Formal Analysis of the Information Leakage of the DC-Nets and Crowds Anonymity Protocols

Arthur Américo[1], Artur Vaz[1], Mário S. Alvim[1], Sérgio V. A. Campos[1], and Annabelle McIver[2]

[1] Universidade Federal de Minas Gerais, Brazil
[2] Macquarie University, Australia

Abstract. A crucial goal in computer security is to protect sensitive information from unwanted disclosure. However, some leakage is often unavoidable, be it by design of the system or by technological limitations. The field of *Quantitative Information Flow* (QIF) is concerned with the quantification, and limitation, of information leakage in systems.
The QIF framework models systems as information-theoretic *channels* taking *(secret) inputs* and producing *(observable) outputs*, thereby increasing the adversary's knowledge about the secret value, as measured by some *information metric.*
In this paper we use probabilistic model checking to obtain channels modeling two popular anonymity protocols, the *Dining Cryptographers* (a.k.a. *DC-Nets*) and *Crowds*, in two versions each. We then derive the systems' *capacities* w.r.t. the g-leakage framework, which are robust upper bounds on information leakage that hold irrespectively of the probability distribution on secret values, or of the interests and goals of the adversary. To the best of our knowledge, this is the most general QIF analyses of such protocols.

Keywords: Quantitative Information Flow, Formal Methods, Model Checking, Dining Cryptographers, Crowds, g-leakage.

1 Introduction

Protecting sensitive information is a crucial goal of computational security, and the more dependent human affairs are on computational systems, the more pressing becomes the matter. Ideally, we would like to prevent *all* leakage of sensitive information, but this might not be achievable in practice. For example, a password checker on an ATM will always leak some information—either by accepting the user's input (which completely reveals the password value), or by rejecting it (which rules out one possible value).

Nevertheless we use ATMs regularly, and many other systems suffering from similar issues . We are comfortable in doing so not because they do not leak sensitive information, but because we consider the *amount* of information they leak to be "acceptable". In fact, most systems, either by technical limitation or by design, leak some information, and developing ways to measure *how much* information is leaked is essential in order to analyze the security of such systems. However, quantifying leakage or guaranteeing that it is limited is a difficult task.

Quantitative Information Flow (QIF) is the branch of security that studies the amount of information leaked by a system. It has seen growing interest over the past decade, including foundational works [15,22,11,35,27,8,4], verification of information flow properties [16,24,6,10,14,13,36], detection of real system vulnerabilities [20,23], and, of course, methods to reduce information leakage.

In QIF, security systems are modeled as information-theoretical *channels*, from which various properties of interest can be deduced. One crucial, and in general non-trivial task, however, is to compute the channel corresponding to the behavior of a given computational system—even for small but intricate protocols.

In this paper we describe a general procedure to derive such channels using *probabilistic model checking*. Using this procedure we model the *Dining Cryptographers* (a.k.a. *DC-Nets*) [12], and *Crowds* [33], two well-known anonymity protocols, in two variations each: (i) the standard DC-Nets, in which nodes are organized in a ring; (ii) a version of DC-Nets in which nodes are all connected to each other; (iii) the standard Crowds protocols; and (iv) a version of Crowds in which nodes are organized in a grid and each node can only communicate with its immediate neighbors. We then analyze them using the state of art in QIF metrics: the *g-vulnerability* framework [5]. More precisely, we derive *g-capacities* [3] of such channels, which are robust upper bounds on the information leakage they may present in any possible context of execution. This means that the bounds computed hold irrespectively of the probability distribution on secret values, or of the interests and goals of an adversary. To the best of our knowledge, this is the the most general information-flow analyses of such protocols.

The main contributions of this paper are:

1. Allowing anonymity protocols to be expressed in a precise modeling language which closely reflects their implementation.
2. A direct computation of the relevant channels.
3. The first characterization of the g-capacities of the Dining Cryptographers and the Crowds anonymity protocols, which are state-of-the-art robust measures of information flow.
4. A detailed comparison of the superiority of one variant of each protocol over the other in terms of information leakage guarantees.

Future work could lead to a general purpose tool support to allow the computation of critical information flow properties.

This remaining of this paper is organized as follows. Section 2 reviews necessary background on QIF and on probabilistic model checking, including the PRISM tool. Section 3 describes the Dining Cryptographers and the Crowds anonymity protocols, in two variations each. Section 4 describes the general procedure using probabilistic model checking to derive channels from protocols, and presents the channels produced for the protocols we study. Section 5 analyzes the channels obtained the in light of QIF metrics. Finally, Section 6 discusses related work, and 7 discusses future work, and concludes.

2 Preliminaries

In this section we review basic concepts from quantitative information flow, and from probabilistic model checking.

2.1 Quantitative Information Flow

Secrets and vulnerability. A *secret* is some piece of sensitive information the defender wants to protect, such as a user's password, social security number, or current location. The attacker usually only has some partial knowledge about the value of a secret, represented as a probability distribution on secrets called a *prior*. We denote by $\mathcal{X}$ the set of possible secrets, and we typically use π to denote a prior belonging to the set $\mathbb{D}\mathcal{X}$ of probability distributions over $\mathcal{X}$.

The *vulnerability* of a secret is a measure of the utility of the attacker's knowledge about the secret. Several notions of vulnerability (or their dual concept, *entropy*) have been proposed in the literature, including Shannon entropy [34], guessing entropy [25], and Bayes vulnerability/risk [35,11].

Recently, the *g-vulnerability* framework [5] has been proposed, consisting of a family of vulnerability measures that capture various adversarial models. It has been shown that these functions coincide with the set of continuous and convex functions on $\mathbb{D}\mathcal{X}$, and are, in a precise sense, the most general information measures w.r.t. a set of basic axioms. [3] In this paper we shall adopt g-vulnerabilities as our measures of information.

The operational scenario captured by g-vulnerabilities is parameterized by a set $\mathcal{W}$ of *guesses* (possibly infinite) that the attacker can take w.r.t. a secret, and a *gain function* $g : \mathcal{W} \times \mathcal{X} \to \mathbb{R}$. The gain $g(w, x)$ expresses the attacker's benefit for making guess w when the actual secret is x. Given a distribution π, *(prior) g-vulnerability* measures the attacker's success as the expected gain of an optimal guess, being defined as

$$V_g\left[\pi\right] \overset{\text{def}}{=} \max_{w \in \mathcal{W}} \sum_{x \in \mathcal{X}} \pi(x) g(w, x).$$

Channels, posterior vulnerability, and leakage. Systems can be modeled as information theoretic channels. A *channel matrix*, or simply a *channel*, $C : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is a function in which $\mathcal{X}$ is a set of *input values*, $\mathcal{Y}$ is a set of *output values*, and $C(x, y)$ represents the conditional probability of the channel producing output $y \in \mathcal{Y}$ when input $x \in \mathcal{X}$ is provided. Every channel C satisfies $0 \leq C(x, y) \leq 1$ for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, and $\sum_{y \in \mathcal{Y}} C(x, y) = 1$ for all $x \in \mathcal{X}$.

A distribution $\pi \in \mathbb{D}\mathcal{X}$ and a channel C with inputs $\mathcal{X}$ and outputs $\mathcal{Y}$ induce a joint distribution $p(x, y) = \pi(x) C(x, y)$ on $\mathcal{X} \times \mathcal{Y}$, producing joint random variables X, Y with marginal probabilities $p(x) = \sum_y p(x, y)$ and $p(y) =$

[3] More precisely, if posterior vulnerability is defined as the expectation of the vulnerability of posterior distributions, the measure respects the data-processing inequality and yields non-negative leakage iff vulnerability is convex.

$\sum_x p(x,y)$, and conditional probabilities $p(x|y) = {}^{p(x,y)}/_{p(y)}$ if $p(y) \neq 0$. For a given y (s.t. $p(y) \neq 0$), the conditional probabilities $p(x|y)$ for each $x \in \mathcal{X}$ form the *posterior distribution* $p_{X|y}$.[4]

A channel C in which $\mathcal{X}$ is a set of secret values and $\mathcal{Y}$ is a set of observable values produced by a system can be used to model computations on secrets. Assuming the attacker has prior knowledge π about the secret value, knows how a channel C works, and can observe the channel's outputs, the effect of the channel is to update the attacker's knowledge from a prior π to a collection of posteriors $p_{X|y}$, each occurring with probability $p(y)$.[5]

Example 1. Given $\mathcal{X} = \{x_1, x_2, x_3\}$ and $\mathcal{Y} = \{y_1, y_2, y_3, y_4\}$, and the channel matrix C below, the (uniform) prior $\pi = (1/3, 1/3, 1/3)$ combined with C leads to the joint matrix J as follows.

$$\begin{array}{|c|cccc|} \hline C & y_1 & y_2 & y_3 & y_4 \\ \hline x_1 & 1 & 0 & 0 & 0 \\ x_2 & 0 & 1/2 & 1/4 & 1/4 \\ x_3 & 1/2 & 1/3 & 1/6 & 0 \\ \hline \end{array} \quad \xrightarrow{\pi} \quad \begin{array}{|c|cccc|} \hline J & y_1 & y_2 & y_3 & y_4 \\ \hline x_1 & 1/3 & 0 & 0 & 0 \\ x_2 & 0 & 1/6 & 1/12 & 1/12 \\ x_3 & 1/6 & 1/9 & 1/18 & 0 \\ \hline \end{array}$$

Summing the columns of J gives the marginal distribution $p_Y = (1/2, 5/18, 5/36, 1/12)$, and normalizing gives the posterior distributions $p_{X|y_1} = (2/3, 0, 1/3)$, $p_{X|y_2} = (0, 3/5, 2/5)$, $p_{X|y_3} = (0, 3/5, 2/5)$, and $p_{X|y_4} = (0, 1, 0)$. □

The *posterior vulnerability* is the vulnerability of the secret after the attacker observed the output of the channel. Formally, given a g-vulnerability V_g, the *posterior g-vulnerability* w.r.t. a prior π and a channel C is defined as

$$\begin{aligned} V_g[\pi, C] &\stackrel{\text{def}}{=} \sum_{y \in \mathcal{Y}} p(y) V_g(p(X|_y) \\ &= \sum_{y \in \mathcal{Y}} \max_{w \in \mathcal{W}} \sum_{x \in \mathcal{X}} \pi(x) C(x,y) g(w,x). \end{aligned}$$

The *information leakage* of a channel C under a prior π is a comparison between the vulnerability of the secret before the system was run—called *prior vulnerability*—and the posterior vulnerability of the secret. Leakage, then, reflects by how much the observation of the system's outputs increases the utility of the attacker's knowledge about the secret. It can be defined either

$$\textit{multiplicatively:} \qquad \mathcal{L}_g[\pi, C] = \frac{V_g[\pi, C]}{V_g[\pi]},$$

which measures the relative increase in the adversary's information about the secret; or

$$\textit{additively:} \qquad \mathcal{L}_g^+[\pi, C] = V_g[\pi, C] - V_g[\pi],$$

[4] To avoid ambiguity, we may write probabilities with subscripts, e.g., p_{XY} or p_Y.

[5] This collection of posterior distributions is, in fact, a distribution on (posterior) distributions, and is called a *hyper-distribution* on secrets [27].

which measures the absolute increase in the adversary's information.

Multiplicative and additive versions of leakage provide complimentary information about the behavior of a channel. Depending on the system itself, on the nature of the secret inputs, and even on the interests of the adversary, one definition of leakage may be more suitable than the other to express information leakage on a certain scenario, but in general a proper assessment of leakage may have to take both versions into consideration [3].

Capacities. Although both multiplicative and additive g-leakages represent useful quantities, to properly compute them one needs to know not only the channel C representing the system, but also the prior π and the gain-function g, and both can vary depending on the adversary's knowledge and interests. For robustness, we can consider *capacities*, which are leakage measures that universally quantify over the prior π, over the gain function g, or over both, making the measurements less dependent on the particular context in which the system will run.

Quantifying over the prior π acknowledges that, in many situations, it is unknown and the assumption that it is uniform is not reasonable. Quantifying over the gain function g acknowledges that we might not know the value to the adversary of different sorts of partial information about the secret, neither now nor even in the future. Combining all ways of quantifying over π and g (one, other, or both), and the two versions of leakage (multiplicative and additive), we arrive at a total of six types of capacities, which are depicted in Table 1.

Types of Capacities	Multiplicative Leakage	Additive Leakage
For all π, fixed g	$\mathcal{L}_g[\forall, C] = \max_{\pi} \mathcal{L}_g[\pi, C]$	$\mathcal{L}_g^+[\forall, C] = \max_{\pi} \mathcal{L}_g^+[\pi, C]$
Fixed π, for all g	$\mathcal{L}_\forall[\pi, C] = \max_{g} \mathcal{L}_g[\pi, C]$	$\mathcal{L}_\forall^+[\pi, C] = \max_{g} \mathcal{L}_g^+[\pi, C]$
For all π, for all g	$\mathcal{L}_\forall[\forall, C] = \max_{\pi,g} \mathcal{L}_g[\pi, C]$	$\mathcal{L}_\forall^+[\forall, C] = \max_{\pi,g} \mathcal{L}_g^+[\pi, C]$

Table 1: Types of capacities.

Although finding a way to compute the capacities $\mathcal{L}_g[\forall, C]$ and $\mathcal{L}_\forall^+[\forall, C]$ is still an open problem, there are known algorithms for computing the other four capacities [3]. More precisely, $\mathcal{L}_\forall[\pi, C]$, $\mathcal{L}_\forall[\forall, C]$, and $\mathcal{L}_\forall^+[\pi, C]$ can be computed in time linear on the size of the channel C. $\mathcal{L}_g^+[\forall, C]$, however, is NP-hard. We will use these capacities to compare our protocols in Section 5.

Capacities are upper bounds on the information leakage of a protocol over a variety of combinations of adversarial prior knowledge about the secret (captured by different priors), and of adversarial intentions and interests (captured by different gain functions). For this reason, they are particularly useful bounds on the leakage of channels that will execute in possibly unknown contexts.

2.2 Probabilistic Model Checking

Here we briefly review key concepts from probabilistic model checking, and some of the basic features of the model checker we use, PRISM[1]. Our formalism and notation are similar to that used by C. Baier and J.-P. Katoen [7], and D. Parker [31].

Discrete Time Markov Chains. Probabilistic model checking works by modeling the system of interest as a probabilistic automaton. All protocols in this paper are modeled as discrete-time Markov chains (DTMC).

A *discrete-time Markov chain* M is a tuple $M = (S, P, i, AP, L)$ such that S is a (finite and nonempty) set of *states*, $P : S \times S \to [0, 1]$ is the *probabilistic transition function*, $i \in S$ is the initial state, AP is the set of atomic propositions, and $L : S \to 2^{AP}$ is a labeling function. We also require that, for all $s \in S$, $\sum_{s' \in S} P(s, s') = 1$.

A *path* ω in a DTMC is a infinite sequence of states $s_0 s_1 ...$ such that, for all $k \geq 0$, $P(s_k, s_{k+1}) > 0$. Any execution of a DTMC corresponds to a path. Therefore, in order to reason about probabilities over executions of a DTMC, we must first associate a probability to each path. For each $s \in S$, we define $Path_s$ to be the set of all paths that start on s. A probability distribution $Prob_s$ over $Path_s$ is defined as follows. Let $\omega_f = s\, s_1 \ldots s_n$ be any finite path starting in s, and $Cyl(\omega_f)$ be the set of (infinite) paths that have ω_f as a prefix. Let Σ_s be the smallest σ-algebra on $Path_s$ that contains $Cyl(\omega_f)$ for all ω_f starting in s. We define $Prob_s$ as the unique probability distribution on Σ_s such that $Prob_s(Cyl(\omega_f)) = P(s, s_1)...P(s_{n-1}, s_n)$ for all finite paths ω_f starting in s.

PCTL. [18] The temporal logic used by PRISM to verify properties of DTMCs is the PCTL (Probabilistic Computational Tree Logic), whose syntax is given by:

$$\phi ::= true \mid a \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \mathcal{P}_{\bowtie p}(\psi)$$
$$\psi ::= \mathcal{X}\phi \mid \phi_1 \mathcal{U}^{\leq k} \phi_2 \mid \phi_1 \mathcal{U} \phi_2$$

Here ϕ represents state formulas and ψ path formulas, a is an atomic proposition, $p \in [0, 1]$, and $\bowtie$ is a symbol to represent either $\leq, <, >$ or $\geq$. The semantics of the *probabilistic path operators* $\mathcal{P}_{\bowtie p}$ is $s \models \mathcal{P}_{\bowtie p}(\psi) \Leftrightarrow Prob_s(\{\omega \in Path_s | \omega \models \psi\}) \bowtie p$, for all $s \in S$. The operators *next* ($\mathcal{X}$), *bounded until* ($\mathcal{U}^{\leq k}$) and *until* ($\mathcal{U}$) are defined as usual.

Intuitively, $\mathcal{P}_{\bowtie p}(\psi)$ is satisfied by a state s if the probability of taking a path starting at s which satisfies ψ is in the interval determined by $\bowtie p$. This operator allows PRISM to calculate probabilities of certain event ocurring, a feature that is extremely useful in calculating channels, as we discuss in section 4.

3 The Dining Cryptographers and the Crowds anonymity protocols

In this section we describe two well-known anonymity protocols from the literature, and their variations, the leakage analyses of which we performed.

3.1 The Dining Cryptographers protocol

The *Dining Cryptographers (DC)* anonymity protocol was proposed by David Chaum [12]. It is usually described within the following setting. Three cryptographers are invited by the NSA (The U.S. National Security Agency) to have dinner at a restaurant. Along with the invitation, one of them might have been secretly told by the NSA to pay the bill. Otherwise, the NSA itself would pay. The cryptographers wish to know whether one of them was asked to pay the bill (as opposed to the NSA paying the bill), without revealing, however, which one of them is the payer. In order to do so, they execute the following protocol.

Sitting in a round table, each cryptographer flips a coin, and shares the result with the cryptographer to their right. In this way each cryptographer sees the results of only two of the coins: the one he himself flipped, and the one flipped by the cryptographer sitting to his left. Each cryptographer then makes a public announcement. If he is not paying the bill, he announces 0 if the results of the two coins he sees are the same (i.e., both heads or both tails), and announces 1 if they are different. However, if the cryptographer is the payer, he announces 1 if the results of the two coins coincide, and announces 0 otherwise.

The cryptographers now can learn whether the NSA is paying: if the sum of all three announcements (modulo 2) equals 0, the NSA is paying. If the sum equals 1, then one of them is paying. This can be easily seen from the fact that the announcement of each cryptographer not paying the bill is the number of heads he has seen (modulo 2). If no one is paying, then the final result is equal to twice the number of coins that landed heads up to modulo 2, which is certainly 0. If one of them is paying, however, the final result will be 1.

If the coins are fair, the identity of the cryptographer who pays the bill is totally preserved, both in relation to the other two cryptographers and to any external observer. If the coins are biased, however, the announcements made by the cryptographers might make one of them more likely to be the payer than the others. For example, if the coin tosses are very likely to yield tails, and only one cryptographer announces 1, then he is probably paying the bill.

We are specially interested in scenarios with a biased coin, for some information is leaked by the protocol. We can use QIF to precisely quantify this leakage and we can determine by how much the attacker can improve his guessing strategy. In this paper we study two different generalizations of the DC protocol, which expand the number of cryptographers involved.

The cycle-DC variation. Our first variation of the DC protocol is akin to the original, but the number of cryptographers can be any integer greater than

2. Similarly to the original protocol, the cryptographers are arranged in a circular table, each tosses a coin and shares the result to the cryptographer at his right. The announcements are made in the same manner as before. Also in this scenario, one of the cryptographers is the payer if, and only if, the sum of the announcements (modulo 2) equals 1.

The complete-DC variation. In our second variation of the DC protocol, all pairs of cryptographers share a coin toss result (i.e., they form a clique). If there are N cryptographers, each one has access to $N-1$ coin-toss results. After all the coin tosses are made, each cryptographer computes the number of heads he has seen (modulo 2). If he is not paying for the bill, this is the number that he announces. If he is paying, however, he inverts the announcement. Since each heads is counted twice, we also have that one of the cryptographers is paying if, and only if, the sum of the announcements (modulo 2) equals 1.

3.2 The Crowds protocol

The *Crowds* protocol was first devised to protect anonymity on web transactions. Suppose there is a group of users who wish to make requests to a server, without revealing their identities to that server.

The users agree to cooperate on the protocol, and take the following steps. (1) If a user wants to send a request (we call such user an *initiator*), he chooses at random a user in the group (including himself), and forwards the request to this user. (2) If a user receives a request, he forwards it to a random user with probability p_f, and forwards it directly to the server with probability $1-p_f$. The second step is repeated until the request reaches the server.

The protocol protects the initiator's identity because, after being forwarded for the first time, the request has an equal probability of landing at any user of the system. Therefore, the server does not acquire any information by observing which user sent the request to him at the end of the process.

The analysis of the protocol becomes more interesting when there are some *corrupt* users in the group. These corrupt users are in collusion with the server, and reveal to it the identity of any regular user that sent them a request—in this case, we say that the regular user in question was *detected.* Because the initiator must be in any path of the message on its way to the server, whenever a user is detected, he is the most likely to be the initiator. As expected, the level of anonymity provided by the protocol in this scenario depends on the number of users, on the number of corrupt users, and on the probability p_f.

In this paper we consider two variants of the Crowds protocol.

The (original) Crowds variation. In this variation, each user can communicate with any other user (they form a clique), and there is no restriction on who can forward a message to whom.

The grid-Crowds variation. A common variation of this protocol occurs when a user is able to forward a request only to a subset of the remaining users. One particular instance of this scenario is when users are placed on a grid, as illustrated in Figure 1. Edges represent users who can communicate, and we consider the edges going off the grid to connect users at opposite sides, e.g., user 1 is connected to users 2, 3, 4 and 7. We consider that every user can also communicate directly with the server.

Other than this limitation, the protocol works as the original: upon receiving a request, each user forwards to a user with whom he can communicate (including himself) with probability p_f, or sends it directly to the server with probability $1-p_f$. In this scenario, even if there are no corrupt users, the server can infer some information about the originator of the request. For example, if the server receives a request from user 2 in Figure 1, there is a greater chance that it was originated by user 1 than by user 6.

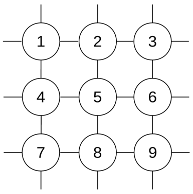


Fig. 1: 3×3 instance of grid-Crowds.

On this grid variation, the information leakage of the protocol depends not only on the number of corrupt users and on p_f, but also on where the corrupt users are located in the grid. One of our goals is to study the effects these topological variations have on QIF measures.

4 Deriving the channels corresponding to the protocols

In this section we show how to use the PRISM model checker [1] to derive the channels representing the behavior of the protocols we analyze.

The general procedure to compute the channel corresponding to a protocol is the following. (1) We identify the sets $\mathcal{X}$ and $\mathcal{Y}$ representing, respectively, the secret and observable values of the protocol. (2) We implement the protocol in PRISM, creating variables that can uniquely identify each element on the sets $\mathcal{X}$ and $\mathcal{Y}$. A variable that signals the end of the protocol's execution is used. (3) We set the variables accordingly for each value $x \in \mathcal{X}$, and use PRISM to calculate the conditional probability $p(y \mid x)$ for each $y \in \mathcal{Y}$.

The third step can easily be accomplished by observing the first step, with the aid of an operator present in PRISM. Given a model, it is possible to verify the probability of taking a path from the initial state that respects a property `pathprop` with the operator `P =? [pathprop]`. If the second step is correctly observed, there is, for any $y \in \mathcal{Y}$, a way to make `pathprop` equivalent to the path formula $\mathcal{F}y$, where $\mathcal{F}$ is the finally operator. This path formula holds if and only if the system's output equals y. By setting the variables of the system to make the secret value $x \in \mathcal{X}$ for all x, we can use this operator to systematically calculate $p(y|x)$ for every pair x, y, which defines our channel.

Next we illustrate how our general procedure can be applied to derive the channels corresponding to all variations of the protocols we consider.

4.1 Modeling the Dining Cryptographers

We now discuss how to derive the channel for both variations of the DC protocol: cycle-DC, and complete-DC. The first task is to characterize $\mathcal{X}$ and $\mathcal{Y}$, and to devise a suitable representation of them to implement in our code.

Let N denote the number of cryptographers in an instance of the Dining Cryptographers protocol. In both variations of the protocol, the secret value is the identity of who pays the bill. We have, then, $\mathcal{X} = \{c_1, c_2, ..., c_N, n\}$ where each c_i represents the case in which cryptographer c_i is the payer, and n represents in which the case the NSA pays.

The observable values of the protocol, in both variations, are the public announcements made by all cryptographers. We can represent these announcements by a string of N bits, where the value of the bit at position i corresponds to the announcement of the cryptographer c_i. To ilustrate, consider a protocol with four cryptographers. If c_1 and c_2 announced 1, and c_3 and c_4 announced 0, this would be represented by the string 1100. Hence, we can represent all possible outputs by taking $\mathcal{Y} = \{0, 1\}^N$.

Having established $\mathcal{X}$ and $\mathcal{Y}$, as well as how to translate their elements into variables in the code, we can write the protocol in PRISM language. Our implementation is available online [2]. Table 2 depicts the channels (omitting the NSA output) computed by PRISM for cycle- and complete-DC, with the probability of heads equal to 0.7.

$C_{\textit{cycle-DC}}$	1000	1100	0010	1110	0001	1101	1011	0111
c_1	0.2482	0.1218	0.0882	0.1218	0.1218	0.0882	0.1218	0.0882
c_2	0.1218	0.2482	0.1218	0.0882	0.0882	0.1218	0.0882	0.1218
c_3	0.0882	0.1218	0.2482	0.1218	0.1218	0.0882	0.1218	0.0882
c_4	0.1218	0.0882	0.1218	0.0882	0.2482	0.1218	0.0882	0.1218

(a) Channel $C_{\textit{cycle-DC}}$ for the cycle-DC protocol.

$C_{\textit{comp-DC}}$	1000	1100	0010	1110	0001	1101	1011	0111
c_1	0.1666	0.1218	0.1218	0.1218	0.1218	0.1218	0.1218	0.1026
c_2	0.1218	0.1666	0.1218	0.1218	0.1218	0.1218	0.1026	0.1218
c_3	0.1218	0.1218	0.1666	0.1218	0.1218	0.1026	0.1218	0.1218
c_4	0.1218	0.1218	0.1218	0.1026	0.1666	0.1218	0.1218	0.1218

(b) Channel $C_{\textit{comp-DC}}$ for the complete-DC protocol.

Table 2: Channels for both variations of the *Dining Cryptographers* protocol, with the probability of heads equal to 0.7

4.2 Modeling Crowds

We now discuss how to derive the channel for both variations of the Crowds protocol: original Crowds, and grid-Crowds. The first step is to identify what the sets $\mathcal{X}$ and $\mathcal{Y}$ shall represent, and to find a suitable implementation of them.

In both variations of the protocol, the secret value is the identity of the initiator of the request. There is no need to represent corrupt users, as we assume they do not initiate requests. Therefore, we can represent the secret values set of Crowds with N honest users by $\mathcal{X} = \{u_1, u_2, ..., u_N\}$.

The observable values are different in the two variations. In original Crowds, the server does not gain any information by identifying the user who forwarded the request to him, therefore we must have one output value d_i representing the scenario in which each honest user u_i was detected by a corrupt one, and another case s representing the scenario where the server receives the request. Therefore, we have $\mathcal{Y} = \{d_1, ..., d_N, s\}$. In grid-Crowds, however, the identity of a user that forwards a request to the server is relevant. We need, therefore, to break the output s into multiple ones, indicating which user forwarded the request to the server. Thus, in this second variation, we must have $\mathcal{Y} = \{d_1, ..., d_N, s_1, ..., s_N\}$.

Having determined $\mathcal{X}$ and $\mathcal{Y}$, it is possible to implement the protocols in the PRISM language. Our implementation of all protocols are available online [2].

5 QIF analyses of the protocols

We now analyze the information-leakage of the channels corresponding to the protocols. Recall that the smaller the capacity of a channel (c.f.r. Section 2.1), the less information an adversary will obtain about the secret by observing the output of that channel, and the safer, hence, the channel is considered.

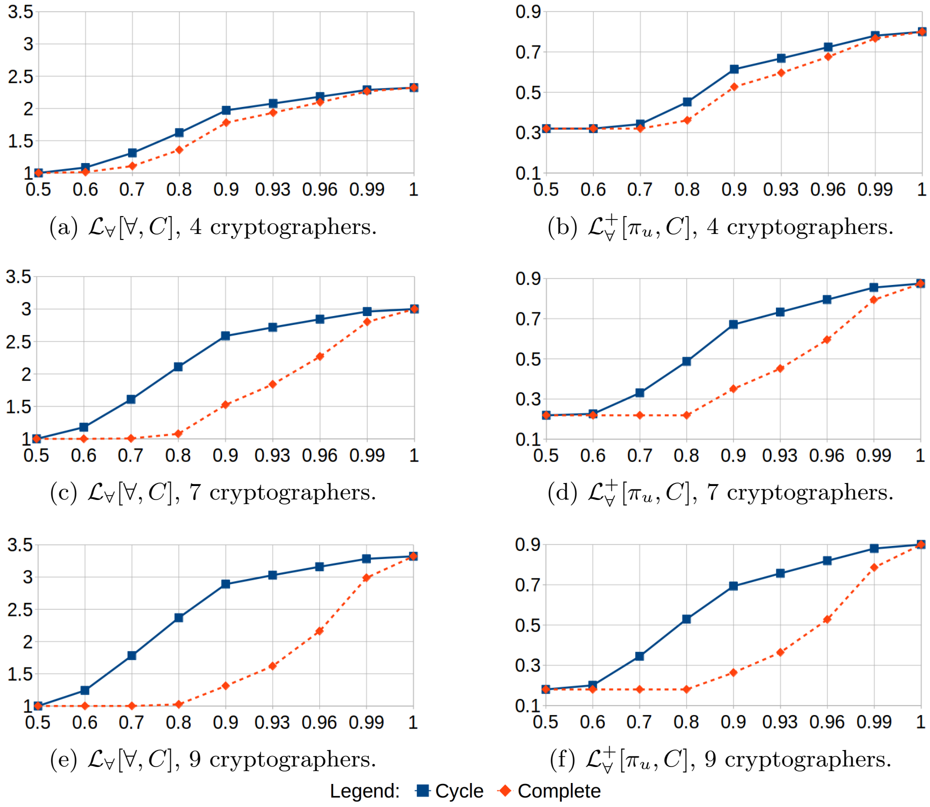
5.1 Analyses of the Dining Cryptographers

We implemented both variations of the Dining cryptographers for 5, 6, 7, 8, and 9 cryptographers. Our results suggest the complete-DC variation is safer than the cycle-DC variation, yielding smaller values for all capacities measured.

Results for multiplicative capacities. It has been proven [3] that the multiplicative capacity $\mathcal{L}_\forall[\pi, C]$ (which quantifies over all gain-functions g for a fixed prior π) collapses into the multiplicative capacity $\mathcal{L}_\forall[\forall, C]$ (which quantifies over all priors and gain-functions) when the prior π has full-support. Since we consider any cryptographer can be the payer, the prior has full support, and we can focus only on the latter capacity, which can be computed as follows.

Theorem 1 ([9]) *Given channel* $C{:}\mathcal{X}{\times}\mathcal{Y}{\to}\mathbb{R}$, $\mathcal{L}_\forall[\forall, C]{=}\log\sum_{y\in\mathcal{Y}}\max_{x\in\mathcal{X}} C(x,y)$.

Figure 2 shows the values of this capacity for both variants of the DC protocol, and varying values of the probability p of heads. Note that the graph of



(a) $\mathcal{L}_{\forall}[\forall, C]$, 4 cryptographers.

(b) $\mathcal{L}^{+}_{\forall}[\pi_u, C]$, 4 cryptographers.

(c) $\mathcal{L}_{\forall}[\forall, C]$, 7 cryptographers.

(d) $\mathcal{L}^{+}_{\forall}[\pi_u, C]$, 7 cryptographers.

(e) $\mathcal{L}_{\forall}[\forall, C]$, 9 cryptographers.

(f) $\mathcal{L}^{+}_{\forall}[\pi_u, C]$, 9 cryptographers.

Fig. 2: Capacities for both variations of the Dining Cryptographers, and different probabilities of heads. The x-axis, for the value of probability, is *not* in scale.

capacity must be symmetric w.r.t. $p = 0.5$, for a coin with probability of heads $1-p$ is nothing more than a coin with probability of tails p.

Note also that in all instances of both variations, the minimum capacity is 1, and occurs at $p = 0.5$. This reflects the fact that, if the coins are fair, the only information leaked is whether the NSA is paying the bill. When $p = 1$, however, all coin tosses yield heads, and any observer can deduce with certainty who is paying the bill from the announcements of the cryptographers. In this case the capacity reaches its maximum, being $\log(n + 1)$ for n cryptographers, for the outputs of the protocol always reveal the payer's identity.

It is clear from the graph that the complete variation leaks less than the cycle one. Also, while the capacity of the cycle variation increases rapidly even when p approaches 0.5, the complete variation is less susceptible to these small changes, maintaining information leakage close to 0.

Results for additive capacity. The additive capacity $\mathcal{L}^{+}_{\forall}[\pi, C]$ (quantifying over all gain functions g, for a given prior π) can be computed by $\mathcal{L}^{+}_{\forall}[\pi, C] = \sum_{x,y} \pi(x)|C(x, y) - \sum_{x'} \pi(x')C(x', y)|$ [3]. Note that, unlike its multiplicative counterpart, this capacity actually depends on the prior π.

Figure 2 shows the values for this capacity for an uniform prior π_u, and varying probabilities p of heads. The graphs confirm that the minimum and maximum values of information leakages occur, respectively, for $p = 0.5$ and $p = 1$. We can see that complete-DC is always more secure than cycle-DC, and its capacity keeps almost unaltered until p deviates substantially from 0.5.

5.2 Analyses of Crowds

We implemented both variants of the Crowds protocol for 9 users, and from 1 up to 3 corrupted users.

In particular, in grid-Crowds, we need to consider another variable: the position of the corrupt users makes a great difference in the channels' capacities. Figure 3 shows all possible positions for three corrupt users on a 3x3 grid, up to symmetry. (Recall that the edges going off the grid connect at opposite sides—e.g., user 1 can communicate with user 3 and with user 7.) If, for instance, the corrupt users are 1,2 and 3, each honest user would be connected to only one corrupt user. If the corrupt were 1,5 and 9, however, each honest user would be connected to two corrupt ones. Therefore, the chance that the initiator will forward a message directly to a corrupt user is 20% on the first scenario, and 40% on the second one (please recall that the initiator may send the message to itself). Thus, it is natural to expect that the capacities for the former will be smaller than for the latter.

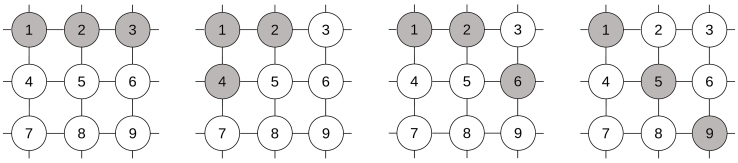
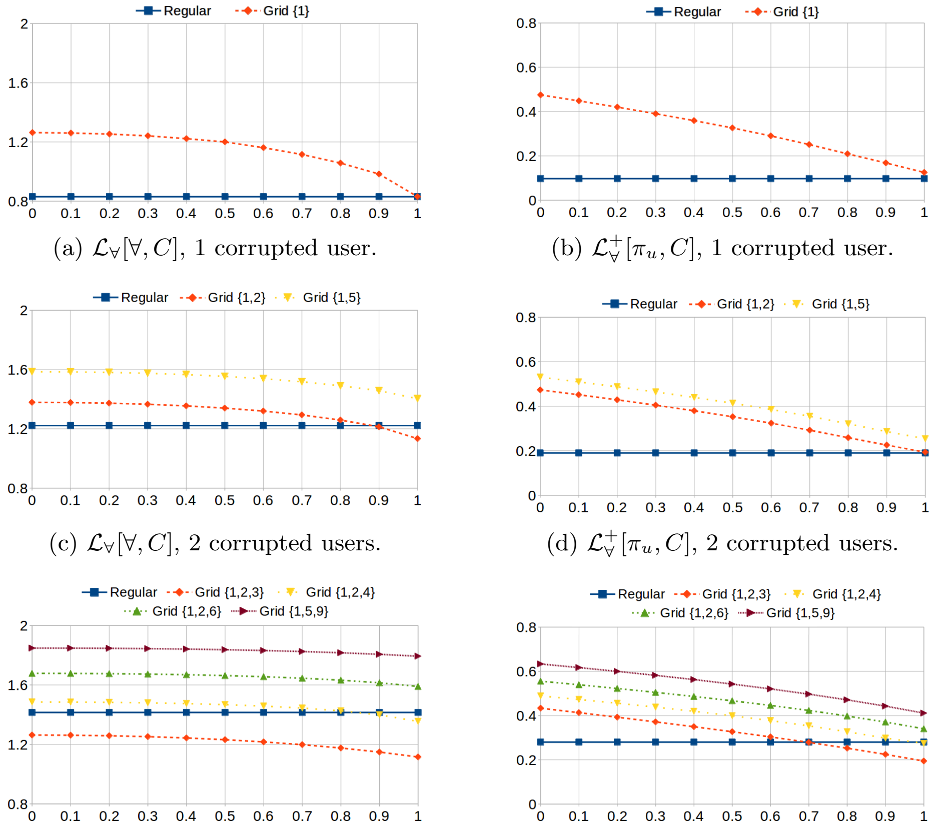


Fig. 3: All positions (up to symmetry) for three corrupted users (colored in dark) on a 3x3 grid.

Results for multiplicative capacities. We compute $\mathcal{L}_{\forall}[\forall, C]$ using Theorem 1. Figure 4 shows the corresponding values for 9 users, both in original DC and in grid-DC. Note that the probability p_f of forwarding does not influence multiplicative capacity in the original DC, which confirms a known result from the literature [17]. However, we can see that it does for the grid variation. To understand this, notice that even if the originator does not forward the message to a corrupt user at first, his immediate neighbors are more likely to receive the message than the users he cannot communicate with. For example, if user 8 is detected in the 3x3 grid, it is more probable that the originator was user 7 than user 4. Therefore, as the expected number of interactions between users decreases with p_f, the more likely it is that a message is detected by a corrupt user or forwarded to the server near its originator.



(a) $\mathcal{L}_\forall[\forall, C]$, 1 corrupted user.

(b) $\mathcal{L}^+_\forall[\pi_u, C]$, 1 corrupted user.

(c) $\mathcal{L}_\forall[\forall, C]$, 2 corrupted users.

(d) $\mathcal{L}^+_\forall[\pi_u, C]$, 2 corrupted users.

(e) $\mathcal{L}_\forall[\forall, C]$, 3 corrupted users.

(f) $\mathcal{L}^+_\forall[\pi_u, C]$, 3 corrupted users.

Fig. 4: Capacities for both variations of Crowds, and varying number of corrupt users. The brackets in the captions indicate positions of corrupt users.

As we can verify, the capacities of the channels on the grid variation varies considerably according to the position of the corrupt users. The results also reiterate our intuition that, for the 3x3 grid with three corrupt users, the protocol where 1,2 and 3 are the corrupt users would be the safer one.

It is also interesting to notice that some choice of corrupt users in grid-DC actually yield smaller capacities than the original variation. For 9 users with 3 corrupt ones, for instance, the odds of the initiator being detected are 33%. As we have seen, when the corrupt users are 1, 2 and 3 in the 3x3 grid, this chance is 20%, and our data suggest that this difference is sufficient to compensate the extra leakage usually caused by the grid structure.

Results for additive capacity. To calculate this capacity, we again use the equation on section 5.1 and consider a uniform prior π_u. The results are shown on Figure 4. We can verify that the additive capacities for the original Crowds protocol also do not vary with p_f. Also, they behave quite differently from their multiplicative counterparts. For example, consider again the protocol with 3 corrupt users in positions 1, 2 and 3 in the 3x3 grid. The multiplicative capacity of this scenario is always smaller that of the regular protocol, but this is not true at all for the additive capacity.

6 Related work

In this paper we have explored the use of model checking to compute bounds on quantitative information flow. Our approach is to express protocols as transition systems and then to use probabilistic model checking to compute a channel abstraction. The benefits of this approach are that protocols can be expressed in a direct way, and their abstraction as a channel can be easily computed. By computing the whole channel, rather than say a specific leakage or capacity measure, we make it available for use with any appropriate gain function.

Other work on computing information flow [28] gives a semantics of programs as hidden Markov models, of which channels are a special case. This allows hyper-distributions—a compact form of posterior joint distributions—to be computed directly. With this generality the monadic features of functional programming languages can be exploited to compute leakage w.r.t. arbitrary gain functions [29].

Other approaches to computing information flow typically use alternative measures. For example, McCamant and Ernst [26] provide estimates for the quantity of bits flowing from input to output (of programs) using network flow capacity. Novakovic [30] uses a model based on mutual information (Shannon) and min-entropy; the starting point for the analysis is a program expressed in a probabilistic imperative language, with an interpretation based on DTMCs. High Order Logic theorem provers were used by Hölz and Nipkow[21] to study properties of the Crowds protocol and its behavior regarding Shannon entropy, and by Helali et al. [19] to derive general results regarding min-entropy and belief min-entropy. Finally, Phan et al. [32] use reliability analysis to quantify leaks, also based on Shannon- and min-entropy.

7 Conclusions and Future Work

In this work we presented a systematic way of deriving channels representing the behavior of security protocols, and used these channels to derive robust information-flow guarantees about these protocols. More precisely, we provided the first analyses of additive and multiplicative g-capacities of two versions of the Dining Cryptographers and the Crowds anonymity protocols. The bounds provided hold irrespectively of the probability distribution on secret values, or of the interests and goals of an adversary, constituting, to the best of our knowledge, the most general information-flow analyses of such protocols ever performed.

Future work could lead to a general purpose tool support to allow the computation of critical information flow properties. Moreover, we want to explore algorithms for computing capacities for systems whose possible contexts of execution are limited in a more restricted set of priors and gain-functions.

Acknowledgments Atrhur Américo, Artur Vaz, Mário S. Alvim, and Sérgio V. A. Campos were supported by CNPq, CAPES, and FAPEMIG.

References

1. PRISM: A Probabilistic Symbolic Model Checker. `www.prismmodelchecker.org/`.
2. `http://homepages.dcc.ufmg.br/~arturvaz/sbmf/`.
3. M. S. Alvim, K. Chatzikokolakis, A. McIver, C. Morgan, C. Palamidessi, and G. Smith. Additive and multiplicative notions of leakage, and their capacities. In *Proc. of CSF*, pages 308–322. IEEE, 2014.
4. M. S. Alvim, K. Chatzikokolakis, A. McIver, C. Morgan, C. Palamidessi, and G. Smith. Axioms for information leakage. In *Proc. of CSF*, pages 77–92, 2016.
5. M. S. Alvim, K. Chatzikokolakis, C. Palamidessi, and G. Smith. Measuring information leakage using generalized gain functions. In *Proc. of CSF*, pages 265–279, 2012.
6. M. E. Andrés, C. Palamidessi, P. van Rossum, and G. Smith. Computing the leakage of information-hiding systems. In *Proc. of TACAS*, volume 6015 of *LNCS*, pages 373–389. Springer, 2010.
7. C. Baier and J.-P. Katoen. *Principles of Model Checking.* The MIT Press, 2008.
8. M. Boreale and F. Pampaloni. Quantitative information flow under generic leakage functions and adaptive adversaries. *Logical Methods in Computer Science*, 11(4), 2015.
9. C. Braun, K. Chatzikokolakis, and C. Palamidessi. Quantitative notions of leakage for one-try attacks. 249:75–91, 2009.
10. K. Chatzikokolakis, T. Chothia, and A. Guha. Statistical measurement of information leakage. In *Proc. of TACAS*, volume 6015 of *LNCS*, pages 390–404. Springer, 2010.
11. K. Chatzikokolakis, C. Palamidessi, and P. Panangaden. On the Bayes risk in information-hiding protocols. *J. of Comp. Security*, 16(5):531–571, 2008.
12. D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.
13. T. Chothia, Y. Kawamoto, and C. Novakovic. LeakWatch: Estimating information leakage from java programs. In *Proc. of ESORICS 2014 Part II*, pages 219–236, 2014.
14. T. Chothia, Y. Kawamoto, C. Novakovic, and D. Parker. Probabilistic point-to-point information leakage. In *Proc. of CSF*, pages 193–205. IEEE Computer Society, 2013.

15. D. Clark, S. Hunt, and P. Malacaria. Quantitative information flow, relations and polymorphic types. *J. of Logic and Computation*, 18(2):181–199, 2005.
16. D. Clark, S. Hunt, and P. Malacaria. A static analysis for quantifying information flow in a simple imperative language. *J. of Comp. Security*, 2007.
17. B. Espinoza and G. Smith. Min-entropy as a resource. *Inf. and Comp.*, 226:57–75, 2013.
18. H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.
19. G. Helali, O. Hasan, and S. Tahar. *Formal Analysis of Information Flow Using Min-Entropy and Belief Min-Entropy.* Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
20. J. Heusser and P. Malacaria. Quantifying information leaks in software. In *Proc. of ACSAC*, pages 261–269. ACM, 2010.
21. J. Hölzl and T. Nipkow. Interactive verification of markov chains: Two distributed protocol case studies. 103, 12 2012.
22. B. Köpf and D. A. Basin. An information-theoretic model for adaptive side-channel attacks. In *Proc. of CCS*, pages 286–296. ACM, 2007.
23. B. Köpf, L. Mauborgne, and M. Ochoa. Automatic quantification of cache side-channels. In *Computer Aided Verification - 24th Int. Conf., CAV 2012, Berkeley, CA, USA, July 7-13, 2012 Proceedings*, volume 7358 of *LNCS*, pages 564–580. Springer, 2012.
24. B. Köpf and A. Rybalchenko. Approximation and randomization for quantitative information-flow analysis. In *Proc. of CSF*, pages 3–14. IEEE, 2010.
25. Massey. Guessing and entropy. In *Proceedings of the IEEE Int. Symposium on Information Theory*, page 204. IEEE, 1994.
26. S. McCamant and M. D. Ernst. Quantitative information flow as network flow capacity. In *Proc. of SIGPLAN*, pages 193–205, Tucson, AZ, USA, June 9–11, 2008.
27. A. McIver, L. Meinicke, and C. Morgan. Compositional closure for bayes risk in probabilistic noninterference. In *Proc. of ICALP*, volume 6199 of *LNCS*, pages 223–235. Springer, 2010.
28. A. McIver, C. Morgan, and T. M. Rabehaja. Abstract hidden markov models: A monadic account of quantitative information flow. In *Proc. of LICS*, pages 597–608, 2015.
29. C. Morgan. A Haskell program to compute hyperdistributions for measuring information leakage.
30. C. Novakovic. *Computing and Estimating Information Leakage with a Quantitative Point-to-Point Information Flow Model.* PhD thesis, 2014. PhD thesis, Birmingham University, UK.
31. D. Parker. *Implementation of Symbolic Model Checking for Probabilistic Systems.* PhD thesis, University of Birmingham, 2002.
32. Q. Phan, P. Malacaria, C. S. Pasareanu, and M. d'Amorim. Quantifying information leaks using reliability analysis. In *Proc. of SPIN*, pages 105–108, 2014.
33. M. K. Reiter and A. D. Rubin. Crowds: anonymity for Web transactions. *ACM Trans. on Information and System Security*, 1(1):66–92, 1998.
34. C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 625–56, 1948.
35. G. Smith. On the foundations of quantitative information flow. In *Proc. of FOSSACS*, volume 5504 of *LNCS*, pages 288–302. Springer, 2009.
36. H. Yasuoka and T. Terauchi. Quantitative information flow as safety and liveness hyperproperties. *Theor. Comp. Sci.*, 538:167–182, 2014.