. Regular expressions for PCTL counterexamples. In *Proc. of the Int. Conf. on Quantitative Evaluation of SysTems 2008*, pages 179–188, Los Alamitos, CA, USA, 2008. IEEE Press.
9. C. Daws. Symbolic and parametric model checking of discrete-time markov chains. In Z. Liu and K. Araki, editors, *ICTAC*, volume 3407 of *LNCS*, pages 280–294. Springer, 2005.
10. Y. Deng, J. Pang, and P. Wu. Measuring anonymity with relative entropy. In T. Dimitrakos, F. Martinelli, P. Y. A. Ryan, and S. A. Schneider, editors, *Proc. of the of the 4th Int. Worshop on Formal Aspects in Security and Trust*, volume 4691 of *LNCS*, pages 65–79. Springer, 2006.
11. J. Desharnais, R. Jagadeesan, V. Gupta, and P. Panangaden. The metric analogue of weak bisimulation for probabilistic processes. In *Proceedings of the 17th Annual IEEE Symposium on Logic in Computer Science*, pages 413–422. IEEE Computer Society, 2002.
12. Jflap website. `http://www.jflap.org/`.
13. I. S. Moskowitz, R. E. Newman, D. P. Crepeau, and A. R. Miller. Covert channels and anonymizing networks. In S. Jajodia, P. Samarati, and P. F. Syverson, editors, *Workshop on Privacy in the Electronic Society 2003*, pages 79–88. ACM, 2003.
14. C. Neumann. Converting deterministic finite automata to regular expressions. 2005. `http://neumannhaus.com/christoph/papers/2005-03-16.DFA_to_RegEx.pdf`.
15. Prism website. `http://www.prismmodelchecker.org`.
16. M. K. Reiter and A. D. Rubin. Crowds: anonymity for Web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.
17. A. Sabelfeld and A. C. Myers. Language-based information flow security. *IEEE Journal on Selected Areas in Communications*, 21(1):5–19, Jan. 2003.
18. R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, June 1995. Tech. Rep. MIT/LCS/TR-676.
19. G. Smith. On the foundations of quantitative information flow. In L. De Alfaro, editor, *Proc. of the 12th Int. Conf. on Foundations of Software Science and Computation Structures*, volume 5504 of *LNCS*, pages 288–302, York, UK, 2009. Springer.