Spécification et vérification des protocoles de sécurité probabilistes

Konstantinos Chatzikokolakis

Probabilistic and Information-Theoretic Approaches to Anonymity

Konstantinos Chatzikokolakis

October 2007

Contents

Contents i

1 Introduction 1
- 1.1 The probabilistic dimension . . . 2
- 1.2 Information theory . . . 4
- 1.3 Hypothesis testing . . . 4
- 1.4 Interplay between nondeterminism and probabilities . . . 5
- 1.5 Plan of the thesis - Contributions . . . 5
- 1.6 Publications . . . 7
- 1.7 Acknowledgments . . . 8

2 Preliminaries 9
- 2.1 Probability spaces . . . 9
- 2.2 Information theory . . . 10
- 2.3 Convexity . . . 11
- 2.4 Simple probabilistic automata . . . 12
- 2.5 CCS with internal probabilistic choice . . . 14

3 Anonymity Systems 17
- 3.1 Anonymity properties . . . 17
- 3.2 Anonymity protocols . . . 18
  - 3.2.1 Dining Cryptographers . . . 18
  - 3.2.2 Crowds . . . 20
  - 3.2.3 Other protocols . . . 22

I Probabilistic Approach 25

4 A probabilistic framework to model anonymity protocols 27
- 4.1 Formal definition of an anonymity system . . . 28
  - 4.1.1 Finite anonymity systems . . . 31
- 4.2 Protocol composition . . . 32
- 4.3 Example: modeling a system using probabilistic automata . . . 32

5 Strong Anonymity 35
- 5.1 Formal definition . . . 36
- 5.2 Strong anonymity of the dining cryptographers protocol . . . 37
- 5.3 Protocol composition . . . 39

6 Probable Innocence **41**
6.1 Existing definitions of probable innocence 43
6.1.1 First approach (limit on the probability of detection) . . 44
6.1.2 Second approach (limit on the attacker's confidence) . . 46
6.2 A new definition of probable innocence 47
6.3 Relation to other definitions 51
6.3.1 Definition by Reiter and Rubin 51
6.3.2 Definition of Halpern and O'Neill 52
6.3.3 Strong anonymity 52
6.4 Protocol composition 53
6.5 Application to anonymity protocols 55
6.5.1 Crowds 55
6.5.2 Dining cryptographers 57

II Information Theory and Hypothesis Testing **63**

7 An information-theoretic definition of anonymity **65**
7.1 Loss of Anonymity as Channel Capacity 67
7.1.1 Relative Anonymity 68
7.2 Computing the channel's capacity 72
7.3 Relation with existing anonymity notions 74
7.3.1 Capacity 0: strong anonymity 74
7.3.2 Conditional capacity 0: strong anonymity "within a group" 74
7.3.3 Probable innocence: weaker bounds on capacity 75
7.4 Adding edges to a dining cryptographers network 79
7.5 Computing the degree of anonymity of a protocol 84
7.5.1 Dining cryptographers 84
7.5.2 Crowds 86
7.6 Related work 88

8 A monotonicity principle **91**
8.1 The monotonicity principle 92
8.2 Binary channels 94
8.3 Relations between channels 97
8.3.1 Algebraic information theory 97
8.3.2 A new partial order on binary channels 98
8.3.3 The coincidence of algebra, order and geometry 100
8.4 Relations between monotone mappings on channels 102
8.4.1 Algebraic relations 102
8.4.2 Inequalities 105

9 Hypothesis testing and the probability of error **109**
9.1 Hypothesis testing and the probability of error 111
9.2 Convexly generated functions and their bounds 113
9.2.1 An alternative proof for the Hellman-Raviv and Santhi-Vardy bounds 116
9.3 The corner points of the Bayes risk 117
9.3.1 An alternative characterization of the corner points . . . 122
9.3.2 Examples 126

9.4 Application: Crowds . . . 129
9.4.1 Crowds in a clique network . . . 130
9.4.2 Crowds in a grid network . . . 132
9.5 Protocol composition . . . 135
9.5.1 Independence from the input distribution . . . 136
9.5.2 Bounds on the probability of error . . . 137
9.6 Related work . . . 140

III Adding Nondeterminism **141**

10 The problem of the scheduler **143**
10.1 A variant of CCS with explicit scheduler . . . 146
10.1.1 Syntax . . . 146
10.1.2 Semantics . . . 147
10.1.3 Deterministic labelings . . . 149
10.2 Expressiveness of the syntactic scheduler . . . 151
10.2.1 Using non-linear labelings . . . 153
10.3 Testing relations for CCS_σ processes . . . 154
10.4 An application to security . . . 162
10.4.1 Encoding secret value passing . . . 162
10.4.2 Dining cryptographers with probabilistic master . . . 162
10.4.3 Dining cryptographers with nondeterministic master . . . 165
10.5 Related work . . . 166

11 Analysis of a contract-signing protocol **169**
11.1 Syntactic extensions of CCS_σ . . . 170
11.1.1 Creating and splitting tuples . . . 170
11.1.2 Polyadic value passing . . . 170
11.1.3 Matching . . . 171
11.1.4 Using extended syntax in contexts . . . 171
11.2 Probabilistic Security Protocols . . . 172
11.2.1 1-out-of-2 Oblivious Transfer . . . 172
11.2.2 Partial Secrets Exchange Protocol . . . 173
11.3 Verification of Security Properties . . . 175
11.3.1 A specification for PSE . . . 176
11.3.2 Proving the correctness of PSE . . . 177
11.4 Related Work . . . 180

Bibliography **183**

One

Introduction

> Qu'on me donne six lignes écrites de la main du plus honnête homme, j'y trouverai de quoi le faire pendre.[1]
>
> *Armand Jean du Plessis, Cardinal de Richelieu*

The concept of *anonymity* comes into play in those cases in which we want to keep secret the identity of the agents participating to a certain event. There is a wide range of situations in which this property may be needed or desirable; for instance: voting, web surfing, anonymous donations, and posting on bulletin boards.

Anonymity is often formulated in a more general way as an *information-hiding* property, namely the property that a part of information relative to a certain event is kept secret. One should be careful, though, not to confuse anonymity with other properties that fit the same description, notably *confidentiality* (aka *secrecy*). Let us emphasize the difference between the two concepts with respect to sending messages: confidentiality refers to situations in which the content of the message is to be kept secret; in the case of anonymity, on the other hand, it is the identity of the originator, or of the recipient, that has to be kept secret. Analogously, in voting, anonymity means that the identity of the voter associated with each vote must be hidden, and not the vote itself or the candidate voted for. Other notable properties in this class are *privacy* and *non-interference*. Privacy refers to the protection of certain data, such as the credit card number of a user. Non-interference means that a "low" user will not be able to acquire information about the activities of a "high" user. A discussion about the difference between anonymity and other information-hiding properties can be found in [HO03, HO05].

An important characteristic of anonymity is that it is usually relative to the capabilities of the observer. In general the activity of a protocol can be observed by diverse range of observers, differing in the information they have

[1] If one would give me six lines written by the hand of the most honest man, I would find something in them to have him hanged.

access to. The anonymity property depends critically on what we consider as observables. For example, in the case of an anonymous bulletin board, a posting by one member of the group is kept anonymous to the other members; however, it may be possible that the administrator of the board has access to some privileged information that may allow him to infer the identity of the member who posted it.

In general anonymity may be required for a subset of the agents only. In order to completely define anonymity for a protocol it is therefore necessary to specify which set(s) of members have to be kept anonymous. A further generalization is the concept of *anonymity with respect to a group*: the members are divided into a number of sets, and we are allowed to reveal to which group the user responsible for the action belongs, but not the identity of the user himself.

Various formal definitions and frameworks for analyzing anonymity have been developed in literature. They can be classified into approaches based on process-calculi ([SS96, RS01]), epistemic logic ([SS99, HO03]), and "function views" ([HS04]). Most of these approaches are based on the so-called "principle of confusion": a system is anonymous if the set of possible observable outcomes is saturated with respect to the intended anonymous users. More precisely, if in one computation the *culprit* (the user who performs the action) is i and the observable outcome is o, then for every other agent j there must be a computation where j is the culprit and the observable is still o. This approach is also called *possibilistic*, and relies on *nondeterminism*. In particular, probabilistic choices are interpreted as nondeterministic. We refer to [RS01] for more details about the relation of this approach to the notion of anonymity.

1.1 The probabilistic dimension

The possibilistic approach to anonymity, described in previous section, is elegant and general, however it is limited in that it does not cope with quantitative information. Now, several anonymity protocols use randomized primitives to achieve the intended security properties. This is the case, for instance, of the Dining Cryptographers [Cha88], Crowds [RR98], Onion Routing [SGR97], and Freenet [CSWH00]. Furthermore, attackers may use statistical analyses to try to infer the secret information from the observables. This is a common scenario for a large class of security problems.

Another advantage of taking probabilistic information into account is that it allows to classify various notions of anonymity according to their strength. The possibilistic approaches to information hiding are rather coarse in this respect, in the sense that they do not distinguish between the various levels of leakage of probabilistic information. For instance, the notion of anonymity that Reiter and Rubin call "possible innocence" [RR98] is satisfied whenever the adversary cannot be absolutely certain of the identity of the culprit. This is the weakest notion of anonymity. So, the possibilistic approach distinguishes between the total lack of anonymity and "some" anonymity, but considers equivalent all protocols that provide anonymity to some extent, from the least to the maximum degree.

A very good example that demonstrates the need for a probabilistic analysis of voting protocols is due to Di Cosmo ([DC07]). In this article, an old attacking technique used in Italy twenty years ago is demonstrated, and it is shown that

protocols today still fail to cope with this simple attack. We briefly describe it here: in the voting system used in Italy during the 70's and 80's, voters were using the following voting procedure. They first had to choose a party. Then, they could state their preferences by selecting a limited number of candidates out of a long list proposed by the party, and writing them in the ballot in any desirable order. Then a complex algorithm was used to determine the winner of which the relevant part is that the party with more votes would have more seats and, among the candidates of the same party, the one with the most preferences would have more chances to get a seat.

Then the technique to break this system works as follows. The local boss makes a visit to a sizable number of voters susceptible not to vote for his party, accompanied by a couple of well built bodyguards. The boss gives to each voter a specific sequence of candidates, in which he himself appears in the top position, and asks the voter to vote for his party and mark this exact sequence in the ballot. Given that the total number of candidates is big and voters can state up to four preferences, there are enough combinations for the boss to give a distinct sequence to each individual voter. Then the boss tells the voter that if this specific sequence that was given to him doesn't show up during the counting of the ballots (a procedure which is of course performed publicly) then a new visit will be made, an event quite unfortunate for the voter.

If the voter doesn't comply, then there is still a chance that the voter will escape the second visit, if it happens that someone else votes for the exact sequence that was given by the boss. However, the probability of this to happen is very low so the technique was quite effective for two decades, until the fraud was revealed and the number of preferences was reduced to only one to avoid this attack.

What is even more interesting, as shown in [DC07], is that even today, voting protocols such as the Three Ballot protocol ([Riv06]) are vulnerable to the same attack due to the high number of choices that are available to the voter on the same ballot. Moreover, many anonymity definitions, like the one proposed in [DKR06], fail to detect this problem and are satisfied by protocols vulnerable to it. This example clearly demonstrates that, in order to cope with subtle attacks like the one presented, we need a finer analysis involving probabilistic models and techniques.

A probabilistic notion of anonymity was developed (as a part of a general epistemological approach) in [HO03]. The approach there is purely probabilistic, in the sense that both the protocol and the users are assumed to act probabilistically. In particular the emphasis is on the probability of the users being the culprit.

In this thesis we take the opposite point of view, namely we assume that we may know nothing about the users and that the definition of anonymity should not depend on the probabilities of the users performing the action of interest. We consider this a fundamental property of a good notion of anonymity. In fact, a protocol for anonymity should be able to guarantee this property for every group of users, no matter what is their probability distribution of being the culprit.

1.2 Information theory

Recently it has been observed that at an abstract level information-hiding protocols can be viewed as *channels* in the information-theoretic sense. A channel consists of a set of input values $\mathcal{A}$, a set of output values $\mathcal{O}$ and a transition matrix which gives the conditional probability $p(o|a)$ of producing o in the output when a is the input. In the case of privacy preserving protocols, $\mathcal{A}$ contains the information that we want to hide and $\mathcal{O}$ the facts that the attacker can observe. This framework allows us to apply concepts from information theory to reason about the knowledge that the attacker can gain about the input by observing the output of the protocol.

In the field of information flow and non-interference there have been various works [McL90, Gra91, CHM01, CHM05, Low02] in which the *high information* and the *low information* are seen as the input and output respectively of a (noisy) channel. Non-interference is formalized in this setting as the converse of channel capacity.

Channel capacity has been also used in relation to anonymity in [MNCM03, MNS03]. These works propose a method to create covert communication by means of non-perfect anonymity.

A related line of work is [SD02, DSCP02], where the main idea is to express the lack of (probabilistic) information in terms of entropy.

1.3 Hypothesis testing

In information-hiding systems the attacker finds himself in the following scenario: he cannot directly detect the information of interest, namely the actual value of the random variable $A \in \mathcal{A}$, but he can discover the value of another random variable $O \in \mathcal{O}$ which depends on A according to a known conditional distribution. This kind of situation is quite common also in other disciplines, like medicine, biology, and experimental physics, to mention a few. The attempt to infer A from O is called *hypothesis testing* (the "hypothesis" to be validated is the actual value of A), and it has been widely investigated in statistics. One of the most used approaches to this problem is the Bayesian method, which consists of assuming known the *a priori* probability distribution of the hypotheses, and deriving from that (and from the matrix of the conditional probabilities) the a posteriori distribution after a certain fact has been observed. It is well known that the best strategy for the adversary is to apply the MAP (Maximum Aposteriori Probability) criterion, which, as the name says, dictates that one should choose the hypothesis with the maximum a posteriori probability for the given observation. "Best" means that this strategy induces the smallest probability of error in the guess of the hypothesis. The probability of error, in this case, is also called *Bayes risk*.

A major problem with the Bayesian method is that the *a priori* distribution is not always known. This is particularly true in security applications. In some cases, it may be possible to approximate the *a priori* distribution by statistical inference, but in most cases, especially when the input information changes over time, it may not (see Section 1.4 for more discussion on this point). Thus other methods need to be considered, which do not depend on the *a priori* distribution. One such method is the one based on the so-called *Maximum*

Likelihood criterion.

1.4 Interplay between nondeterminism and probabilities

We have already argued that the purely possibilistic approach, in the case of probabilistic protocols, is too coarse and therefore not very useful. Here we want to point out that in many cases the purely probabilistic approach is not very suitable either, and that it is better to consider a setting in which both aspects (probabilities and nondeterminism) are present. There are, indeed, two possible sources of nondeterminism:

(1) The users of the protocol, who may be totally unpredictable and even change over time, so that their choices cannot be quantified probabilistically, not even by repeating statistical observations[1].

(2) The protocol itself, which can behave nondeterministically in part, due, for instance, to the interleaving of the parallel components. In the following we will refer to the "scheduler" as an entity that determines the interleaving.

The case (2) has some subtle implications, related to the fact that the traditional notion of scheduler may reveal the outcome of the protocol's random choices, and therefore the model of the adversary is too strong even for obviously correct protocols. In this case we would like to limit the power of the scheduler and make him oblivious to this sensitive information. This issue is one of the hot topics in security, it was for instance one of the main subject of discussion at the panel of CSFW 2006.

1.5 Plan of the thesis - Contributions

The thesis is organized into three parts. In Part I a probabilistic framework to model anonymity protocols is introduced. We use the framework to model two basic anonymity properties, strong anonymity and probable innocence. In Part II we focus on information theory and hypothesis testing. We model protocols as noisy channels and use the notion of capacity to measure their degree of anonymity. A general monotonicity principle for channels is developed and its implications for binary channels are explored. In the case of hypothesis testing a technique to obtain bounds of piecewise linear functions by considering a finite set of points is developed and used in the case of the probability of error. Finally, Part III deals with nondeterminism and the problem that arises if the outcome of probabilistic choices is visible to the scheduler.

Apart from these three parts there are three introductory chapters, the first being the present introduction. Chapter 2 introduces some preliminary notions used throughout the thesis. Chapter 3 provides an introduction to anonymity

[1]Some people consider nondeterministic choice as a probabilistic choice with unknown probabilities. Our opinion is that the two concepts are different: the notion of probability implies that we can gain knowledge of the distribution by repeating the experiment under the same conditions and by observing the frequency of the outcomes. In other words, from the past we can predict the future. This prediction element is absent from the notion of nondeterminism.

systems. A discussion of anonymity properties is made and two anonymity protocols, serving as running examples throughout the thesis, are presented.

We now summarize each one of the three main chapters in greater detail.

Part I - Probabilistic approach

In Chapter 4 we describe the general probabilistic framework that is used to model anonymity protocols and we give the definition of an *anonymity system* and an *anonymity instance*. This framework is used in all subsequent chapters. In Chapter 5 we give a definition of strong anonymity and we show that the Dining Cryptographers protocol satisfies it under the assumption of fair coins. The case of protocol repetition is also considered, showing that if a protocol is strongly anonymous then any repetition of it is also strongly anonymous.

Chapter 6 contains most of the results of the first part. We examine two formal definitions of probable innocence and show cases in which they do not express the intuition behind this anonymity notion. We then combine the two definitions into a new one that is equivalent to them under certain conditions but that overcomes their shortcomings in the general case. Using the new definition is it shown that a repetition of a protocol unboundedly many times satisfies strong anonymity if and only if the protocol is strongly anonymous. The new definition is also applied to Dining Cryptographers, obtaining sufficient and necessary conditions on various kinds of network graphs, and to Crowds giving an alternative proof for its conditions for probable innocence.

Part II - Information theory and hypothesis testing

This part is the largest of the three in terms of material and new results. In Chapter 7 a quantitative measure of anonymity is proposed, based on the concept of capacity, and an extended notion of capacity is developed to deal with situations where some information is leaked by design. A compositionality result is shown for the latter case, and also a method to compute the capacity in the presence of certain symmetries. Then the relation of this measure with existing anonymity properties is examined, in particular with the ones of Part I. Applying the new measure to the Dining Cryptographers protocol we show that the anonymity always improves when we add an edge to any network graph. This result also allows us to give sufficient and necessary conditions for strong anonymity. Finally a model-checking approach is demonstrated in both the Dining Cryptographers and Crowds, calculating their degree of anonymity while varying some parameters of the protocol.

In Chapter 8 we focus on channels and we develop a monotonicity principle for capacity, based on its convexity as a function of the channel matrix. We then use this principle to show a number of results for binary channels. First we develop a new partial order for algebraic information theory with respect to which capacity is monotone. This order is much bigger than the interval inclusion order and can be characterized in three different ways: with a simple formula, geometrically and algebraically. Then we establish bounds on the capacity based on easily computable functions. We also study its behavior along lines of constant capacity leading to graphical methods for reasoning about capacity that allow us to compare channels in "most" cases.

In Chapter 9 we consider the probability of error in the case of hypothesis testing using the maximum a posteriori probability rule. We first show

how to obtain bounds for functions that are convexly generated by a subset of their points. We use this result to give a simple alternative proof of two known bounds for the probability of error from the literature. Then we show that the probability of error is convexly generated by a finite set of points, depending only on the matrix of the channel, and we give a characterization of these points. We use this result to improve the previous bounds and obtain new ones that are tight in at least one point. This technique is demonstrated in an instance of the Crowds protocol using model-checking methods. Finally we consider hypothesis testing in the case of protocol repetition using the maximum likelihood rule, showing that given enough observations this rule can simulate the MAP rule, and providing bounds for the probability of error in various cases.

Part III - Adding nondeterminism

In Chapter 10 we consider a problem that arises in the analysis of probabilistic security protocols in the presence of nondeterminism. Namely, if the scheduler is unrestricted then it could reveal the outcome of probabilistic choices by basing its decisions on them. We develop a solution to this problem in terms of a probabilistic extension of CCS with a syntactic scheduler. The scheduler uses labels to guide the execution of the process. We show that using pairwise distinct labels the syntactic scheduler has all the power of the semantic one. However, by using multiple copies of the same label we can effectively limit the power of the scheduler and make it oblivious to certain probabilistic choices. We also study testing preorders for this calculus and show that they are precongruences wrt all operators except + and that, using a proper labeling, probabilistic choice distributes over all operators except !. Finally we apply the new calculus to the dining cryptographers problem in the case that the order of the announcements is chosen nondeterministically. We show that in this case the protocol is strongly anonymous if the decision of the master and the outcome of the coins are invisible to the scheduler. We also study a variant of the protocol with a nondeterministic master.

In Chapter 11 we study a probabilistic contract-signing protocol, namely the Partial Secrets Exchange protocol. We model the protocol in the calculus of Chapter 11 and we also create a specification expressing its correct behavior. We prove the correctness of the protocol by showing that it is related to the specification under the may-testing preorder. The proof of this result uses the distributivity of the probabilistic sum in the calculus of Chapter 11, showing its use for verification.

1.6 Publications

Many of the results in this thesis have been published in journals or in the proceedings of conferences or workshops. More specifically, the results of Chapter 6 appeared in [CP06a] and an extended version was published in [CP06b]. Some of the results in Chapter 7 appeared in [CPP06] and an extended version was published in [CPP07a]. The results of Chapter 8 are in preparation for publication ([CM07]). The results of Chapter 9 appeared in [CPP07b], an extended journal version is under preparation. The results of Chapter 10 appeared in [CP07]. The results of Chapter 11 appeared in [CP05a] and an extended ver-

sion was published in [CP05b]. Finally, some of the material of Chapter 3 appeared in [CC05].

1.7 Acknowledgments

It is hard to express my gratitude to my coauthors for their involvement in our joint papers. Prakash Panangaden, through numerous discussions during my visits in Montreal, in Paris as well as in more exotic Caribbean places, inspired and contributed to the results of Chapters 7 and 9. Moreover, his enthusiasm in answering technical questions makes him a keen teacher and is greatly appreciated. I'm also particularly grateful to Keye Martin for his hard work on our joint paper during the last month, while I was fulltime occupied in the writing of this thesis. Chapter 8 would not be made possible without his help. Last, but not least, Tom Chothia contributed heavily to our joint survey paper from which much of the material of Chapter 3 is taken. Moreover, through numerous discussions during his stay at LIX, he was a constant source of information and inspiration on anonymity related topics.

Two

Preliminaries

In this chapter we give a brief overview of the technical concepts from literature that will be used through the thesis.

2.1 Probability spaces

We recall here some basic notion of Probability Theory.

Let Ω be a set. A *σ-field* over Ω is a collection $\mathcal{F}$ of subsets of Ω closed under complement and countable union and such that $\Omega \in \mathcal{F}$. If $\mathcal{F}$ is only closed under finite union then it is a *field* over Ω. If U is a collection of subsets of Ω then the σ-field *generated by* U is defined as the intersection of all σ-fields containing U (note that there is at least one since the powerset of Ω is a σ-field containing U).

A *measure* on $\mathcal{F}$ is a function $\mu : \mathcal{F} \mapsto [0, \infty]$ such that

1. $\mu(\emptyset) = 0$ and
2. $\mu(\bigcup_i C_i) = \sum_i \mu(C_i)$ where C_i is a countable collection of pairwise disjoint elements of $\mathcal{F}$.

A *probability measure* on $\mathcal{F}$ is a measure μ on $\mathcal{F}$ such that $\mu(\Omega) = 1$. A *probability space* is a tuple $(\Omega, \mathcal{F}, \mu)$ where Ω is a set, called the *sample space*, $\mathcal{F}$ is a σ-field on Ω and μ is a probability measure on $\mathcal{F}$.

A probability space and the corresponding probability measure are called *discrete* if $\mathcal{F} = 2^\Omega$ and

$$\mu(C) = \sum_{x \in C} \mu(\{x\}) \quad \forall C \in \mathcal{F}$$

In this case, we can construct μ from a function $p : \Omega \mapsto [0, 1]$ satisfying $\sum_{x \in \Omega} p(x) = 1$ by assigning $\mu(\{x\}) = p(x)$. The function p is called a *probability distribution* over Ω.

The set of all discrete probability measures with sample space Ω will be denoted by $Disc(\Omega)$. We will also denote by $\delta(x)$ (called the *Dirac measure* on x) the probability measure s.t. $\mu(\{x\}) = 1$.

The elements of a σ-field $\mathcal{F}$ are also called *events*. If A, B are events then $A \cap B$ is also an event. If $\mu(A) > 0$ then we can define the *conditional probability*

$p(B|A)$, meaning "the probability of B given that A holds", as

$$p(B|A) = \frac{\mu(A \cap B)}{\mu(A)}$$

Note that $p(\cdot|A)$ is a new probability measure on $\mathcal{F}$. In continuous probability spaces, where many events have zero probability, it is possible to generalize the concept of conditional probability to allow conditioning on such events. However, this is not necessary for the needs of this thesis. Thus we will use the "traditional" definition of conditional probability and make sure that we never condition on events of zero probability.

Let $\mathcal{F}, \mathcal{F}'$ be two σ-fields on Ω, Ω' respectively. A *random variable* X is a function $X : \Omega \mapsto \Omega'$ that is *measurable*, meaning that the inverse of every element of $\mathcal{F}'$ belongs to $\mathcal{F}$:

$$X^{-1}(C) \in \mathcal{F} \quad \forall C \in \mathcal{F}'$$

Then given a probability measure μ on $\mathcal{F}$, X induces a probability measure μ' on $\mathcal{F}'$ as

$$\mu'(C) = \mu(X^{-1}(C)) \quad \forall C \in \mathcal{F}'$$

If μ' is a discrete probability measure then it can be constructed by a probability distribution over Ω', called *probability mass function (pmf)*, defined as $P([X = x]) = \mu(X^{-1}(x))$ for each $x \in \Omega'$. The random variable in this case is called discrete. If X, Y are discrete random variables then we can define a discrete random valuer (X, Y) by its pmf $P([X = x, Y = y]) = \mu(X^{-1}(x) \cap X^{-1}(y))$. If X is a real-valued discrete random variable then its *expected value* (or *expectation*) is defined as

$$EX = \sum_i x_i \, P([X = x_i])$$

Notation: We will use capital letters X, Y to denote random variables and calligraphic letters $\mathcal{X}, \mathcal{Y}$ to denote their image. With a slight abuse of notation we will use p (and $p(x), p(y)$) to denote either

- a probability distribution, when $x, y \in \Omega$, or
- a probability measure, when $x, y \in \mathcal{F}$ are events, or
- the probability mass function $P([X = x]), P([Y = y])$ of the random variables X, Y respectively, when $x \in \mathcal{X}, y \in \mathcal{Y}$.

2.2 Information theory

Information theory reasons about the uncertainty of a random variable and the information that it can reveal about another random variable. In this section we recall the notions of *entropy*, *mutual information* and *channel capacity*, we refer to [CT91] for more details. We consider only the discrete case since it is enough for the scope of this thesis.

Let X be a discrete random variable with image $\mathcal{X}$ and pmf $p(x) = P([X = x])$ for $x \in \mathcal{X}$. The *entropy* $H(X)$ of X is defined as

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x)$$

The entropy measures the uncertainty of a random variable. It takes its maximum value $\log |\mathcal{X}|$ when X's distribution is uniform and its minimum value 0 when X is constant. We usually take the logarithm with base 2 and measure entropy in *bits*. Roughly speaking, m bits of entropy means that we have 2^m values to choose from, assuming a uniform distribution.

The *relative entropy* or *Kullback–Leibler distance* between two probability distributions p, q on the same set $\mathcal{X}$ is defined as $D(p \parallel q) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)}$. It is possible to show that $D(p \parallel q)$ is always non-negative, and it is 0 if and only if $p = q$.

Now let X, Y be random variables. The *conditional entropy* $H(X|Y)$ is

$$H(X|Y) = -\sum_{y \in \mathcal{Y}} p(y) \sum_{x \in \mathcal{X}} p(x|y) \log p(x|y)$$

Conditional entropy measures the amount of uncertainty of X when Y is known. It can be shown that $0 \leq H(X|Y) \leq H(X)$. It takes its maximum value $H(X)$ when Y reveals no information about X, and its minimum value 0 when Y completely determines the value of X.

Comparing $H(X)$ and $H(X|Y)$ gives us the concept of *mutual information* $I(X;Y)$, which is defined as

$$I(X;Y) = H(X) - H(X|Y)$$

or equivalently

$$I(X;Y) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \tag{2.1}$$

Mutual information measures the amount of information that one random variable contains about another random variable. In other words, it measures the amount of uncertainty about X that we lose when observing Y. It can be shown that it is symmetric ($I(X;Y) = I(Y;X)$) and that $0 \leq I(X;Y) \leq H(X)$.

A *communication channel* is a tuple $(\mathcal{X}, \mathcal{Y}, p_c)$ where $\mathcal{X}, \mathcal{Y}$ are the sets of input and output symbols respectively and $p_c(y|x)$ is the probability of observing output $y \in \mathcal{Y}$ when $x \in \mathcal{X}$ is the input. Given an input distribution $p(x)$ over $\mathcal{X}$ we can define the random variables X, Y for input and output respectively. The maximum mutual information between X and Y over all possible distributions $p(x)$ is known as the channel's *capacity*.

$$C = \max_{p(x)} I(X;Y)$$

The capacity of a channel gives the maximum rate at which information can be transmitted using this channel.

2.3 Convexity

Let $\mathbb{R}$ be the set of real numbers. The elements $\lambda_1, \lambda_2, \ldots, \lambda_k \in \mathbb{R}$ constitute a set of *convex coefficients* iff $\forall i\ \lambda_i \geq 0$ and $\sum_i \lambda_i = 1$.

Let V be a vector space over $\mathbb{R}$. A *convex combination* of $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_k \in V$ is a vector of the form

$$\vec{x} = \sum_i \lambda_i \vec{x}_i$$

where the λ_i's are convex coefficients. A subset S of V is *convex* iff every convex combination of vectors in S is also in S. Given a subset S of V, the *convex hull* of S, denoted by $ch(S)$, is the smallest convex set containing S. Since the intersection of convex sets is convex, it is clear that $ch(S)$ always exists.

A function $f : S \to \mathbb{R}$ defined on a convex set S is *convex* iff

$$f(\sum_i \lambda_i \vec{x}_i) \leq \sum_i \lambda_i f(\vec{x}_i) \qquad \forall x_1, \ldots, x_k \in S$$

where the λ_i's are convex coefficients. A function is *strictly convex* if, assuming pairwise distinct $\vec{x}_i$'s, equality (in the above inequality) holds iff $\lambda_i = 1$ for some i. A function f is *(strictly) concave* if $-f$ is (strictly) convex. If X is a real-valued random variable and f is convex, then Jensen's inequality states that

$$Ef(X) \leq f(EX)$$

where E denotes the expected value. If f is concave then the inequality is reversed.

2.4 Simple probabilistic automata

We recall here some basic notions about probabilistic automata, following the settings of [Seg95].

A *simple probabilistic automaton*[1] is a tuple $(S, q, A, \mathcal{D})$ where S is a set of states, $q \in S$ is the *initial state*, A is a set of actions and $\mathcal{D} \subseteq S \times A \times Disc(S)$ is a *transition relation*. Intuitively, if $(s, a, \mu) \in \mathcal{D}$ then there is a transition from the state s performing the action a and leading to a distribution μ over the states of the automaton. We also write $s \xrightarrow{a} \mu$ if $(s, a, \mu) \in \mathcal{D}$. The idea is that the choice of transition among the available ones in $\mathcal{D}$ is performed nondeterministically, and the choice of the target state among the ones allowed by μ (i.e. those states q such that $\mu(q) > 0$) is performed probabilistically. A probabilistic automaton M is *fully probabilistic* if from each state of M there is at most one transition available.

An execution fragment α of a probabilistic automaton is a (possibly infinite) sequence $s_0 a_1 s_1 a_2 s_2 \ldots$ of alternating states and actions, such that for each i there is a transition $(s_i, a_{i+1}, \mu_i) \in \mathcal{D}$ and $\mu_i(s_{i+1}) > 0$. The concatenation of a finite execution fragment $\alpha_1 = s_0 \ldots a_n s_n$ and an execution fragment $\alpha_2 = s_n a_{n+1} s_{n+1} \ldots$ is the execution fragment $\alpha_1 \cdot \alpha_2 = s_0 \ldots a_n s_n a_{n+1} s_{n+1} \ldots$. A finite execution fragment α_1 is a *prefix* of α, written $\alpha_1 \leq \alpha$, if there is an execution fragment α_2 such that $\alpha = \alpha_1 \cdot \alpha_2$. We will use $fstate(\alpha), lstate(\alpha)$ to denote the first and last state of a finite execution fragment α respectively. An execution is an execution fragment such that $fstate(\alpha) = q$. An execution

[1] For simplicity in the following we will refer to a simple probabilistic automaton as *probabilistic automaton*. Note however that simple probabilistic automata are a subset of the probabilistic automata defined in [Seg95, SL95].

α is *maximal* if it is infinite or there is no transition from $lstate(\alpha)$ in $\mathcal{D}$. We denote by $exec^*(M)$ and $exec(M)$ the sets of all the finite and of all the executions of M respectively.

A *scheduler* of a probabilistic automaton $M = (S, q, A, \mathcal{D})$ is a function

$$\zeta : exec^*(M) \to \mathcal{D}$$

such that $\zeta(\alpha) = (s, a, \mu) \in \mathcal{D}$ implies that $s = lstate(\alpha)$. The idea is that a scheduler selects a transition among the ones available in $\mathcal{D}$ and it can base its decision on the history of the execution.

The *execution tree* of M relative to the scheduler ζ, denoted by $etree(M, \zeta)$, is a fully probabilistic automaton $M' = (S', q', A', \mathcal{D}')$ such that $S' \subseteq exec(M)$, $q' = q$, $A' = A$, and $(\alpha, a, \mu') \in \mathcal{D}'$ if and only if $\zeta(\alpha) = (lstate(\alpha), a, \mu)$ for some μ and $\mu'(\alpha a s) = \mu(s)$. Intuitively, $etree(M, \zeta)$ is produced by unfolding the executions of M and resolving all nondeterministic choices using ζ. Note that $etree(M, \zeta)$ is a simple[2] and fully probabilistic automaton.

Given a fully probabilistic automaton $M = (S, q, A, \mathcal{D})$ we can define a probability space $(\Omega_M, \mathcal{F}_M, P_M)$ on the space of executions of M as follows:

- $\Omega_M \subseteq exec(M)$ is the set of maximal executions of M.
- If α is a finite execution of M we define the cone with prefix α as $C_\alpha = \{\alpha' \in \Omega_M | \alpha \leq \alpha'\}$. Let $\mathcal{C}_M$ be the collection of all cones of M. Then $\mathcal{F}$ is the σ-field generated by $\mathcal{C}_M$ (by closing under complement and countable union).
- We define the probability of a cone C_α where $\alpha = s_0 a_1 s_1 \ldots a_n s_n$ as

$$P(C_\alpha) = \prod_{i=1}^{n} \mu_i(s_i)$$

  where μ_i is the (unique because the automaton is fully probabilistic) measure such that $(s_{i-1}, a_i, \mu_i) \in \mathcal{D}$. We define P_M as the measure extending P to $\mathcal{F}$ (see [Seg95] for more details about this construction).

Now we define the probability space $(\Omega_T, \mathcal{F}_T, P_T)$ on the traces of a fully probabilistic automaton M. Let $ext(M) \subseteq A$ be the set of external actions of M. We define $\Omega_T = ext(M)^* \cup ext(M)^\omega$ to be the set of finite and infinite traces of M and $\mathcal{F}_T$ to be the σ-field generated by the cones C_β for all $\beta \in ext(M)^*$. Let $f : \Omega_M \mapsto \Omega_T$ be a function that assigns to each execution its trace. We can show that f is measurable, and we define P_T as the measure induced by f: $P_T(E) = P_M(f^{-1}(E))$ $\forall E \in \mathcal{F}_T$.

Finally, given a simple probabilistic automaton M and a scheduler ζ for M, we can define a probability space on the set traces of M by using the same construction on $etree(M, \zeta)$, which is a fully probabilistic automaton.

Bisimulation The notion of bisimulation, originally defined for transition systems by Park [Par81], became very popular in Concurrency Theory after

[2]This is true because we do not consider probabilistic schedulers. If we considered such schedulers then the execution tree would no longer be a simple automaton.

Milner used it as one of the fundamental notions in his *Calculus of Communicating Systems* [Mil89]. In the probabilistic setting, an extension of this notion was first proposed by Larsen and Skou [LS91]. Later, many variants were investigated, for various probabilistic models. We recall here the definition of (probabilistic) bisimulation, tailored to probabilistic automata.

If $\mathcal{R}$ is an equivalence relation over a set S, then we can lift the relation to probability distributions over S by considering two distributions related if they assign the same probability to the same equivalence classes. More formally two distributions μ_1, μ_2 are equivalent, written $\mu_1 \mathcal{R} \mu_2$, iff for all equivalence classes $\mathcal{E} \in S/\mathcal{R}$, $\mu_1(\mathcal{E}) = \mu_2(\mathcal{E})$.

Let $(S, q, A, \mathcal{D})$ be a probabilistic automaton. An equivalence relation $\mathcal{R} \subseteq S \times S$ is a *strong bisimulation* iff for all $s_1, s_2 \in S$ and for all $a \in A$

- if $s_1 \xrightarrow{a} \mu_1$ then there exists μ_2 such that $s_2 \xrightarrow{a} \mu_2$ and $\mu_1 \mathcal{R} \mu_2$,
- if $s_2 \xrightarrow{a} \mu_2$ then there exists μ_1 such that $s_1 \xrightarrow{a} \mu_1$ and $\mu_1 \mathcal{R} \mu_2$.

We write $s_1 \sim s_2$ if there is a strong bisimulation that relates them.

2.5 CCS with internal probabilistic choice

In this section we present an extension of standard CCS ([Mil89]) obtained by adding internal probabilistic choice. The resulting calculus can be seen as a simplified version of the probabilistic π-calculus presented in [HP00, PH05] and it is similar to the one considered in [DPP05]. The restriction to CCS and to internal choice is suitable for the scope of this thesis.

Let a range over a countable set of *channel names*. The syntax of CCS_p is the following:

$\alpha ::= a \mid \bar{a} \mid \tau$	**prefixes**
$P, Q ::=$	**processes**
$\alpha.P$	prefix
$\mid P \mid Q$	parallel
$\mid P + Q$	nondeterministic choice
$\mid \sum_i p_i P_i$	internal probabilistic choice
$\mid (\nu a)P$	restriction
$\mid\ !P$	replication
$\mid 0$	nil

where the p_i's in the probabilistic choice should be non-negative and their sum should be 1. We will also use the notation $P_1 +_p P_2$ to represent a binary sum $\sum_i p_i P_i$ with $p_1 = p$ and $p_2 = 1 - p$.

The semantics of a CCS_p term is a probabilistic automaton defined inductively on the basis of the syntax according to the rules in Figure 2.1. We write $s \xrightarrow{a} \mu$ when (s, a, μ) is a transition of the probabilistic automaton. Given a process Q and a measure μ, we denote by $\mu \mid Q$ the measure μ' such that $\mu'(P \mid Q) = \mu(P)$ for all processes P and $\mu'(R) = 0$ if R is not of the form $P \mid Q$. Similarly $(\nu a)\mu = \mu'$ such that $\mu'((\nu a)P) = \mu(P)$.

$$\text{ACT}\quad \frac{}{\alpha.P \xrightarrow{\alpha} \delta(P)} \qquad \text{RES}\quad \frac{P \xrightarrow{\alpha} \mu \qquad \alpha \neq a, \overline{a}}{(\nu a)P \xrightarrow{\alpha} (\nu a)\mu}$$

$$\text{SUM1}\quad \frac{P \xrightarrow{\alpha} \mu}{P + Q \xrightarrow{\alpha} \mu} \qquad \text{SUM2}\quad \frac{Q \xrightarrow{\alpha} \mu}{P + Q \xrightarrow{\alpha} \mu}$$

$$\text{PAR1}\quad \frac{P \xrightarrow{\alpha} \mu}{P \mid Q \xrightarrow{\alpha} \mu \mid Q} \qquad \text{PAR2}\quad \frac{Q \xrightarrow{\alpha} \mu}{P \mid Q \xrightarrow{\alpha} P \mid \mu}$$

$$\text{COM}\quad \frac{P \xrightarrow{a} \delta(P') \quad Q \xrightarrow{\overline{a}} \delta(Q')}{P \mid Q \xrightarrow{\tau} \delta(P' \mid Q')} \qquad \text{PROB}\quad \frac{}{\sum_i p_i P_i \xrightarrow{\tau} \sum_i p_i\, \delta(P_i)}$$

$$\text{REP1}\quad \frac{P \xrightarrow{\alpha} \mu}{!P \xrightarrow{\alpha} \mu \mid !P} \qquad \text{REP2}\quad \frac{P \xrightarrow{a} \delta(P_1) \quad P \xrightarrow{\overline{a}} \delta(P_2)}{!P \xrightarrow{\tau} \delta(P_1 \mid P_2 \mid !P)}$$

Figure 2.1: The semantics of CCS_p.

A transition of the form $P \xrightarrow{a} \delta(P')$, i.e. a transition having for target a Dirac measure, corresponds to a transition of a non-probabilistic automaton (a standard labeled transition system). Thus, all the rules of CCS_p imitate the ones of CCS except from PROB. The latter models the internal probabilistic choice: a silent τ transition is available from the sum to a measure containing all of its operands, with the corresponding probabilities.

Note that in the produced probabilistic automaton, all transitions to non-Dirac measures are silent. This is similar to the *alternating model* [HJ89], however our case is more general because the silent and non-silent transitions are not necessarily alternated. On the other hand, with respect to the simple probabilistic automata the fact that the probabilistic transitions are silent looks like a restriction. However, it has been proved by Bandini and Segala [BS01] that the simple probabilistic automata and the alternating model are essentially equivalent, so, being in between, our model is equivalent as well.

Three

Anonymity Systems

Anonymity is a general notion that arises in activities where the users involved in them wish to keep their identity secret. This chapter provides an introduction to anonymity systems. First, we give a brief discussion about the variety of anonymity notions and their classification. Then, two well-known anonymity protocols from the literature, namely the Dining Cryptographers and Crowds, are presented and their anonymity guarantees are discussed. These protocols serve as running examples throughout the thesis. Finally, we give a brief presentation of various other anonymity protocols, to give an overview of the various designs used for anonymity.

The discussion in this chapter is informal. A formal definition of anonymity systems is given in Chapter 4. The formalization of various anonymity properties is the topic of Chapters 5, 6 and 7.

3.1 Anonymity properties

Due to the generic nature of the term, anonymity does not refer to a uniquely defined notion or property. On the contrary, it describes a broad family of properties with the common feature, generally speaking, that they try to hide the relationship between an observable action (for example, a message sent across a public network) and the identity of the users involved with this action or some other sensitive event that we want to keep private. When we analyze an anonymous system we must define this notion more precisely by answering questions like: "Which identity do we want to hide?", "From whom?" and "To what extent?". The answers to these questions lead to different notions of anonymity.

Even though anonymity protocols can vary a lot in nature, the main agents involved in an anonymity protocol are usually the *sender*, who initiates an action, for example sends a message, and the *receiver* who receives the message and responds accordingly. Since a direct communication between two users is usually exposed, in most protocols these agents are communicating through a number of *nodes* that participate in the protocol, for example by forwarding messages and routing back the replies.

It is worth noting that in the attacker model usually used in the analysis of anonymity systems, the above attacker agents can intercept messages routed

through them, they can send messages to other users, but they cannot intercept messages sent to other members, which is allowed, for example, in the so-called Dolev-Yao model. The reason is that an attacker who can see the whole network is too powerful, leading to the collapse of the anonymity in most of the discussed systems. An attacker with these capabilities is called a *global attacker*

Based on the involved agents we have the following notions of anonymity.

- Sender anonymity to a node, to the receiver or to a global attacker.
- Receiver anonymity to any node, to the sender or to a global attacker.
- Sender-responder unlinkability to any node or a global attacker. This means that a node may know that A sent a message and B received one, but not that A's message was actually received by B.

Moreover, we could consider an attacker that is a combination of a global attacker, sender, receiver and any number of nodes inside the system, or other variations. Pfitzmann and Hanse [PK04] provide an extended discussion on this topic.

Considering the level of anonymity provided by a system, Reiter and Rubin [RR98] provide the following useful classification:

Beyond suspicion
: From the attacker's point of view, a user appears no more likely to be the originator of the message than any other potential user in the system.

Probable innocence
: From the attacker's point of view, a user appears no more likely to be the originator of the message than to not be the originator.

Possible innocence
: From the attacker's point of view, there is a non-negligible probability that the originator is someone else.

The above properties are in decreasing order of strength with each one implying the ones below. Beyond suspicion states that no information about the user can be revealed to the attacker. Probable innocence allows the attacker to suspect a user with higher probability that the others, but gives to the user the right to "plead innocent" in the sense that it is more probable that he did not send the message than that he did. Finally, possible innocence is much weaker, it only requires that the user is not totally exposed.

3.2 Anonymity protocols

3.2.1 Dining Cryptographers

This protocol, proposed by Chaum in [Cha88], is arguably the most well-known anonymity protocol in the literature. It is one of the first anonymity protocols ever studied and one of the few that offers *strong anonymity* (defined in Chapter 5) through the use of a clever mechanism.

The protocol is usually demonstrated in a situation where three cryptographers are dining together with their master (usually the National Security

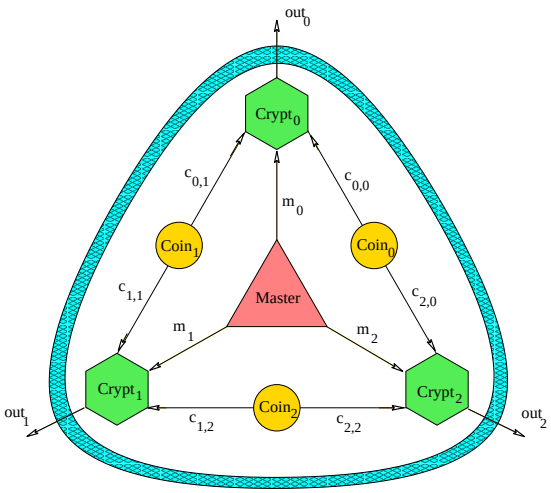


Figure 3.1: The Dining Cryptographers protocol

Agency). At the end of the dinner, each of them is secretly informed by the master whether he should pay the bill or not. So, either the master will pay, or he will ask one of the cryptographers to pay. The cryptographers, or some external observer, would like to find out whether the payer is one of them or the master. However, if the payer is one of them, they also wish to maintain the anonymity of the identity of the payer. Of course, we assume that the master himself will not reveal this information, and also we want the solution to be distributed, i.e. communication can be achieved only via message passing, and there is no central memory or central *coordinator* which can be used to find out this information.

The Dining Cryptographers protocol offers a solution to this problem. Each cryptographer tosses a coin which is visible to himself and to his neighbor to the right, as shown in Figure 3.1. Each cryptographer then observes the two coins that he can see, and announces *agree* or *disagree*. If a cryptographer is not paying, he will announce *agree* if the two sides are the same and *disagree* if they are not. However, if he is paying then he will say the opposite. It can be proved that if the number of *disagrees* is even, then the master is paying; otherwise, one of the cryptographers is paying. Furthermore, if one of the cryptographers is paying, then neither an external observer nor the other two cryptographers can identify, from their individual information, who exactly is paying, assuming that the coins are fair.

The protocol can be easily generalized to an arbitrary number of cryptographers on an arbitrary connection graph, communicating any kind of data. In the general setting, each connected pair of cryptographers share a common secret (the value of the coin) of length n, equal to the length of the transmitted data. The secret is assumed to be drawn uniformly from its set of possible values. Then each user computes the XOR of all its shared secrets and announces publically the sum. The user who wants to transmit data adds also the data to the sum. Then the sum of all announcements is equal to the transmitted data, since all secrets are added twice, assuming that there is only one sender

at the same time. A group of users might collaborate to expose the identity of the sender or, in general, any subset of the secrets might be revealed by any means. After removing the edges corresponding to the revealed secrets, it can be shown that the protocol offers strong anonymity among the connected component of the graph to which the sender belongs, assuming that all coins are fair. That is, the attacker can detect to which connected component the sender belongs but he can gain no more information about which member of the component is the actual sender.

However, these almost perfect anonymity properties come at a cost which, in the case of the Dining Cryptographers, is the low efficiency of the protocol. All users need to communicate at the same time in order to send just one message, thus the protocol can be used only in a relatively small scale. Moreover, if more than one users needs to transmit at the same time then some kind of coordination mechanism is needed to avoid conflicts or detect them and resend the corresponding messages.

The Dining Cryptographers is used as a running example in many parts of this thesis and some interesting new results are also obtained. In Chapter 5 a formal definition of strong anonymity is given and the original proof of Chaum is reproduced, showing that the protocol satisfies strong anonymity in any connected network graph, assuming that the coins are fair. In Chapter 6 the case of unfair coins is considered, where strong anonymity no longer holds. In this case, sufficient and necessary conditions are given for a weaker anonymity property, namely *probable innocence*, for various kinds of network graphs.

In Chapter 7 we consider the case where a new edge (that is a new coin) is added to the graph. We show that for all graphs and any probabilities of the coins this operation strengthens the anonymity of the system, a property expressed in terms of strong anonymity, probable innocence and the quantitative measure of anonymity proposed in the same chapter. Moreover, it is shown that strong anonymity can hold even in the presence of unfair coins and a sufficient and necessary condition is given: an instance of the Dining Cryptographers is strongly anonymous if and only if its graph has a spanning tree consisting only of fair coins. Also in Chapter 7, we demonstrate a model-checking approach and show how to compute the degree of anonymity of the protocol automatically, obtaining a graph of the degree of anonymity as a function of the probability of the coins.

Finally, in Chapter 10 we consider the case where the cryptographers can make their announcements in any order and this order is selected nondeterministically. We extend the notion of strong anonymity to the nondeterministic setting and show that it holds for the Dining Cryptographers only if the scheduler's choices do not depend on the coins or on the selection of the master. An analysis of the protocol with a nondeterministic master is also performed.

3.2.2 Crowds

This protocol, presented in [RR98], allows Internet users to perform web transactions without revealing their identity. When a user communicates with a web server to request a page, the server can know from which IP address the request was initiated. The idea, to obtain anonymity, is to randomly route the request through a crowd of users. The routing protocol ensures that, even when a user appears to send a message, there is a substantial probability that he is simply

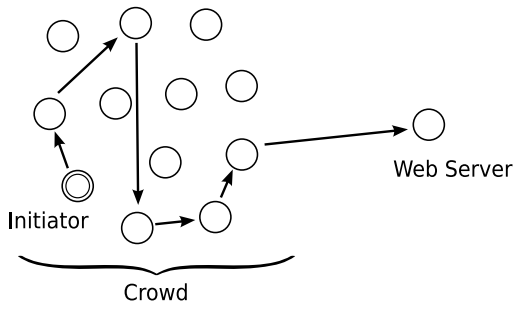


Figure 3.2: The Crowds protocol

forwarding it for somebody else.

More specifically a *crowd* is a group of m users who participate in the protocol. Some of the users may be corrupted which means they can collaborate in order to reveal the identity of the originator. Let c be the number of such users and $p_f \in (0, 1]$ a parameter of the protocol. When a user, called the *initiator* or *originator*, wants to request a web page he must create a *path* between him and the server. This is achieved by the following process, also displayed in Figure 3.2.

- The initiator selects randomly a member of the crowd (possibly himself) and forwards the request to him. We will refer to this latter user as the *forwarder*.
- A forwarder, upon receiving a request, flips a biased coin. With probability $1 - p_f$ he delivers the request directly to the server. With probability p_f he selects randomly, with uniform probability, a new forwarder (possibly himself) and forwards the request to him. The new forwarder repeats the same procedure.

The response from the server follows the same route in the opposite direction to return to the initiator. Moreover, all communication in the path is encrypted using a *path key*, mainly to defend against local eavesdroppers (see [RR98] for more details).

Each user is considered to have access only to the traffic routed through him, so he cannot intercept messages addressed to other users. With respect to the web server the protocol offers strong anonymity. This is ensured by the fact that the initiator never sends the message directly to the server, there is at least one step of forwarding. After this step the message will be in possession of any user with equal probability. As a consequence, the last user in the path, that is the one observed by the web server, can be anyone with equal probability, thus the web server can gain no information about the identity of the initiator.

The more interesting case, however, is the anonymity wrt a corrupted user that participates in the protocol. In this case, the initiator might try to forward the message to the attacker, so the latter can gain more information than the end server. We say that a user is *detected* if he sends a message to a corrupted user. Then it is clear that the initiator, since he always appears in a path, is more likely to be detected than the rest of the users. Thus detecting a user

increases his probability of being the initiator, so strong anonymity cannot hold. However, if the number of corrupted users is not too big, the protocol can still satisfy probable innocence, meaning that the detected user is still less likely to be the originator than all the other users together, even though he is more likely than each other user individually. In [RR98] it is shown that Crowds satisfies probable innocence if $m \geq \frac{p_f}{p_f - 1/2}(c+1)$.

Crowds is also used as a running example in many various parts of the thesis. In Chapter 6 a formal definition of probable innocence is given, combining the features of two existing definitions from the literature. Using the new definition an alternative proof of probable innocence for Crowds is given, arriving at the same sufficient and necessary condition. In Chapter 7 we use model-checking to compute the degree of anonymity of a Crowds instance, while varying the number of corrupted users and the probability p_f of forwarding a message. The obtained graph shows the trade-off between the anonymity and the efficiency of the protocol and can be used to fine-tune its parameters. Finally, in Chapter 9 an instance of Crowds in a non-symmetric network is used to demonstrate an improved bound on the probability of error developed in the same chapter.

3.2.3 Other protocols

MIXes [Cha81] provide anonymity by forwarding messages from node to node, but instead of forwarding each message as it arrives, the nodes wait until they have received a number of messages and then forward them in a mixed order. When done correctly this can provide sender anonymity, receiver anonymity as well as sender-receiver unlinkability, wrt an attacker that can see the whole network. This can be done without requiring all of the nodes to consistently broadcast packets. One draw back is that each node has to hold a message until it has enough messages to properly mix them up, which might add delays if the traffic is low. For this reason, some MIXes implementations add dummy messages if the traffic is low, to provide shelter for the real ones. Another problem is that, if the attacker can send $n-1$ messages to the MIX himself, where n is the MIX capacity, then he can recognize his own messages in the output and thus relate the sender and receiver of the remaining one.

Onion routing is a general-purpose protocol [SGR97] that allows anonymous connection over public networks on condition that the sender knows the public keys of all the other nodes. Messages are randomly routed through a number of nodes called *Core Onion Routers (CORs)*. In order to establish a connection, the initiator selects a random path through the CORs and creates an onion, a recursively layered data structure containing the necessary information for the route. Each layer is encrypted with the key of the corresponding COR. When a COR receives an onion, a layer is "unwrapped" by decrypting it with the COR's private key. This reveals the identity of the next router in the path and a new onion to forward to that router. Since inner layers are encrypted with different keys, each router obtains no information about the path, other than the identity of the following router.

There are two possible configurations for an end-user. They can either run their own COR (local-COR configuration) or use one of the existing ones (remote-COR). The first requires more resources, but the second provides bet-

ter anonymity. Onion routing has also been adapted to a number of other settings.

The Ants protocol [GSB02] was designed for ad-hoc networks, in which nodes do not have fixed positions. In this setting, each node has a pseudo identity which can be used to send messages to a node, but does not give any information about its true identity. In order to search the network, a node broadcasts a search message with its own pseudo identity, a unique message identifier and a time-to-live counter. The search message is sent to all of the node's neighbors, which in turn send the message to all of their neighbors until the time-to-live counter runs out. Upon receiving a message, a node records the connection on which the message was received and the pseudo address of the sender. Each node dynamically builds and maintains a routing table for all the pseudo identities it sees. This table routes messages addressed to a pseudo identity along the connection over which the node has received the most messages from that pseudo identity. To send a message to a particular pseudo identity, a node sends a message with the pseudo identity as a "to" address. If a node has that pseudo address in its table, it forwards the message along the most used connection. Otherwise, it forwards the message to all its neighbors.

This is similar to how real ants behave, they look for food by following the Pheromones traces of other ants. The design for mobile Ad-hoc devices works well for anonymity because mobile devices do not have permanent unique address that can be used for routing, but parts of the protocol, such as continually updating the routing tables are designed for devices that change there location, and may be redundant in a peer-to-peer network of stationary nodes. Gunes et al. provide a detailed efficiency analysis of this protocol, but as yet, there is no published analysis of the anonymity it provides. An important element that affects anonymity in this system is the implementation of the time-to-live counter, which is usually done probabilistically.

Freenet [CSWH00] is a searchable peer-to-peer system for censorship resistant document storage. It is both an original design for anonymity and an implemented system. While it does not aim to hide the provider of a particular file it does aim to make it impossible for an attacker to find all copies of a particular file. A key feature of the Freenet system is that each node will store all the files that pass across it, deleting the least used if necessary. A hash of the title (and other key words) identifies the files. Each node maintains a list of the hashes corresponding to the files on immediately surrounding nodes. A search is carried out by first hashing the title of the file being searched for, and then forwarding the request to the neighboring node that has the file with the most similar hash value. The node receiving the request forwards it in the same way. If a file is found, it is sent back along the path of the request. This unusual search method implements a node-to-node broadcast search one step at a time. Over time it will group files with similar title hash values, making the search more efficient.

Return Address Spoofing can be used to hide the identity of the sender. The headers of messages passed across the Internet include the IP address of

the sender. This address is not used by routers, so it does not have to be correct. The Transmission Control Protocol (TCP) uses this return address to send acknowledgments and control signals, but the User Datagram Protocol (UDP) does not require these controls. Simply by using the UDP protocol and entering a random return address, a sender can effectively send data and hide its identity from the receiver. Without the controls of TCP, packets are liable to loss or congestion. However, if the receiver has an anonymous back channel to communicate with the sender, it can use this to send control signals. A problem with UDP-spoofing is that such behavior is associated with wrongdoing, and so it is often prohibited by ISPs.

Broadcast can be used to provide receiver anonymity by ensuring that enough other people receive the message to obscure the intended recipient. A broadcast can be performed in an overlay network by having each node send a message to all of its neighbors, which in turn send it to all of their neighbors, and so on. If a unique identity is added to the message, nodes can delete recurrences of the same message and stop loops from forming. In large networks it may be necessary to include some kind of time-to-live counter to stop the message flooding the network. In anonymous systems this counter is usually probabilistic. One of the most useful methods of broadcasting is Multicasting [Dee89].

Part I

Probabilistic Approach

Four

A probabilistic framework to model anonymity protocols

In this chapter we establish the basic mathematical settings of our probabilistic approach to anonymity protocols.

Anonymity protocols try to hide the link between a set $\mathcal{A}$ of *anonymous events* and a set $\mathcal{O}$ of *observable events*. For example, a protocol could be designed to allow users to send messages to each other without revealing the identity of the sender. In this case, $\mathcal{A}$ would be the set of (the identities of) the possible users of the protocol, if only one user can send a message at a time, or the powerset of the users, otherwise. On the other hand, $\mathcal{O}$ could contain the sequences of all possible messages that the attacker can observe, depending on how the protocol works.

From the mathematical point of view, a probability distribution on $\mathcal{A} \times \mathcal{O}$ provides all the information that we need about the joint behavior of the protocol and the users. From $p(a, o)$ (in the discrete case) we can derive, indeed, the marginal distributions $p(a)$ and $p(o)$, and the conditional distributions $p(o|a)$ and $p(a|o)$.

Most of the times, however, one is interested in abstracting from the specific users and their distribution, and proving properties about the protocol itself, aiming at *universal anonymity properties* that hold for all possible sets of users (provided they follow the rules of the protocol). For this purpose, it is worth recalling that the joint distribution $p(a, o)$ can be decomposed as $p(a, o) = p(o|a)p(a)$. This decomposition singles out exactly the contributions of the protocol and of the users to the joint probability: $p(a)$, in fact, is the probability associated to the users, while $p(o|a)$ represents the probability that the protocol produces o given that the users have produced a. The latter clearly depends only on the internal mechanisms of the protocol, not on the users.

As a consequence in the next section we define an *anonymity system* as a collection of probability measures $p_c(\cdot|a)$ on $\mathcal{O}$ (or a σ-field on $\mathcal{O}$ in the general case), one for each anonymous event a. The measure $p_c(\cdot|a)$ describes the outcome of the system when it is executed with a as the anonymous event. The intention is that $p_c(\cdot|a)$ is a conditional probability, however it is not defined as such, it is given as a specification of the system and we use the notation p_c to remind us of this fact. The system, together with a probability distribution

on the anonymous events, will define an *anonymity instance* which induces a probability measure on $\mathcal{A} \times \mathcal{O}$, extending the construction $p(a, o) = p(o|a)p(a)$ to the general case. Following the intuition, the conditional probabilities on the induced measure will coincide with the probability measures $p_c(\cdot|a)$ of the system.

Finally, in Section 4.2 will define anonymity systems that are produced by composing two systems or by repeating the same system multiple times, with the same anonymous event.

Examples of anonymous events In protocols where one user performs an action of interest (such as paying in our Dining Cryptographers example) and we want to protect his identity, the set $\mathcal{A}$ would be the same as the set I of the users of the protocol. In the dining cryptographers, we take $\mathcal{A} = \{c_1, c_2, c_3, m\}$ where c_i means that cryptographer i is paying and m that the master is paying. In protocols where k users can perform the action of interest simultaneously at each protocol execution, $\mathcal{A}$ would contain all k-tuples of elements of I. Another interesting case are MIX protocols, in which we are not interested in protecting the fact that someone sent a message (this is indeed detectable), but instead, the link between the sender and the receiver, when k senders send messages to k receivers simultaneously. In that case we consider the sets I_s, I_r of senders and receivers respectively, and take $\mathcal{A}$ to contain all k-tuples of pairs (a, a') where $a \in I_s, a' \in I_r$.

4.1 Formal definition of an anonymity system

Let $\mathcal{A}$ be a set of hidden or anonymous events, that we assume to be countable and let $\mathcal{O}$ be a set of observables, possibly uncountable. The restriction on countable sets $\mathcal{A}$ is realistic, since the anonymous information in practice is the identity of users, or data that have a finite representation, to be stored on a machine. On the other hand, the observable information might be the outcome of an infinite procedure, for example traces of an infinite process, which in general can be uncountable.

We assume that any possible outcome of our system consists of a pair (a, o) where $a \in \mathcal{A}$ is the anonymous event that "happened", for example the user who sent a message in a network or the password that was chosen, and $o \in \mathcal{O}$ is the observable that was produced. Thus we would like to define our sample space as $\mathcal{A} \times \mathcal{O}$ and obtain a probability measure on a σ-field on $\mathcal{A} \times \mathcal{O}$. However, as explained in the beginning of this chapter, defining such a measure would require us to assign probabilities to the anonymous events, but these probabilities are not part of the system: they model the "behavior" of the users at a specific instance of the system. The protocol itself assigns probabilities to each observable event when some anonymous event happens, independently form the probability of the anonymous event.

Thus, we first define a σ-field $\mathcal{F}_o$ on $\mathcal{O}$. The elements of $\mathcal{F}_o$ are called *observable events* and correspond to the events that the attacker can observe and assign probabilities to. Then we provide for every anonymous event $a \in \mathcal{A}$ a probability measure $P_c(\cdot|a)$ over $\mathcal{F}_o$ which models the behavior of our system when a occurs.

Definition 4.1.1 (Anonymity system)**.** *An* anonymity system *is a tuple* $(\mathcal{A}, \mathcal{O}, \mathcal{F}_o, P_c)$ *where* $\mathcal{A}$ *is a countable set of anonymous events,* $\mathcal{O}$ *is a set (possibly uncountable) of observables,* $\mathcal{F}_o$ *is a* σ*-field over* $\mathcal{O}$ *and* $P_c = \{P_c(\cdot|a) \mid a \in \mathcal{A}\}$ *is a collection of probability measures over* $\mathcal{F}_o$*.*

Note that the above definition is similar to the definition of a channel on a generic probability space (see for example [Gra90]) with the extra restriction that the set of input values is countable.

Up to now we have not considered probabilities on anonymous events. To describe completely an instance of an anonymity system we also need to specify a (discrete) probability distribution $P_{\mathcal{A}}$ on $\mathcal{A}$, that is a function $P_{\mathcal{A}} : \mathcal{A} \mapsto [0, 1]$ such that

$$\sum_{a \in \mathcal{A}} P_{\mathcal{A}}(a) = 1 \tag{4.1}$$

We can now define an anonymity instance and the probability space on $\mathcal{A} \times \mathcal{O}$ that it induces.

Definition 4.1.2 (Anonymity instance)**.** *An* instance *of an anonymity system is a tuple* $(\mathcal{A}, \mathcal{O}, \mathcal{F}_o, P_c, P_{\mathcal{A}})$ *where* $(\mathcal{A}, \mathcal{O}, \mathcal{F}_o, P_c)$ *is an anonymity system and* $P_{\mathcal{A}}$ *is a discrete probability distribution on* $\mathcal{A}$*. We define the probability space* $(\Omega, \mathcal{F}, P)$ *induced by the anonymity instance as follows:*

- $\Omega = \mathcal{A} \times \mathcal{O}$
- *Let* $\mathcal{R} = \{A \times O | A \in 2^{\mathcal{A}}, O \in \mathcal{F}_o\}$ *and define* $\mathcal{F}$ *as the* σ*-field generated by* $\mathcal{R}$*.*
- *We define* $P_r : \mathcal{R} \to [0, 1]$ *as*

$$P_r(E) = \sum_{a \in \mathcal{A}} P_c(\mathrm{obs}_a(E)|a) P_{\mathcal{A}}(a) \quad \forall E \in \mathcal{R} \tag{4.2}$$

  where $\mathrm{obs}_a(E) = \{o | (a, o) \in E\}$*. Then* P *is the unique probability measure that extends* P_r *on* $\mathcal{F}$*.*

We first have to show that the probability space in the above definition is well defined. The proof will be based on an extension theorem to lift a measure from a semiring to a σ-field.

Definition 4.1.3. *Let* X *be a set. A collection* $\mathcal{D}$ *of subsets of* X *is called a* semiring *iff* $\emptyset \in \mathcal{D}$ *and for all* $A, B \in \mathcal{D}$ *we have* $A \cap B \in \mathcal{D}$ *and* $A \setminus B = \bigcup_{i=1}^{n} C_i$ *for some finite* n *and pairwise disjoint* $C_i \in \mathcal{D}$*.*

Theorem 4.1.4 ([Bil95], Theorem 11.3, page 166)**.** *Let* $\mathcal{D}$ *be a semi-ring and let* $\mu : \mathcal{D} \to [0, \infty]$ *be a function that is finitely additive, countably subadditive and such that* $\mu(\emptyset) = 0$*. Then* μ *extends to a unique measure on the* σ*-field generated by* $\mathcal{D}$*.*

Proposition 4.1.5. *Let* $(\mathcal{A}, \mathcal{O}, \mathcal{F}_o, P_c, P_{\mathcal{A}})$ *be an anonymity instance. The probability space* $(\Omega, \mathcal{F}, P)$ *of Definition 4.1.1 is well-defined.*

Proof. We first show that $\mathcal{R} = \{A \times O | A \in 2^{\mathcal{A}}, O \in \mathcal{F}_o\}$ is a semiring. We have that $(A_1 \times O_1) \cap (A_2 \times O_2) = (A_1 \cap A_2) \times (O_1 \cap O_2)$ and $2^{\mathcal{A}}, \mathcal{F}_o$ are closed under intersection, so $\mathcal{R}$ is also closed under intersection. Also, if $R_1, R_2 \in \mathcal{R}$ with $R_2 = (A_2 \times O_2)$ then

$$\begin{aligned} R_1 \setminus R_2 &= R_1 \cap (A_2 \times O_2)^c \\ &= R_1 \cap ((A_2^c \times O_2) \cup (A_2 \times O_2^c) \cup (A_2^c \times O_2^c)) \\ &= (R_1 \cap (A_2^c \times O_2)) \cup (R_1 \cap (A_2 \times O_2^c)) \cup (R_1 \cap (A_2^c \times O_2^c)) \end{aligned}$$

Since $2^{\mathcal{A}}, \mathcal{F}_o$ are closed under complement, we see that $R_1 \setminus R_2$ can be written as a finite union of pairwise disjoint elements of $\mathcal{R}$. Also $\emptyset \in \mathcal{R}$ so $\mathcal{R}$ is a semiring.

We now show that P_r is countably additive on $\mathcal{R}$. We notice that

$$\text{obs}_a(\bigcup_i E_i) = \bigcup_i \text{obs}_a(E_i) \tag{4.3}$$

thus:

$$\begin{array}{lcll} P_r(\bigcup_i E_i) & = & \sum_{a \in \mathcal{A}} P_c(\text{obs}_a(\bigcup_i E_i)|a) P_{\mathcal{A}}(a) & (4.2) \\ & = & \sum_{a \in \mathcal{A}} P_c(\bigcup_i \text{obs}_a(E_i)|a) P_{\mathcal{A}}(a) & (4.3) \\ & = & \sum_{a \in \mathcal{A}} \left(\sum_i P_c(\text{obs}_a(E_i)|a) \right) P_{\mathcal{A}}(a) & P_c(\cdot|a) \text{ count. additive} \\ & = & \sum_i \sum_{a \in \mathcal{A}} P_c(\text{obs}_a(E_i)|a) P_{\mathcal{A}}(a) & \text{rearrangement} \\ & = & \sum_i P_r(E_i) & (4.2) \end{array}$$

The rearrangement is possible since the sum converges and its terms are non-negative. So by Theorem 4.1.4 the measure P exists and it is unique. We finally have to show that it is a *probability* measure, that is $P(\Omega) = 1$

$$\begin{array}{lcll} P(\Omega) & = & P_r(\Omega) & \Omega \in \mathcal{R} \\ & = & \sum_{a \in \mathcal{A}} P_c(\text{obs}_a(\Omega)|a) P_{\mathcal{A}}(a) & (4.2) \\ & = & \sum_{a \in \mathcal{A}} P_c(\mathcal{O}|a) P_{\mathcal{A}}(a) & \text{obs}_a(\Omega) = \mathcal{O} \\ & = & \sum_{a \in \mathcal{A}} P_{\mathcal{A}}(a) & P_c(\cdot|a) \text{ is a measure on } \mathcal{F}_o \\ & = & 1 & (4.1) \end{array}$$

□

The intuition behind the construction of P is that we want a measure that assigns probabilities to anonymous events according to $P_{\mathcal{A}}$ and conditional probabilities given an anonymous event a according to $P_c(\cdot|a)$. We define $[a] = \{a\} \times \mathcal{O},\ a \in \mathcal{A}$ and $[O] = \mathcal{A} \times O,\ O \subseteq \mathcal{O}$. We show that the behavior of the constructed measure follows this intuition.

Proposition 4.1.6. *Let $(\mathcal{A}, \mathcal{O}, \mathcal{F}_o, P_c, P_{\mathcal{A}})$ be an anonymity instance and $(\Omega, \mathcal{F}, P)$ the probability space induced by it. The following holds for all $a \in \mathcal{A}$ and all $O \in \mathcal{F}_o$*

1. $P([a]) = P_{\mathcal{A}}(a)$
2. $P([O]|[a]) = P_c(O|a)$ *if* $P([a]) > 0$

Proof.

$$\begin{aligned} P([a]) &= \textstyle\sum_{a' \in \mathcal{A}} P_c(\mathrm{obs}_{a'}([a])|a') P_{\mathcal{A}}(a') && (4.2) \\ &= P_c(\mathrm{obs}_a([a])|a) P_{\mathcal{A}}(a) && \mathrm{obs}_{a'}([a]) = \emptyset \text{ for } a' \neq a \\ &= P_{\mathcal{A}}(a) && \mathrm{obs}_a([a]) = \mathcal{O},\ P_c(\mathcal{O}|a) = 1 \end{aligned}$$

$$\begin{aligned} & P([O]|[a]) \\ &= \frac{P([O] \cap [a])}{P([a])} \\ &= \frac{1}{P([a])} \textstyle\sum_{a' \in \mathcal{A}} P_c(\mathrm{obs}_{a'}([O] \cap [a])|a') P_{\mathcal{A}}(a') && (4.2) \\ &= \frac{1}{P([a])} P_c(\mathrm{obs}_a([O] \cap [a])|a) P_{\mathcal{A}}(a) && \mathrm{obs}_{a'}([O] \cap [a]) = \emptyset, a' \neq a \\ &= P_c(\mathrm{obs}_a([O] \cap [a])|a) && \text{Prop. 4.1.6 case (1)} \\ &= P_c(O|a) && \text{definition of } \mathrm{obs}_a \end{aligned}$$

□

For simplicity we will sometimes write $P(a), P(O)$ for $P([a]), P([O])$ and we will use $P([O]|[a])$ and $P_c(O|a)$ interchangeably.

4.1.1 Finite anonymity systems

In the case where $\mathcal{A}, \mathcal{O}$ are finite we can describe our system using discrete probabilistic distributions. More specifically we always consider $\mathcal{F}_o = 2^{\mathcal{O}}$ and define $P_c(\cdot|a)$ by assigning probabilities to the individual observables.

Definition 4.1.7. *A finite anonymity system is a tuple $(\mathcal{A}, \mathcal{O}, p_c)$ where $\mathcal{A}$ is a finite set of anonymous events, $\mathcal{O}$ is a finite set of observables and for all $a \in \mathcal{A}$: $p_c(\cdot|a)$ is a discrete probability distribution on $\mathcal{O}$, that is*

$$\sum_{o \in \mathcal{O}} p_c(o|a) = 1 \quad \forall a \in \mathcal{A}$$

p_c can be represented by a $|\mathcal{A}| \times |\mathcal{O}|$ matrix M such that $m_{i,j} = p_c(o_j|a_i)$:

	o_1	$\cdots$	o_m
a_1	$p_c(o_1\|a_1)$	$\ldots$	$p_c(o_m\|a_1)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
a_n	$p_c(o_1\|a_n)$	$\ldots$	$p_c(o_m\|a_n)$

Definition 4.1.8. *A finite anonymity instance is a tuple $(\mathcal{A}, \mathcal{O}, p_c, p_{\mathcal{A}})$ where $(\mathcal{A}, \mathcal{O}, p_c)$ is a finite anonymity system and $p_{\mathcal{A}}$ is a discrete probability distribution on $\mathcal{A}$. The induced probability distribution on $\mathcal{A} \times \mathcal{O}$ is defined as*

$$p((a, o)) = p_{\mathcal{A}}(a) p_c(o|a) \quad \forall a \in \mathcal{A}, o \in \mathcal{O}$$

which corresponds to the construction of Definition 4.1.2 in the discrete case.

In Part II we study exclusively finite anonymity systems.

4.2 Protocol composition

In protocol analysis, it is often easier to split complex protocols in parts, analyze each part separately and then combine the results. In this section we define a type of composition where two protocols are executed independently with the same anonymous event.

Definition 4.2.1. *Let* $S_1 = (\mathcal{A}, \mathcal{O}_1, \mathcal{F}_{o1}, P_{c_1}), S_2 = (\mathcal{A}, \mathcal{O}_2, \mathcal{F}_{o2}, P_{c_2})$ *be two anonymity systems with the same set of anonymous events. The* independent composition *of* S_1, S_2*, written* $S_1; S_2$ *is an anonymity system* $(\mathcal{A}, \mathcal{O}, \mathcal{F}_o, P_c)$ *such that*

- $\mathcal{O} = \mathcal{O}_1 \times \mathcal{O}_2$
- *Let* $\mathcal{R} = \{O_1 \times O_2 | O_1 \in \mathcal{F}_{o1}, O_2 \in \mathcal{F}_{o2}\}$*,* $\mathcal{F}_o$ *is the* σ*-field generated by* $\mathcal{R}$*,*
- $P_c(\cdot|a)$ *is the unique probability measure on* $\mathcal{F}_o$ *such that*

$$P_c(O|a) = P_{c_1}(\text{proj}_1(O)|a)\, P_{c_2}(\text{proj}_2(O)|a) \quad \forall O \in \mathcal{R}$$

where $\text{proj}_i(O) = \{o_i | (o_1, o_2) \in O\}$

We can show that this is a well-defined anonymity system in a way similar to Proposition 4.1.5. Note that $P_c(\cdot|a)$ is known in probability theory as the product probability measure of $P_{c_1}(\cdot|a)$, $P_{c_2}(\cdot|a)$.

An interesting case of composition is when a protocol is "repeated" multiple times with the same anonymous event. This situation arises when an attacker can force a user to repeat the protocol many times.

Definition 4.2.2. *The* n*-repetition of an anonymity system* $S = (\mathcal{A}, \mathcal{O}, \mathcal{F}_o, P_c)$ *is the anonymity system* $S^n = S; \ldots; S$*,* n *times.*

4.3 Example: modeling a system using probabilistic automata

Let $\mathcal{A}$ be a finite (for simplicity) set of user identities involved in a protocol that we wish to keep anonymous. For each user $a \in \mathcal{A}$ we have a fully probabilistic automaton $M(a)$ modeling the behavior of the system when a executes the protocol.

We assume that these automata have the same set of external actions, thus the same set of traces. Let $(\Omega_T, \mathcal{F}_T, P_{T(a)})$ be the probability space induced by $M(a)$ on the set of its traces (see Section 2.4), $\Omega_T, \mathcal{F}_T$ being common for all automata. We define our anonymity system $(\mathcal{A}, \mathcal{O}, \mathcal{F}_o, P_c)$ as follows

- $\mathcal{O} = \Omega_T$
- $\mathcal{F}_o = \mathcal{F}_T$
- $P_c(\cdot|a) = P_{T(a)}$

In this system, the observable events that the attacker can see are cones of traces, not single traces, which is reasonable since infinite traces require infinite time to be observed. The probability of observing a cone when a certain user executes the protocol is given by $P_c(\cdot|a)$.

Five

Strong Anonymity

In this chapter we consider the strongest form of anonymity that a system can achieve. In literature there are two interpretations of this notion in the probabilistic setting. One, proposed by Halpern and O'Neill in [HO03, HO05], focuses on the lack confidence of the attacker, and expresses the fact that the a posteriori probabilities of the anonymous events, after each observation, are the same, so the attacker cannot distinguish them. Formally this means that, for any a, a', and o with positive probability

$$p(a|o) = p(a'|o)$$

The other notion focuses on the fact that the attacker cannot learn anything about the anonymous events from the observable outcome of the protocol. Formally, this idea can be expressed as the requirement that the a posteriori probability of each anonymous event after an observation be the same as its a priori probability. This property was used by Chaum in his seminal paper [Cha88] and it was called *conditional anonymity* by Halpern and O'Neill in [HO03, HO05]. An equivalent condition is that the anonymous events and the observable events be (probabilistically) independent, so there is no link that the attacker can establish between the observation and the anonymous event that has produced it. There is yet another equivalent formulation, which consists in requiring that, for each observation, the *likelihood* of the anonymous events be the same. The likelihood of a after observing o is defined as the conditional probability $p(o|a)$, so formally this can be stated as the condition that, for any o and a, a' with positive probability

$$p(o|a) = p(o|a')$$

We can see that this latter property depends only on the protocol, not on the probabilities of the users. This is a feature that we consider crucial, as argued in previous chapters. Hence we will adopt this formulation as definition of the notion of strong anonymity. Note that the difference between our notion and the one proposed as strong anonymity by Halpern and O'Neill consists in replacing $p(a|o)$ by $p(o|a)$.

We should mention that Halpern and O'Neill also propose a formal interpretation of the notion of "beyond suspicion", which is the strongest notion in Reiter and Rubin's hierarchy. This interpretation requires the a posteriori

probability of the anonymous event which actually took place to be smaller or equal to that of any other anonymous event. They state that this definition is strictly weaker than their definition of strong anonymity. In our framework, however, it can be shown that the two definitions would be equivalent. This is because in our framework the probabilities do not depend on the anonymous event that actually took place.

5.1 Formal definition

In this thesis we will adopt the following definition of strong anonymity, similar to the notion of probabilistic anonymity proposed in [BP05]:

Definition 5.1.1 (Strong anonymity)**.** *An anonymity system* $(\mathcal{A}, \mathcal{O}, \mathcal{F}_o, P_c)$ *is* strongly anonymous *if* $\forall a, a' \in \mathcal{A}$*:*

$$P_c(\cdot|a) = P_c(\cdot|a')$$

In the case of a finite anonymity system $(\mathcal{A}, \mathcal{O}, p_c)$, the above definition is equivalent to requiring that $p_c(o|a) = p_c(o|a')$ for all $a, a' \in \mathcal{A}$ and $o \in \mathcal{O}$, which is the same as saying that all the rows of the probability matrix are equal.

The idea is that if all anonymous events produce the same observable events with the same probability, then the attacker can learn no information by observing the output of the protocol. Note that this definition does not depend on the probability of the anonymous events themselves, which in fact is not even part of an anonymity system.

An alternative definition considers all instances of the anonymity system and requires the probability of an anonymous action to be the same before and after the observation. This is the property that was proved by Chaum for the Dining Cryptographers ([Cha88]) and corresponds, as we already mentioned, to the property of *conditional anonymity* in [HO03].

Definition 5.1.2 (Conditional anonymity)**.** *An anonymity system* $(\mathcal{A}, \mathcal{O}, \mathcal{F}_o, P_c)$ *satisfies* conditional anonymity *if for all probability distributions* P_A *on* $\mathcal{A}$*, for all* $a \in \mathcal{A}$*, and all observable events* $O \in \mathcal{F}_o$ *such that* $P([O]) > 0$*, the following holds*

$$P([a]) = P([a]|[O])$$

where P *is the probability measure induced by the anonymity instance* $(\mathcal{A}, \mathcal{O}, \mathcal{F}_o, P_c, P_A)$*.*

We remind that $[a]$ is defined as $[a] = a \times \mathcal{O}$ and $[O] = \mathcal{A} \times O$.

We now show that the two above definitions are equivalent. This is a standard result in probability theory, but we include the proof for the interested reader, since it is only a few lines.

Theorem 5.1.3. *Strong anonymity (Def. 5.1.1) is equivalent to conditional anonymity (Def. 5.1.2).*

Proof. For simplicity we write $P(a), P(O), \ldots$ for $P([a]), P([O]), \ldots$.

$\Rightarrow$) Let P_A be a distribution over $\mathcal{A}$, P the probability measure induced by the anonymity instance and $O \in \mathcal{F}_o$ an observable event such that $P(O) > 0$.

If $P(a) = 0$ then $P(a|O) = 0$ and we are finished. Otherwise from Def. 5.1.1 and since $P_c(O|a) = P(O|a)$ (Prop. 4.1.6) we have $P(O|a) = P(O|a')$ for all $a, a' \in \mathcal{A}$. We first show that $P(O) = P(O|a)$:

$$\begin{array}{rcll} P(O) & = & \sum_{a' \in \mathcal{A}} P(O \cap a') & \text{countable additivity} \\ & = & \sum_{a' \in \mathcal{A}, P(a')>0} P(O|a')P(a') & \\ & = & P(O|a) \sum_{a' \in \mathcal{A}, P(a')>0} P(a') & P(O|a') \text{ constant} \\ & = & P(O|a) & \end{array}$$

Then $P(a|O) = \dfrac{P(O|a)P(a)}{P(O)} = P(a)$.

$\Leftarrow$) Let P_A be a uniform distribution over $\mathcal{A}$, P the probability measure induced by the anonymity instance and $O \in \mathcal{F}_o$ an observable event. If $P(O) = 0$ then $P(O|a) = 0$, the same for all $a \in \mathcal{A}$. Otherwise $P(O|a) = \dfrac{P(a|O)P(O)}{P(a)} = P(O)$, the same for all $a \in \mathcal{A}$. Since $P_c(O|a) = P([O]|[a])$ (Prop. 4.1.6) then $P_c(\cdot|a) = P_c(\cdot|a')$ for all $a, a' \in \mathcal{A}$. □

5.2 Strong anonymity of the dining cryptographers protocol

We give now a proof that the Dining Cryptographers protocol, described in section 3.2.1, satisfies strong anonymity under the assumption of fair coins. The proof comes from [Cha88].

We consider a generalized version of the Dining Cryptographers, with an arbitrary number of cryptographers and coins. Each coin can give either head (interpreted as 0) or tail (interpreted as 1), with uniform probability (fair coin). The coins are placed in an arbitrary way, but each coin is adjacent to (can be read by) exactly two cryptographers. We assume that there is at most one payer, and the goal is to conceal his identity.

In this generalized version, the protocol works as follows: after the payer (if any) is chosen, all the coins get tossed. Then each cryptographer calculates the binary sum of of all its adjacent coins, adding 1 in case he is the payer, and announces the outcome. The protocol reveals the presence of a payer, because the binary sum of all the announcements is 1 iff and only if one of the cryptographers is the payer. This is easy to see: each coin is counted twice, hence the contribution of all coins is 0. More interestingly, the protocol provides anonymity, and it is robust to the possible cooperation of some cryptographers with the attacker, where by cooperation we mean that the values of the coins visible to the corrupted cryptographers are revealed to the attacker.

To state formally the property of anonymity, let us consider the graph G whose vertices are the cryptographers and whose edges are the coins, with the obvious adjacency relation. From G we create a new graph G_o by removing all the edges corresponding to the coins visible to corrupted cryptographers, since these coins are revealed to the attacker. G_o may not be connected, and in particular each corrupted cryptographer is disconnected from all the others since all his edges are removed. Chaum proved that strong anonymity holds within each connected component of G_o. More precisely, from the observation the attacker can single out the connected component G_c of G_o to which the

payer belongs, but he does not gain any information concerning the precise identity of the payer within C_c.

In order to present the proof of anonymity, we need some preliminary definitions. Let n be the number of vertices (cryptographers) of G_c and m the number of edges (coins). Let B be the $n \times m$ incidence matrix of G_c, defined as $b_{i,j} = 1$ if the vertex i is connected to the edge j, 0 otherwise. Each coin c_i takes a value in GF(2), the finite field consisting of $0, 1$ with addition and multiplication modulo 2. Let $\vec{c} = (c_1, \ldots, c_m)$ be a vector in $\mathrm{GF}(2)^m$ composed of the values of all coins. Also let $\vec{r} = (r_1, \ldots, r_n) \in \mathrm{GF}(2)^n$ be the inversion vector defined as $r_k = 1$ if cryptographer k is the payer, 0 otherwise. By the assumption that there is no more than one payer, there is at most one k such that $r_k = 1$. We will denote by $\vec{r}_i$ the inversion vector with $r_i = 1$.

Each cryptographer outputs the sum of its adjacent coins plus a possible inversion, so the output of the protocol is a vector $\vec{o} \in \mathrm{GF}(2)^n$ computed as

$$\vec{o} = B\vec{c} \oplus \vec{r}$$

where operations are performed in $GF(2)$ (that is modulo 2). Since each column of B has exactly two 1s we know that $B\vec{c}$ has even parity (number of 1s), so $\vec{o}$ has the same parity as $\vec{r}$, odd if there is a payer, even otherwise.

Now assuming that there is always a payer in G_c we define a finite anonymity system $S(G_c) = (\mathcal{A}, \mathcal{O}, p_c)$ as follows:

- $\mathcal{A} = \{a_1, \ldots, a_n\}$ where a_i means that cryptographer i is the payer,
- $\mathcal{O} = \{\vec{o} \in \mathrm{GF}(2)^n \,|\, \sum_i o_i = 1\}$, the possible outcomes of the protocol, and
- $p_c(\vec{o}|a_i)$ is the probability of having output $\vec{o}$ when cryptographer i is the payer, that is when the inversion vector is $\vec{r}_i$.

Theorem 5.2.1 (Chaum, [Cha88]). *The anonymity system $S(G_c) = (\mathcal{A}, \mathcal{O}, p_c)$ corresponding to the connected component G_c satisfies strong anonymity.*

Proof. Fix an observable $\vec{o} \in \mathcal{O}$ and a cryptographer $a_i \in \mathcal{A}$. To compute the probability $p_c(\vec{o}|a_i)$ we have to compute all the possible coin configurations that will produce $\vec{o}$ as output. These will be given by the following system of linear equations in GF(2):

$$B\vec{x} = \vec{o} \oplus \vec{r}_i$$

Since all columns of B have exactly two 1s, the sum of all its rows is $\vec{0}$, so they are linearly dependent. On the other hand, all strict subsets of rows of B are linearly independent in GF(2). This is because the sum of two rows (vertices) gives a vertex combining all the edges of the two. If the sum of a subset of rows is $\vec{0}$ it would mean that there is no edge joining them to the rest of the vertices, which is impossible since G_c is connected.

Hence the rank of B is $n-1$ and there are 2^{n-1} vectors in $\mathrm{GF}(2)^n$ that can be written as a linear combination of the columns of B, that is all vectors with even parity (since all columns have even parity). Since $\vec{o} \oplus \vec{r}_i$ has even parity it can be written as a linear combination of the columns, thus the system is solvable and has $2^{m-(n-1)}$ solutions. So $2^{m-(n-1)}$ coin configurations produce the output $\vec{o}$ and since the coins are assumed fair the probability of each configuration is 2^{-m} and the probability of getting $\vec{o}$ (when $\vec{r}_i$ is the inversion vector) is $2^{-(n-1)}$. This is true for all inversion vectors so $p_c(\vec{o}|a_i) = p_c(\vec{o}|a_j) = 2^{-(n-1)}$ for all $a_i, a_j \in \mathcal{A}$ and $\vec{o} \in \mathcal{O}$. □

5.3 Protocol composition

In some cases of anonymity protocols, a user may need to execute the protocol multiple times, and it may be possible that the attacker discovers that the culprit is the same in all executions, even though he does not know which. For example, the dining cryptographers protocol could be performed many times to allow a user to transmit a message in the form of a binary sequence: at each run of the protocol, there is either no payer (transmission of 0) or the payer is the selected user (transmission of 1). The attacker may know that the protocol is being used in this way, thus he may know that whenever there is a payer, it's always the same payer.

The information that the culprit is always the same increases the knowledge of the attacker, hence in principle the repetition of the protocol may weaken the anonymity property. However in case of strong anonymity this is not the case, as we prove in the rest of this section.

First we show that the composition of two anonymity systems (defined in Section 4.2) satisfies strong anonymity if and only if both systems satisfy it.

Proposition 5.3.1. *Let* $S_1 = (\mathcal{A}, \mathcal{O}_1, \mathcal{F}_{o1}, P_{c_1}), S_2 = (\mathcal{A}, \mathcal{O}_2, \mathcal{F}_{o2}, P_{c_2})$ *be two anonymity systems and* $S_1; S_2 = (\mathcal{A}, \mathcal{O}, \mathcal{F}_o, P_c)$. $S_1; S_2$ *satisfies strong anonymity iff both* S_1 *and* S_2 *satisfy it.*

Proof. **if)** If $P_{c_1}(\cdot|a) = P_{c_1}(\cdot|a')$ and $P_{c_2}(\cdot|a) = P_{c_2}(\cdot|a')$ for all $a, a' \in \mathcal{A}$ then

$$\begin{aligned} P_c(O|a) &= P_{c_1}(\mathrm{proj}_1(O)|a)\, P_{c_2}(\mathrm{proj}_2(O)|a) \\ &= P_{c_1}(\mathrm{proj}_1(O)|a')\, P_{c_2}(\mathrm{proj}_2(O)|a') \\ &= P_c(O|a') \end{aligned}$$

for all $O \in \mathcal{F}_o$.
only if) If $\exists O \in \mathcal{O}$ such that $P_c(O|a) \neq P_c(O|a')$ then either $P_{c_1}(\mathrm{proj}_1(O)|a) \neq P_{c_1}(\mathrm{proj}_1(O)|a')$ or $P_{c_2}(\mathrm{proj}_2(O)|a) \neq P_{c_2}(\mathrm{proj}_2(O)|a')$. □

As a corollary we get that any repetition of a strongly anonymous protocol is also strongly anonymous, which conforms to the intuition that a strongly anonymous protocol leaks no information at all.

Corollary 5.3.2. *Let* $S = (\mathcal{A}, \mathcal{O}, \mathcal{F}_o, P_c)$ *be an anonymity system. The* n*-repetition* S^n *of* S *is strongly anonymous, for all* $n >= 1$*, iff* S *is strongly anonymous.*

Proof. Proposition 5.3.1 together with the fact that $S^n = S; \ldots; S$. □

Six

Probable Innocence

The notion of strong anonymity discussed in previous chapter describes the ideal situation in which the protocol does not leak any information concerning the identity of the user. We have shown that this property is satisfied by the Dining Cryptographers with fair coins [Cha88]. Protocols used in practice, however, especially in the presence of attackers or corrupted users, are only able to provide a weaker notion of anonymity.

In [RR98] Reiter and Rubin have proposed a hierarchy of notions of probabilistic anonymity in the context of Crowds. We recall that Crowds is a system for anonymous web surfing aimed at protecting the identity of the users when sending (originating) messages. This is achieved by forwarding the message to another user selected randomly, which in turn forwards the message, and so on, until the message reaches its destination. Some of the users may be corrupted (attackers), and one of the main purposes of the protocol is to protect the identity of the originator of the message from those attackers.

We recall the hierarchy of Reiter and Rubin, already discussed in Section 3.1. Here the *sender* stands for the user that forwards the message to the attacker.

Beyond suspicion From the attacker's point of view, the sender appears no more likely to be the originator of the message than any other potential sender in the system.

Probable innocence From the attacker's point of view, the sender appears no more likely to be the originator of the message than to not be the originator.

Possible innocence From the attacker's point of view, there is a non-negligible probability that the real sender is someone else.

This chapter focuses on the notion of probable innocence. The first goal is, of course, to give a formal definition of this notion. Let us first discuss the formal approaches proposed in literature.

In [RR98] Reiter and Rubin also considered a formal definition of probable innocence, tailored to the characteristics of the Crowds system. This definition is not given explicitly, but it can be derived from the formula that they proved to hold for Crowds under certain conditions. The formula says that the probability that the originator forwards the message to an attacker (given that an

attacker receives eventually the message) is at most 1/2. In other words, their definition expresses a bound on the probability of detection.

Later Halpern and O'Neill proposed in [HO05] a formal interpretation of the hierarchy above in more general terms, and focusing on the confidence of the attacker. In particular their definition of probable innocence holds if for the attacker, given the events that he has observed, the probability that a user i is the culprit (i.e. has performed the action of interest) is not greater than 1/2. Analogously their interpretation of beyond suspicion holds if for the attacker, given the events that he has observed, the probability to be the culprit is not greater for the actual culprit than for any other user in the system.

However, the property of probable innocence that Reiter and Rubin prove formally for the system Crowds in [RR98] does not mention the user's probability of being the originator, but only the probability of the event observed by the attacker. Their property depends only on the way the protocol works, and on the number of the attackers. It is totally independent from the probability distribution on the users to originate the message. As argued in previous chapters, this is a very desirable property, since we do not want the correctness of a protocol to depend on the users' intentions of originating a message. For the stronger notion of anonymity considered in previous chapter, this abstraction from the users' probabilities leads to our notion of strong anonymity, which, we recall, corresponds to the notion of probabilistic anonymity defined in [BP05]. In this sense the formal property considered by Reiter and Rubin is to Halpern and O'Neill's interpretation of probable innocence as our notion of strong anonymity is to Halpern and O'Neill's interpretation of beyond suspicion. The parallel is even stronger. We will see in fact that the difference between the two notions consists in exchanging $p(o|a)$ with $p(a|o)$.

Another desired feature for a general notion of probable innocence is the abstraction from the specific characteristics of Crowds. In Crowds, (at least in its original formulation as given in [RR98]) there are certain symmetries that derive from the assumption that the probability that user i forwards the message to user j is the same for all i and j. The property of probable innocence proved for Crowds in [RR98] depends strongly on this assumption. We want a general notion which protocols may satisfy in the case they do not satisfy the original Crowds' symmetry assumptions.

For completeness, we also consider the composition of protocol executions, with specific focus on the case that the originator is the same and the protocol to be executed is the same. This situation can arise, for instance, when an attacker can induce the originator to repeat the protocol (multiple paths attack). We extend the definition of probable innocence to the case of protocol composition under the same originator, and we study how this property depends on the number of compositions.

Contribution In this chapter we propose a general notion of probable innocence which combines the spirit of the approach of Reiter and Rubin and of the one of Halpern and O'Neill. Namely it expresses a limit both on the attacker's confidence and on the probability of detection. Furthermore, our notion avoids the shortcomings of those previous approaches, namely it does not depend on symmetry assumptions or on the probabilities of the users to perform the action of interest.

We also show that our definition, while being more general than the property that Reiter and Rubin have proved for Crowds, it agrees with the latter under the specific symmetry conditions satisfied by Crowds. Furthermore, we show that in the particular case that the users have uniform probability of being the originator, we obtain a property similar to the definition of probable innocence given by Halpern and O'Neill.

Another contribution is the analysis of the robustness of probable innocence under multiple paths attacks, which induce a repetition of the protocol. We show a general negative result, namely that no protocol can ensure probable innocence under an arbitrary number of repetitions, unless the system is strongly anonymous. This generalizes the result, already known in literature, that Crowds cannot guarantee probable innocence under unbounded multiple path attacks.

Plan of the chapter In the next section we illustrate the Crowds protocol. In Section 6.1 we recall the property proved for Crowds and the definition of probable innocence by Halpern and O'Neill, and we discuss them. In Section 6.2 we propose our notion of probable innocence and we compare it with those of Section 6.1. In Section 6.4 we consider the repetition of an anonymity protocol and we show that we cannot guarantee probable innocence for arbitrary repetition unless the protocol is strongly anonymous. Finally, in Section 6.5 we present some applications and results of our notion to Crowds and to the Dining Cryptographers.

6.1 Existing definitions of probable innocence

As explained in the introduction, in literature there are two different approaches to a formal definition of probable innocence. The first, implicitly considered by Reiter and Rubin, focuses on the probability of the observables and constrains the probability of detecting a user. The second, proposed by Halpern and O'Neill, focuses on the probability of the users and limits the attacker's confidence that the detected user is the originator.

In this section we present the two existing definitions in the literature, and we argue that each of them has a shortcoming: the first does not seem satisfactory when the system is not symmetric. The second depends on the probability distribution of the users.

The Crowds protocol We briefly recall the Crowds protocol, already discussed in Section 3.2.2. The protocol allows Internet users to perform web transactions without revealing their identity. A crowd is a group of m users who participate in the protocol. Some of the users may be corrupted which means they can collaborate in order to reveal the identity of the originator. Let c be the number of such users and p_f a parameter of the protocol, explained below. When a user, called the *initiator* or *originator*, wants to request a web page he must create a *path* between him and the server. This is achieved by the following process: The initiator selects randomly a member of the crowd (possibly himself) and forwards the request to him. We will refer to this latter user as the *forwarder*. A forwarder, upon receiving a request, flips a biased coin. With probability $1 - p_f$ he delivers the request directly to the server.

With probability p_f he selects randomly, with uniform probability, a new forwarder (possibly himself) and forwards the request to him. The new forwarder repeats the same procedure.

6.1.1 First approach (limit on the probability of detection)

Reiter and Rubin ([RR98]) consider a notion which limits the probability of the originator being observed by a corrupted member, that is being directly before him in the path. More precisely, let I denote the event "the originator is observed by a corrupted member" and H the event "at least one corrupted member appears in the path". Then the intended property is expressed by

$$p(I|H) \leq 1/2 \tag{6.1}$$

In [RR98] it is proved that this property is satisfied by Crowds if $m \geq \frac{p_f}{p_f - 1/2}(c+1)$.

For simplicity, we suppose that a corrupted user will not forward a request to other crowd members, so at most one user can be observed. This approach is also followed in [RR98, Shm02, WALS02] and the reason is that by forwarding the request the corrupted users cannot gain any new information since forwarders are chosen randomly.

We now express the above definition in the framework of Chapter 4. Since $I \Rightarrow H$ we have $p(I|H) = p(I)/p(H)$. If A_i denotes that "user i is the originator" and D_i is the event "the user i was observed by a corrupted member (appears in the path right before the corrupter member)" then $p(I) = \sum_i p(D_i \wedge A_i) = \sum_i p(D_i|A_i)p(A_i)$. Since $p(D_i|A_i)$ is the same for all i then the definition (6.1) can be written $\forall i : p(D_i|A_i)/p(H) \leq 1/2$.

Assuming that there is at least one corrupted user ($c \geq 1$), we create a finite anonymity system $(\mathcal{A}, \mathcal{O}, p_c)$ as follows:

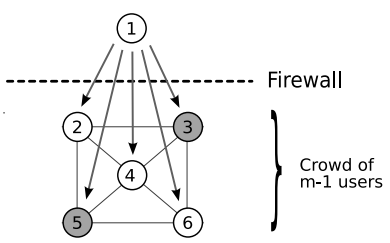
- $\mathcal{A} = \{a_1, \ldots, a_n\}$ is the set of honest crowd members, where $n = m - c$
- $\mathcal{O} = \{o_1, \ldots, o_n\}$ where o_i means that the user i was detected by a corrupted user.
- $p_c(o_i|a_i) = \dfrac{p(D_i|A_i)}{p(H)}$

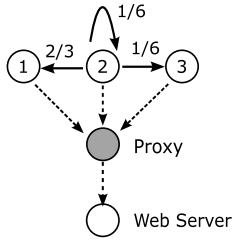
This system considers only the honest users (there is no anonymity requirement for corrupted users) under the assumption that a user is always detected, so all probabilities are conditioned on H (if no user is detected then anonymity is not an issue). Essentially a_i denotes A_i and o_i denotes D_i, so equation (6.1) can now be written as $p_c(o_i|a_i) \leq \frac{1}{2}$ which can be generalized as a definition of probable innocence.

Definition 6.1.1 (RR-probable innocence)**.** *A finite anonymity system* $(\mathcal{A}, \mathcal{O}, p_c)$ *satisfies* RR-probable innocence *iff*

$$p_c(o|a) \leq \frac{1}{2} \quad \forall a \in \mathcal{A}, o \in \mathcal{O}$$

This is indeed an intuitive definition for Crowds. However there are many questions raised by this approach. For example, we are only interested in the





	o_1	o_2	$\cdots$	o_n
a_1	$\frac{c}{m-p_f}$	l	$\cdots$	l
a_2	0	n-1 Crowd		
$\vdots$	$\vdots$			
a_n	0			

	o_1	o_2	o_3
a_1	2/3	1/6	1/6
a_2	2/3	1/6	1/6
a_3	2/3	1/6	1/6

Figure 6.1: Examples of arbitrary (non symmetric) protocols. The value at position i, j represents $p_c(o_j|a_i)$ for user a_i and observable o_j.

probability of some events, what about other events that might reveal the identity of the initiator? For example the event $\neg o$ will have probability at least $1/2$, is this important? In fact, that's the reason we stated this definition only for finite systems, since we cannot ask the probability of all events to be less that $1/2$ (for example $\mathcal{O}$ is itself an event with probability always 1). Moreover, suppose in the example of Crowds that the probability of o_i under a different user j is negligible. Then, if we observe o_i, isn't it more probable that user i sent the message, even if $p_c(o_i|a_i)$ is less than $1/2$?

If we consider arbitrary protocols, then there are cases where Definition 6.1.1 does not express the expected properties of probable innocence. We give two examples of such systems in Figure 6.1 and we explain them below.

Example 1 On the left-hand side of Figure 6.1, m users are participating in a Crowds-like protocol. The only difference, with respect to the standard Crowds, is that user 1 is behind a firewall, which means that he can send messages to any other user but he cannot receive messages from any of them. In the corresponding table we give the conditional probabilities $p_c(o_j|a_i)$ for the honest users, where we recall that o_j means that j is the user who sends the message to the corrupted member, and a_i means that i is the initiator. When user 1 is the initiator there is a c/m chance that he sends the message to a corrupted user directly and there is also a chance that he forwards it to himself and sends it to a corrupted user in the next round. So $p_c(o_1|a_1) = \frac{c}{m} + \frac{1}{m} p_f \, p_c(o_1|a_1)$ which gives $p_c(o_1|a_1) = \frac{c}{m-p_f}$. All other users can be observed with the same probability $l = \frac{1}{n-1}(1 - \frac{c}{m-p_f})$. When any other user is the initiator, however, the probability of observing user 1 is 0, since the latter will never receive the message. In fact, the protocol will behave exactly like a Crowd of $n-1$ honest users as is shown in the table.

Note that Reiter and Rubin's definition (Def. 6.1.1) requires all values of this table to be at most $1/2$. In this example the definition holds provided

that $m - 1 \geq \frac{p_f}{p_f - 1/2}(c + 1)$, since the $(n - 1) \times (n - 1)$ sub-matrix is the same as in the original Crowds (which satisfies the definition) and the first row also satisfies it. However, if a corrupted member observes user 1 he can be sure that he is the initiator since no other initiator leads to the observation of user 1. The problem here is that Reiter and Rubin's definition constrains only the probability of detection of user 1 and says nothing about the attacker's confidence in case of detection. We believe that totally revealing the identity of the initiator with non-negligible probability is undesirable and should be considered as a violation of an anonymity notion such as probable innocence.

Example 2 On the right-hand side we have an opposite counter-example. Three users want to communicate with a web server, but they can only access it through a proxy. We suppose that all users are honest but they do not trust the proxy so they do not want to reveal their identity to him. So they use the following protocol: the initiator first forwards the message to one of the users 1, 2 and 3 with probabilities $2/3, 1/6$ and $1/6$ respectively, regardless of which is the initiator. The user who receives the message forwards it to the proxy. The probabilities of observing each user are shown in the corresponding table. Regardless of which is the initiator, user 1 will be observed with probability $2/3$ and the others with probability $1/6$ each.

In this example Reiter and Rubin's definition does not hold since $p_c(o_1|a_1) > 1/2$. However all users produce the same observables with the same probabilities hence we cannot distinguish between them. Indeed the system is strongly anonymous (Def. 5.1.1 holds)! Thus, in the general case, we cannot adopt Def. 6.1.1 as the definition of probable innocence since we want such a notion to be implied by strong anonymity.

However, it should be noted that in the case of Crowds the definition of Reiter and Rubin is correct, because of a special symmetry property of the protocol. This is discussed in detail in Section 6.3.

Finally, note that the above definition does not mention the probability of the users to be the originator. It only considers such events as conditions in the conditional probability of the event o_i *given that* i *is the originator*. The value of such conditional probability does not imply anything for the user, he might have a very small or very big probability of initiating the message. This is a major difference with respect to the next approach.

6.1.2 Second approach (limit on the attacker's confidence)

Halpern and O'Neill propose in [HO03] a general framework for defining anonymity properties. We give a very abstract idea of this framework, detailed information is available in [HO03]. In this framework a system consists of a group of agents, each having a local state at each point of the execution. The local state contains all information that the user may have and does not need to be explicitly defined. Each point in the execution of the system is represented by a tuple (r, m) where r is a function from time to global states and m is the current time. At each point (r, m) user i can only have access to his local state $r_i(m)$. So he does not know the actual point (r, m) but at least he knows that it must be a point (r', m') such that $r'_i(m') = r_i(m)$. Let $K_i(r, m)$ be the set of all these points. If a formula ϕ is true in all points of $K_i(r, m)$ then we say

that i knows ϕ. In the probabilistic setting it is possible to create a measure on $K_i(r, m)$ and draw conclusions of the form "formula ϕ is true with probability p".

To define probable innocence Halpern and O'Neill first define a formula $\theta(i, a)$ meaning "user i performed the event a". We then say that a system has probable innocence if for all points (r, m), the probability of $\theta(i, a)$ at this point for all users j (that is, the probability that arises by measuring $K_j(r, m)$) is at most one half.

This definition can be expressed in the framework of Chapter 4. The probability of a formula ϕ for user j at the point (r, m) depends only on the set $K_j(r, m)$ which itself depends only on $r_j(m)$. The last is the local state of the user, that is the only thing that he can observe. In our framework this corresponds to the observable events. Thus, we can reformulate the definition of Halpern and O'Neill as follows.

Definition 6.1.2 (HO-probable innocence)**.** *An anonymity instance* $(\mathcal{A}, \mathcal{O}, \mathcal{F}_o, P_c, P_{\mathcal{A}})$ *satisfies* HO-probable innocence *iff*

$$P([a]|[O]) \leq \frac{1}{2} \quad \forall a \in \mathcal{A}, O \in \mathcal{F}_o$$

where P is the probability measure induced by the instance.

Although this definition appears to be similar to the one of Reiter and Rubin, it's quite different. It requires that the probability of any anonymous event, given any observation, should be at most one half. Intuitively, this would mean that the attacker is not confident enough about which anonymous event occurred. However, in contrast to RR-probable innocence, this definition doesn't constrain the probability of the observable event itself.

The problem with this definition is that the probabilities of the anonymous events are not part of the system and we can make no assumptions about them. In fact, this is the reason that we had to define HO-probable innocence on an anonymity *instance* (which contains also a distribution $P_{\mathcal{A}}$ over $\mathcal{A}$) and not on an anonymity *system*, the probability $P([a]|[O])$ could not be defined on the latter. Moreover, HO-probable innocence cannot hold for an arbitrary user distribution. Consider for example the case where we know that user i visits very often a specific web site, so even if we have 100 users, the probability that he performed a request to this site is 0.99. Then we cannot expect this probability to become less than one half under all observations. This is why we didn't quantify over all distributions $P_{\mathcal{A}}$ as we did in the definition of conditional anonymity (Def. 5.1.2). A similar remark led Halpern and O'Neill to define conditional anonymity. If a user i has higher probability of performing the action than user j then we cannot expect this to change because of the system. Instead we can request that the system does not provide any new information about the originator of the action.

6.2 A new definition of probable innocence

In this section we propose a new notion of probable innocence that combines the two existing ones presented in previous section. Definition 6.2.2 extends Reiter and Rubin's definition while preserving its spirit, which is to constrain the

probability of detection of a user. Definition 6.2.1 follows the spirit of Halpern and O'Neill's definition, which is to constrain the attacker's confidence. Our notion is based on Definition 6.2.2 and it combines both spirits in the sense that it turns out to be equivalent to Definition 6.2.1. Moreover it overcomes the shortcomings discussed in previous section, namely, it does not depend on the symmetry of the system and it does not depend on the users' probabilities. We also show that our notion is a generalization of the existing ones since it can be reduced to the first under the assumption of symmetry, and to the second under the assumption of uniform users' probability.

Let $(\mathcal{A}, \mathcal{O}, \mathcal{F}_o, P_c)$ be an anonymity system. For a given distribution $P_{\mathcal{A}}$ on $\mathcal{A}$ we denote by P the measure induced by the anonymity instance $(\mathcal{A}, \mathcal{O}, \mathcal{F}_o, P_c, P_a)$. For simplicity we will write $P(a), P(O), \ldots$ for $P([a]), P([O]), \ldots$ respectively.

In general we would like our anonymity definitions to quantify over all possible distributions $P_{\mathcal{A}}$ since we should not assume anything about the probabilities of the anonymous events. Thus, Halpern and O'Neill's definition should be written: $\forall P_{\mathcal{A}} \forall a \forall O : P(a|O) \leq 1/2$ which makes it even clearer that it cannot hold for all $P_{\mathcal{A}}$, for example if we take $P_{\mathcal{A}}(a)$ to be very close to 1. On the other hand, Reiter and Rubin's definition contains only probabilities of the form $P(O|a)$ which are independent from $P_{\mathcal{A}}$.

In [HO03], where they define conditional anonymity, Halpern and O'Neill make the following remark about conditional anonymity. Since the probability that a user performs the action of interest is generally unknown, we cannot expect that all users appear with the same probability. All that we can ensure is that the system does not reveal any information, that is that the probability of every user before and after making an observation should be the same. In other words, the fraction between the probabilities of any pair of users should not be 1, but should at least remain the same before and after the observation.

We apply the same idea to probable innocence. We start by rewriting Definition (6.1.2) as

$$1 \geq \frac{P(a|O)}{P(\neg a|O)} \quad \forall a \in \mathcal{A}, \forall O \in \mathcal{F}_o \tag{6.2}$$

As we already explained, if $P_{\mathcal{A}}(a)$ is very high then we cannot expect this fraction to be less than 1. Instead, we could require that it does not surpass the corresponding fraction of the probabilities before the execution of the protocol. So we generalize condition (6.2) in the following definition.

Definition 6.2.1 (Probable innocence 1)**.** *Let $(\mathcal{A}, \mathcal{O}, \mathcal{F}_o, P_c)$ be an anonymity system where $\mathcal{A}$ is finite and let $n = |\mathcal{A}|$. The system satisfies* probable innocence *if for all distributions $P_{\mathcal{A}}$ over $\mathcal{A}$ for all $a \in \mathcal{A}$ and for all $O \in \mathcal{F}_o$ such that $P(O) > 0$, the following holds:*

$$(n-1)\frac{P(a)}{P(\neg a)} \geq \frac{P(a|O)}{P(\neg a|O)}$$

In probable innocence we consider the probability of a user to perform the action of interest compared to the probability of all the other users together. Definition 6.2.1 requires that the fraction of these probabilities after the execution of the protocol should be no bigger than $n-1$ times the same fraction

before the execution. The $n-1$ factor comes from the fact that in probable innocence *some* information about the sender's identity is leaked. For example, if users are uniformly distributed, each of them has probability $1/n$ before the protocol and the sender could appear with probability $1/2$ afterwords. In this case, the fraction between the sender and all other users is $\frac{1}{n-1}$ before the protocol and becomes 1 after. Definition 6.2.1 states that this fraction can be increased, thus leaking some information, but no more than $n-1$ times its original value.

Definition 6.2.1 generalizes Definition 6.1.2 and can be applied in cases where the distribution of the anonymous events is not uniform. However it still involves the probabilities of the anonymous events, which are not part of the anonymity system. We would like a definition similar to the one of strong anonymity (Def. 5.1.1) which involves only conditional probabilities of observable events. To achieve this we rewrite Definition 6.2.1 using the following transformations. In the following sums we take only events $a' \in \mathcal{A}$ such that $P(a') > 0$, this condition is omitted from the sums to simplify the notation.

$$\begin{aligned}
(n-1)\frac{P(a_i)}{P(\bigcup_{a'\neq a} a')} &\geq \frac{P(a|O)}{P(\bigcup_{a'\neq a} a'|O)} \Leftrightarrow \\
(n-1)\frac{P(a)}{\sum_{a'\neq a} P(a')} &\geq \frac{P(a|O)}{\sum_{a'\neq a} P(a'|O)} \Leftrightarrow \\
(n-1)\frac{P(a)}{\sum_{a'\neq a} P(a')} &\geq \frac{\frac{P(O|a)P(a)}{P(O)}}{\sum_{a'\neq a} \frac{P(O|a')P(a')}{P(O)}} \Leftrightarrow \\
(n-1)\sum_{a'\neq a} P(O|a')P(a') &\geq P(O|a)\sum_{a'\neq a} P(a') \qquad (6.3)
\end{aligned}$$

We obtain a lower bound of the left clause by replacing all $P(O|a')$ with their minimum. So we require that

$$\begin{aligned}
(n-1)\min_{a'\neq a}\{P(O|a')\}\sum_{a'\neq a} P(a') &\geq P(O|a)\sum_{a'\neq a} P(a') \Leftrightarrow \\
(n-1)\min_{a'\neq a} P(O|a') &\geq P(O|a) \qquad (6.4)
\end{aligned}$$

Condition (6.4) can be interpreted as follows: given any observable event, the probability of an anonymous event a should be balanced by the corresponding probabilities of the other anonymous events. It would be more natural to have the sum $\sum_{a'\neq a} P(O|a')$ at the left side, in fact the left side of (6.4) is a lower bound of this sum. However, since the distribution of the anonymous events is unknown, we have to consider the "worst" case where the event a' with the minimum $P(O|a')$ has the greatest probability of occurring.

Finally, condition (6.4) is equivalent to the following definition that we propose as a general definition of probable innocence.

Definition 6.2.2 (Probable innocence 2)**.** *Let $(\mathcal{A}, \mathcal{O}, \mathcal{F}_o, P_c)$ be an anonymity system where $\mathcal{A}$ is finite and let $n = |\mathcal{A}|$. The system satisfies probable innocence iff*

$$(n-1)P_c(O|a') \geq P_c(O|a) \quad \forall a, a' \in \mathcal{A}\ \forall O \in \mathcal{F}_o$$

For a finite anonymity system $(\mathcal{A}, \mathcal{O}, p_c)$ this definition can be written $p(n-1)p_c(o|a) \geq p_c(o|a')$ for all $o \in \mathcal{O}, a, a' \in \mathcal{A}$.

The meaning of this definition is that in order for $P(a)/P(\neg a)$ to increase at most by $n-1$ times the corresponding fraction between the probabilities of the observables must be at most $n-1$. Note that in strong anonymity $P_c(O|a)$ and $P_c(O|a')$ are required to be equal. In probable innocence we allow the first to be bigger, thus losing some anonymity, but not arbitrarily bigger. It still has to be smaller than $n-1$ times the corresponding probability of any other anonymous event.

This definition has the advantage of including only the probabilities of the system and not those of the anonymous events, similar to the definition of strong anonymity. It is clear that Definition 6.2.2 implies Definition 6.2.1 since we strengthened the first to obtain the second. Since Definition 6.2.1 considers all possible distributions of the users, the inverse implication also holds.

Theorem 6.2.3. *Definitions 6.2.1 and 6.2.2 are equivalent.*

Proof. Def. 6.2.2 $\Rightarrow$ Def. 6.2.1 is trivial, since we strengthen the second to obtain the first. For the inverse suppose that Def. 6.2.1 holds but Def. 6.2.2 does not, so there exist $a_k, a_l \in \mathcal{A}$ and $O \in \mathcal{F}_o$ such that $(n-1)P_c(O|a_k) < P_c(O|a_l)$. Thus there exists $\epsilon > 0$ s.t.

$$(n-1)(P(O|a_k) + \epsilon) \leq P(O|a_l) \tag{6.5}$$

Def. 6.2.1 should hold for all distributions $P_{\mathcal{A}}$ over $\mathcal{A}$ so we select one which assigns a very small probability to all anonymous event except a_k, a_l, that is $P_{\mathcal{A}}(a_i) = \frac{\delta}{n-2}\ \forall i \neq k, l$. We start from (6.3) which is a transformed version of Def. 6.2.1, and for $a = a_l$ we have:

$$\begin{aligned}
(n-1)\big(P(a_k)P(O|a_k) + & \\
\sum_{a' \neq a_k, a_l} \tfrac{\delta}{n-2} P(O|a')\big) \quad &\geq \quad P(O|a_l)(\delta + P(a_k)) \overset{P(O|a') \leq 1}{\Rightarrow} \\
(n-1)(P(a_k)P(O|a_k) + \delta) \quad &\geq \quad P(O|a_l)(\delta + P(a_k)) \overset{(6.5)}{\Rightarrow} \\
P(a_k)P(O|a_k) + \delta \quad &\geq \quad (P(O|a_k) + \epsilon)(\delta + P(a_k)) \Rightarrow \\
\delta(1 - P(O|a_k) - \epsilon) \quad &\geq \quad \epsilon P(a_k) \overset{(6.5)}{\Rightarrow} \\
\delta \quad &\geq \quad \frac{\epsilon P(a_k)}{1 - \frac{P(O|a_l)}{n-1}}
\end{aligned} \tag{6.6}$$

If $n > 2$ then the right side of inequality (6.6) is strictly positive so it is sufficient to take a smaller δ and we end up with a contradiction. If $n = 2$ then there are no other anonymous events except a_k, a_l and we can proceed similarly. □

Examples Recall now the two examples of Figure 6.1. If we apply Definition 6.2.2 to the first one we see that it does not hold since $(n-1)p_c(o_1|a_2) = 0 \not\geq \frac{c}{n-p_f} = p_c(o_1|a_1)$. This agrees with our intuition of probable innocence being violated when user 1 is observed. In the second example the definition holds since $\forall i, j \forall o : p_c(o|a_i) = p_c(o|a_j)$. Thus, we see that in these two examples our definition reflects correctly the notion of probable innocence.

6.3 Relation to other definitions

6.3.1 Definition by Reiter and Rubin

Reiter and Rubin's definition (Def. 6.1.1) considers the probabilities of the observables and it requires that given any anonymous event, the probability of any observable should be at most 1/2. As we saw with the examples of Figure 6.1 what is important is not the actual probability of an observable given a specific anonymous event, but its relation to the corresponding probabilities under all other anonymous events.

However in Crowds there are some important symmetries. First of all the number of the observables is the same as the number of anonymous events (honest users) . For each user i there is an observable o_i meaning that the user i is observed. When i is the initiator, o_i has a clearly higher probability than the other observables. However, since forwarders are randomly selected, the probability of o_j is the same for all $j \neq i$. The same holds for the observables. o_i is more likely to happen when i is the initiator. However all other users $j \neq i$ have the same probability of producing it. These symmetries can be expressed as $|\mathcal{A}| = |\mathcal{O}| = n$ and:

$$\begin{aligned} &\forall i, k, l \in \{1 \dots n\},\ k, l \neq i : \\ &\quad p_c(o_k|a_i) = p_c(o_l|a_i) && (6.7) \\ &\quad p_c(o_i|a_k) = p_c(o_i|a_l) && (6.8) \end{aligned}$$

Because of these symmetries, we cannot have situations similar to the ones of Figure 6.1. On the left-hand side, for example, the probability $p_c(o_1|a_2) = 0$ should be the same as $p_c(o_3|a_2)$. To keep the value 0 (which is the reason why probable innocence is not satisfied) we should have 0 everywhere in the row (except $p_c(o_2|a_2)$) which is impossible since the sum of the row should be 1/2 and $p_c(o_2|a_2) \leq 1/2$.

So the reason why probable innocence is satisfied in Crowds is not the fact that observing the initiator has low probability (what condition (6.1) ensures) by itself, but the fact that condition (6.1), because of the symmetry, forces the probability of observing any of the other users to be high enough.

Note that the number of anonymous users n is not the same as the number of users m in Crowds, in fact $n = m - c$ where c is the number of corrupted users.

Proposition 6.3.1. *For a finite anonymity systems and under the symmetry requirements (6.7) and (6.8), Definition 6.2.2 is equivalent to RR-probable innocence.*

Proof. Due to the symmetry we show that there are only two distinct values for $p_c(o_i|a_j)$. Let $p_c(o_1|a_1) = \phi$. Then $p_c(o_j|a_1), j > 1$ are all equal because of (6.7) and let $p_c(o_2|a_1) = \ldots = p_c(o_n|a_1) = \chi = \frac{1}{n-1}(1-\phi)$. Then for the second row we have $p_c(o_n|a_2) = p_c(o_n|a_1) = \chi$ from (6.8) so $p_c(o_j|a_2) = \chi, j \neq 2$ and $p_c(o_2|a_2) = 1 - (n-1)\chi = \phi$. Similarly for all the rows, so finally

$$p_c(o_i|a_j) = \begin{cases} \phi & \text{if } i = j \\ \chi & \text{if } i \neq j \end{cases}$$

Note that $\phi + (n-1)\chi = 1$. Assuming Def. 6.2.2 holds, we have

$$\begin{aligned} p_c(o_i|a_i) &\leq (n-1)p_c(o_i|a_j) \Rightarrow \\ \phi &\leq (n-1)\chi \Rightarrow \\ \phi &\leq 1-\phi \Rightarrow \\ p_c(o_i|a_i) &\leq \frac{1}{2} \end{aligned}$$

Also, for $n \geq 3$ we have

$$p_c(o_j|a_i) = \frac{1 - p_c(o_i|a_i)}{n-1} \leq \frac{1}{2} \quad j \neq i$$

so Reiter and Rubin's definition is satisfied. If $n = 2$ then we have $p_c(o_1|a_1) = p_c(o_2|a_2) = 1/2$ so RR-probable innocence is again satisfied. Note that for $n = 1$ none of the definitions can be satisfied. Similarly for the other direction. □

6.3.2 Definition of Halpern and O'Neill

One of the principles of our approach to the definition of anonymity is that it should not depend on the probabilities of the anonymous events. The notion of probable innocence we have proposed satisfies this principle, while the notion proposed by Halpern and O'Neill does not, hence the two notions are different. However, we will show that, if we assume a uniform distribution, then the two notions coincide.

Proposition 6.3.2. *If we restrict Definition 6.2.1 to consider only a uniform distribution on $\mathcal{A}$, that is a distribution P_u s.t. $P_u(a) = 1/|\mathcal{A}|$, $a \in \mathcal{A}$, then it becomes equivalent to HO-probable innocence.*

Proof. Trivial. Since all anonymous events have the same probability then the left side of definition 6.2.1 is equal to 1. □

Note that the equivalence of Def. 6.2.1 and Def. 6.2.2 is based on the fact that the former ranges over all possible distributions $P_{\mathcal{A}}$. Thus Def. 6.2.2 is strictly stronger than the one of Halpern and O'Neill.

6.3.3 Strong anonymity

Since strong anonymity is a stronger notion that probable innocence, we would expect the former to imply the latter. This is indeed true.

Proposition 6.3.3. *Strong anonymity implies probable innocence.*

Proof. Trivial. If Definition 5.1.1 holds then $P_c(O|a) = P_c(O|a')$ for all $a, a' \in \mathcal{A}$ and $O \in \mathcal{F}_o$. □

The relation between the various definitions of anonymity is summarized in Figure 6.2. The classification in columns is based on the type of probabilities that are considered. The first column considers the probability of different anonymous events, the second the probability of the same user before and after an observation and the third the probability of the observables. Concerning

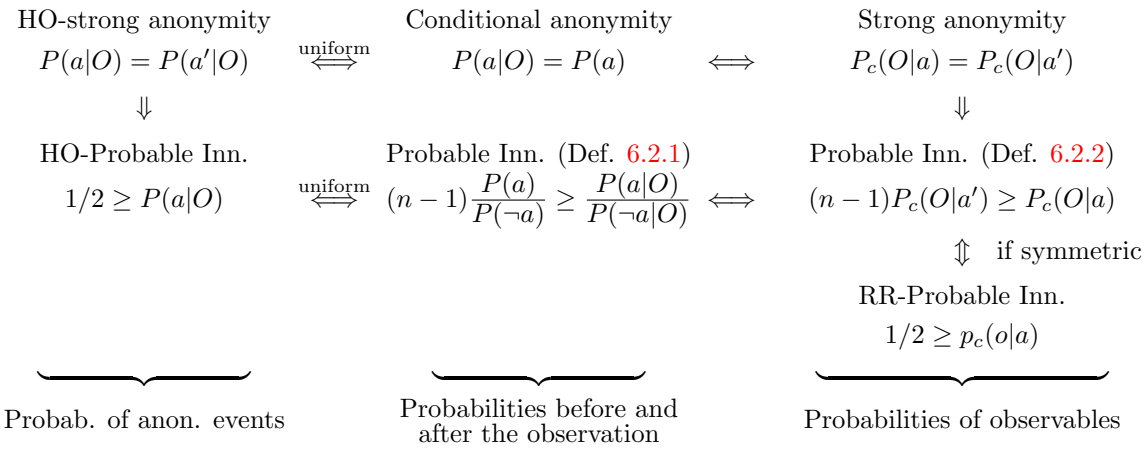


Figure 6.2: Relation between the various anonymity definitions

the rows, the first corresponds to the strong case and the second to probable innocence. It is clear from the table that the new definition is to probable innocence as conditional anonymity is to HO-strong anonymity.

6.4 Protocol composition

In this section we consider the case of protocol composition, described in Section 4.2, and we examine the anonymity guarantees of the resulting protocol with respect to the composed ones. We have already shown in Section 5.3 that the composition of two strongly anonymous systems is also strongly anonymous. This conforms to the intuition that a strongly anonymous protocol leaks no information at all. On the other hand, a protocol satisfying probable innocence is allowed to leak information to some extent. Now consider two systems S_1, S_2 where S_1 is strongly anonymous and S_2 satisfies probable innocence. Intuitively, since S_1 leaks no information we would expect the composed system $S_1; S_2$ to leak as much information as S_2, so it should also satisfy probable innocence.

Proposition 6.4.1. *Let $S_1 = (\mathcal{A}, \mathcal{O}_1, \mathcal{F}_{o1}, P_{c_1}), S_2 = (\mathcal{A}, \mathcal{O}_2, \mathcal{F}_{o2}, P_{c_2})$ be two anonymity systems such that S_1 is strongly anonymous and S_2 satisfies probable innocence. The protocol $S_1; S_2 = (\mathcal{A}, \mathcal{O}, \mathcal{F}_o, P_c)$ also satisfies probable innocence.*

Proof. From the anonymity hypotheses of S_1, S_2 we have for all $a, a' \in \mathcal{A}$:

$$P_{c_1}(O_1|a) = P_{c_1}(O_1|a') \quad \forall O_1 \in \mathcal{F}_{o1} \tag{6.9}$$

$$(n-1)P_{c_2}(O_2|a) = P_{c_2}(O_2|a') \quad \forall O_2 \in \mathcal{F}_{o2} \tag{6.10}$$

So for each observable $O \in \mathcal{F}$ of $S_1; S_2$ we have for all $a, a' \in \mathcal{A}$:

$$\begin{aligned}(n-1)P_c(O|a) &= (n-1)P_{c_1}(\text{proj}_1(O)|a)\, P_{c_2}(\text{proj}_2(O)|a) \\ &= (n-1)P_{c_1}(\text{proj}_1(O)|a')\, P_{c_2}(\text{proj}_2(O)|a) \qquad (6.9) \\ &\geq (n-1)P_{c_1}(\text{proj}_1(O)|a')\, P_{c_2}(\text{proj}_2(O)|a') \qquad (6.10) \\ &= P_c(O|a')\end{aligned}$$

□

However, if both S_1, S_2 satisfy probable innocence, $S_1; S_2$ does not necessarily satisfy it since the leak of both systems together might be bigger than probable innocence allows. We demonstrate this issue in the case of the n-repetition S^n of a protocol S. We examine its anonymity guarantees compared to those of S, obtaining a general result for a class of attacks that appear in protocols such as Crowds.

Consider a finite system with three anonymous events and one observable o with probabilities $p_c(o|a_1) = 1/2$ and $p_c(o|a_2) = p_c(o|a_3) = 1/4$. This system satisfies Definition 6.2.2 thus it provides probable innocence. If we repeat the protocol two times then the probabilities for the event oo will be $p_c(oo|a_1) = 1/4$ and $p_c(oo|a_2) = p_c(oo|a_3) = 1/16$, but now Definition 6.2.2 is violated. In the original protocol the probability of o under a_1 was two times bigger than the corresponding probability under the other anonymous events, but after the repetition it became 4 times bigger and Definition 6.2.2 does not allow this.

In the general case let $(\mathcal{A}, \mathcal{F}, \mathcal{F}_o, P_c)$ be an anonymity system. S^m satisfies probable innocence if (by definition) for all $O \in {\mathcal{F}_o}^n$, $a, a' \in \mathcal{A}$

$$\begin{aligned}(n-1)P(O|a) &\geq P(O|a') \Rightarrow \\ (n-1)\prod_{i=1}^{n} P_c(\text{proj}_i(O)|a) &\geq \prod_{i=1}^{n} P_c(\text{proj}_i(O)|a') \qquad (6.11)\end{aligned}$$

The following lemma states that it is sufficient to check only the events of the form $(O, \ldots, O)$ (the same observable event repeated m times), and expresses the probable innocence of S^n using probabilities of events of S.

Lemma 6.4.2. *Let $S = (\mathcal{A}, \mathcal{F}, \mathcal{F}_o, P_c)$ be an anonymity system. S^n satisfies probable innocence if and only if:*

$$(n-1)P_c^n(O_s|a) \geq P_c^n(O_s|a') \quad \forall O_s \in \mathcal{F}_o,\ \forall a, a' \in \mathcal{A} \qquad (6.12)$$

Proof. **(only if)** We can use equation (6.11) with $O = (O_s, \ldots, O_s)$ and the definition of proj_i to obtain (6.12).
(if) We can write (6.12) as $\sqrt[n]{n-1}P_c(O_s|a) \geq P_c(O_s|a')$. Let $O \in {\mathcal{F}_o}^n$ be an observable event of S^n. Since $\text{proj}_i(O) \in \mathcal{F}_o$, by applying this inequality to all $\text{proj}_i(O)$ we have:

$$\begin{aligned}\sqrt[n]{n-1}P_c(\text{proj}_1(O)|a) &\geq P_c(\text{proj}_1(O)|a') \\ &\vdots \\ \sqrt[n]{n-1}P_c(\text{proj}_n(O)|a) &\geq P_c(\text{proj}_n(O)|a')\end{aligned}$$

Then by multiplying these inequalities we obtain (6.11). □

Lemma 6.4.2 explains our previous example. The probability $p_c(o|a_2) = 1/4$ was smaller than $p_c(o|a_1) = 1/2$ but sufficient to provide probable innocence. But when we raised these probabilities to the power of two, 1/16 was too small so the event *oo* would expose a_1. In fact, if we allow an arbitrary number of repetitions equation (6.12) can never hold, unless the probability of all observable events under any anonymous event is the same, that is if the system is strongly anonymous.

Theorem 6.4.3. *Let $S = (\mathcal{A}, \mathcal{F}, \mathcal{F}_o, P_c)$ be an anonymity system. S^n satisfies probable innocence for all $n \geq 1$ if and only if S is strongly anonymous.*

Proof. **(if)** If S is strongly anonymous then by Corollary 5.3.2 S^n is also strongly anonymous so by Proposition 6.3.3 it satisfies probable innocence. **(only if)** Suppose S is not strongly anonymous so there exist $O \in \mathcal{F}_o$, $a, a' \in \mathcal{A}$ such that $P_c(O|a') > P_c(O|a)$. Assuming $P_c(O|a) > 0$ we rewrite equation (6.12) as:

$$n - 1 \geq \left(\frac{P_c(O|a')}{P_c(O|a)}\right)^n$$

but the condition above cannot hold for all n since $\alpha^n \to \infty$ when $n \to \infty$ for $\alpha > 1$. If $P_c(O|a) = 0$ it is easy to show that $P_c(O|a')$ must be also 0 which is again a contradiction. □

6.5 Application to anonymity protocols

6.5.1 Crowds

Reiter and Rubin have shown in [RR98] that Crowds satisfies RR-probable innocence if $m \geq \frac{p_f}{p_f - 1/2}(c + 1)$. Thus we already know that Crowds satisfies the new definition of probable innocence since Proposition 6.3.1 states that the new definition is equivalent to RR-probable innocence under a symmetry property that Crowds satisfies. In this section, we give an alternative proof of probable innocence using directly the new definition.

Consider an instance of Crowds with m users of which c are corrupted and let $n = m - c$. We assume that there is at least one corrupted user ($c \geq 1$). Similarly to the discussion in Section 6.1.1, let A_i denote that "user i is the originator" and D_i is the event "the user i was observed by a corrupted member (appears in the path right before the corrupter member)". Also let H denote the event "some user was detected by a corrupted member" where $p(H) > 0$ since we assumed $c \geq 1$. As already discussed in the proof of Proposition 6.3.1, due to the symmetry of Crowds, there are only two distinct values for $p(D_i|A_i)$ so let $p(D_i|A_i) = X$ and $p(D_j|A_i) = Y, i \neq j$.

We create a finite anonymity system $Crowds(m, c, p_f) = (\mathcal{A}, \mathcal{O}, p_c)$ as follows:

- $\mathcal{A} = \{a_1, \ldots, a_n\}$ is the set of honest crowd members
- $\mathcal{O} = \{o_1, \ldots, o_n\}$ where o_i means that the user i was detected by a corrupted user.
- $p_c(o_i|a_i) = \begin{cases} X/p(H) & \text{if } i = j \\ Y/p(H) & \text{otherwise} \end{cases}$

This system considers only the honest users (there is no anonymity requirement for corrupted users) under the assumption that a user is always detected, so all probabilities are conditioned on H (if no user is detected then then anonymity is not an issue). We now show that this system satisfies probable innocence.

Theorem 6.5.1. *The anonymity system $Crowds(m, c, p_f)$, $p_f \geq 1/2$, satisfies probable innocence if and only if $m \geq \frac{p_f}{p_f - 1/2}(c+1)$.*

Proof. Assume that user i is the initiator. In order for i to be detected a path must be formed in which i is right before a corrupted user. There are three possibilities for this path: either i forwards the message directly to the attacker, or he forwards it to himself and a sub-path starting from i is created, or he forwards the message to some honest user j and then a sub-path starting from j is created. Since users are selected uniformly, X can be computed as:

$$X = \tfrac{c}{m} + \tfrac{1}{m}\, p_f\, X + \tfrac{n-1}{m}\, p_f\, Y$$

reflecting the three possibilities for the path. Note that if he forwards the message to himself (probability $1/m$) then the probability to form a sub-path starting from i is $p_f X$ since the probability to keep forwarding in the next round is p_f. Similarly for Y:

$$Y = \tfrac{1}{m}\, p_f\, X + \tfrac{n-1}{m}\, p_f\, Y$$

Solving the above system of equations we get

$$\begin{aligned} X &= c\, \frac{1 - \frac{n-1}{m} p_f}{m - n p_f} \\ Y &= X - \frac{c}{m} \end{aligned}$$

And from the definition of probable innocence, assuming $n > 2$:

$$\begin{aligned} (n-1) p_c(o_j|a_i) \geq p_c(o_i|a_i) &\quad \Leftrightarrow \\ (n-1) Y \geq X &\quad \Leftrightarrow \\ (n-1)(X - \frac{c}{m}) \geq X &\quad \Leftrightarrow \\ X \geq \frac{c(n-1)}{m(n-2)} &\quad \Leftrightarrow \\ c\, \frac{1 - \frac{n-1}{m} p_f}{m - n p_f} \geq \frac{c(n-1)}{m(n-2)} &\quad \Leftrightarrow \\ 1 + \frac{p_f}{m - n p_f} \geq 1 + \frac{1}{n-2} &\quad \Leftrightarrow \\ 2(n-1) p_f \geq m &\quad \Leftrightarrow \\ m \geq \frac{p_f}{p_f - 1/2}(c+1) & \end{aligned}$$

For $n \leq 2$ the condition $(n-1)Y \geq X$ cannot be satisfied. □

As expected by the equivalence of the two definitions, we found the same condition as Reiter and Rubin.

Multiple paths attack

As stated in the original paper of Crowds, after creating a random path to a server, a user should use the same path for all the future requests to the same server. However there is a chance that some node in the path leaves the network, in which case the user has to create a new path using the same procedure. In theory the two paths cannot be linked together, that is the attacker cannot know that it is the same user who created the two paths. In practice, however, such a link could be achieved by means unrelated to the protocol such as the url of the server, the data of the request etc. By linking the two requests the attacker obtains more observables that he can use to track down the originator. Since the attacker also participates in the protocol he could voluntarily break existing paths that pass through him in order to force the users to recreate them.

If S is an anonymity system that models Crowds, then the n-paths version corresponds to the n-repetition of S, which repeats the protocol n times with the same user. From Proposition 6.4.3 and since Crowds is not strongly anonymous, we have that probable innocence cannot be satisfied if we allow an arbitrary number of paths. Intuitively this is justified. Even if the attacker sees the event o_1 meaning that user 1 was detected it could be the case (with non-trivial probability) that user 2 was the real originator, he sent the message to user 1 and the latter sent it to the attacker. However, if there are ten paths and the attacker sees $(o_1, \ldots, o_1)$ (ten times) then it is much more unlikely that all of the ten times user 2 sent the message to user 1 and user 1 to the attacker. It appears much more likely that user 1 was indeed the originator.

This attack had been foreseen in the original paper of Crowds and further analysis was presented in [WALS02, Shm04]. However our result is more general since we prove that probable innocence is impossible for any protocol that allows "multiple paths", in other words that can be modeled as an n-repetition, unless the original protocol is strongly anonymous. Also our analysis is simpler since we did not need to calculate the actual probabilities of any observables in a specific protocol.

6.5.2 Dining cryptographers

The dining cryptographers protocol is usually connected to strong anonymity since it satisfies this property under the assumption of fair coins, as shown in Section 5.2. In practice, the users in a dining cryptographers protocol can use any common secret between pairs of users, instead of coins. These secrets would range over a set of possible values and the same analysis would prove that the protocol is strongly anonymous, assuming that the secrets are selected uniformly from their set of values. If the secrets' distribution is not uniform, or if the attacker can enforce a different distribution (which is not unrealistic in practice), then strong anonymity is immediately violated. This would correspond to having unfair coins in the original setting. However, if the bias of the coins is not very big then intuitively we would expect a weaker notion of anonymity, like probable innocence, to hold, giving at least some anonymity guarantees to the users. So it is interesting to find sufficient and necessary conditions to satisfy probable innocence, which is the topic of this section.

We consider again the analysis of Section 5.2, and we refer to the beginning

of that section for the notation.

We first give a sufficient condition for probable innocence for any graph G_c.

Theorem 6.5.2. *The anonymity system $S(G_c) = (\mathcal{A}, \mathcal{O}, p_c)$ corresponding to the connected component G_c satisfies probable innocence if*

$$p_i(v) \geq \frac{1}{1 + \sqrt[m]{n-1}} \qquad \forall i \in \{1, \ldots, m\}\ \forall v \in \{0,1\} \tag{6.13}$$

where $n = |\mathcal{A}|$, m is the number of edges (coins) of G_c and $p_i(0), p_i(1)$ are the probabilities of coin i giving head or tail respectively.

Proof. Fix an observable $\vec{o} \in \mathcal{O}$ and a cryptographer $a_i \in \mathcal{A}$. To compute the probability $p_c(\vec{o}|a_i)$ we have to compute all the possible coin configurations that will produce $\vec{o}$ as output. These will be given by the following system of linear equations in GF(2):

$$B\vec{x} = \vec{o} \oplus \vec{r_i}$$

Following the same reasoning as in the proof of Theorem 5.2.1, we derive that the system is solvable and has $2^{m-(n-1)}$ solutions. Let $\mathcal{X}(a_i, \vec{o})$ be the set of solutions for the specific cryptographer a_i and observable $\vec{o}$.

$\mathcal{X}(a_i, \vec{o})$ contains all the coin configurations that produce the output $\vec{o}$, thus the probability $p_c(\vec{o}|a_i)$ is

$$p_c(\vec{o}|a_i) = \sum_{\vec{x} \in \mathcal{X}(a_i, \vec{o})} \prod_{i=1}^{m} p_i(x_i)$$

To prove that probable innocence holds we need to show that for all $a_i, a_j \in \mathcal{A}$ and $\vec{o} \in \mathcal{O}$:

$$\begin{gathered}(n-1)p_c(\vec{o}|a_i) \geq p_c(\vec{o}|a_j) \Leftrightarrow \\ (n-1) \sum_{\vec{x} \in \mathcal{X}(a_i, \vec{o})} \prod_{i=1}^{m} p_i(x_i) \geq \sum_{\vec{x} \in \mathcal{X}(a_j, \vec{o})} \prod_{i=1}^{m} p_i(x_i)\end{gathered} \tag{6.14}$$

and for this it is sufficient that

$$(n-1)2^{m-(n-1)} \min_{\vec{x} \in \mathcal{X}(a_i, \vec{o})} \prod_{i=1}^{m} p_i(x_i) \geq 2^{m-(n-1)} \max_{\vec{x} \in \mathcal{X}(a_j, \vec{o})} \prod_{i=1}^{m} p_i(x_i) \Leftrightarrow \tag{6.15}$$

$$(n-1) \prod_{i=1}^{m} p_i(y_i) \geq \prod_{i=1}^{m} p_i(z_i) \tag{6.16}$$

where $\vec{y} \in \mathcal{X}(a_i, \vec{o}), \vec{z} \in \mathcal{X}(a_j, \vec{o})$ are the vectors than give the min and max values in the left and right-hand side of inequality (6.15) respectively. It remains to prove inequality (6.16). From (6.13) it is easy to show that $\sqrt[m]{n-1} p_i(v) \geq p_i(u)$ for all $v, u \in \{0,1\}$, so we have

$$\begin{aligned}(n-1) \prod_{i=1}^{m} p_i(y_i) &= \prod_{i=1}^{m} \sqrt[m]{n-1}\, p_i(y_i) \\ &\geq \prod_{i=1}^{m} p_i(z_i)\end{aligned}$$

□

Note that the above theorem gives a sufficient but not a necessary condition for probable innocence on arbitrary graphs, since the inequality between the sums in (6.14) could hold without satisfying inequality (6.15). So in certain types of graphs probable innocence could hold for even more biased coins than the ones allowed by the previous theorem. A sufficient and necessary condition is harder to obtain since we have to take into account all possible connection graphs.

In the rest of this section we obtain such conditions by restricting to specific types of graphs and to the case where all coins are identical.

Chain graphs A graph is a called a *chain* if it is connected, all vertexes have degree at most 2 and at least one vertex has degree 1. Such graphs have exactly $n-1$ edges, which is the minimum number of edges that a connected graph can have, so intuitively it offers the least anonymity protection. Indeed, we show that the condition of Theorem 6.5.2 is sufficient and necessary for chain graphs.

Theorem 6.5.3. *The anonymity system $S(G_c) = (\mathcal{A}, \mathcal{O}, p_c)$ where G_c is a chain graph satisfies probable innocence if and only if*

$$p(v) \geq \frac{1}{1 + \sqrt[n-1]{n-1}} \qquad \forall v \in \{0,1\} \tag{6.17}$$

where $n = |\mathcal{A}|$ and $p(0), p(1)$ are the probabilities of a coin giving head or tail respectively (the same for all coins).

Proof. The fact that condition (6.17) is sufficient is an application of Theorem 6.5.2 since $m = n-1$ on a chain graph. Now suppose that probable innocence holds and consider the observable $\vec{o} = (1, 0, \ldots, 0)$. Since $m = n-1$, for each user a_i there is only one solution to the system of equations $B\vec{x} = \vec{o} \oplus \vec{r_i}$. For a_1 (the first user of the chain) the only solution is $\vec{y} = (0, \ldots, 0)$ and for a_n (the last user of the chain) the only solution is $\vec{z} = (1, \ldots, 1)$. So

$$p_c(\vec{o}|a_1) = \prod_{i=1}^{n-1} p(y_i) = p^{n-1}(0)$$
$$p_c(\vec{o}|a_n) = \prod_{i=1}^{n-1} p(z_i) = p^{n-1}(1)$$

and since probable innocence holds we have

$$\begin{aligned}
(n-1)p_c(\vec{o}|a_1) \geq p_c(\vec{o}|a_n) &\Rightarrow \\
(n-1)p^{n-1}(0) \geq p^{n-1}(1) &\Rightarrow \\
\sqrt[n-1]{n-1}\, p(0) \geq 1 - p(0) &\Rightarrow \\
p(0) \geq \frac{1}{1 + \sqrt[n-1]{n-1}}&
\end{aligned}$$

and similarly for $p(1)$. □

Cycle graphs A graph is called a *cycle* if its vertices are connected in a circular fashion, that is there are n edges connecting vertices i and $i+1$, for $1 \leq i \leq n-1$ with an extra edge connecting vertices 1 and n. A cycle has one more edge than a chain so we would expect to offer better anonymity guarantees. We now give a more relaxed condition than the one of Theorem 6.5.2 that is both sufficient and necessary for cycle graphs.

Theorem 6.5.4. *The anonymity system $S(G_c) = (\mathcal{A}, \mathcal{O}, p_c)$ where G_c is a cycle graph satisfies probable innocence if*

$$p(v) \geq 1 - \frac{1}{1 + \left(n - 1 - \sqrt{n(n-2)}\right)^{\frac{2}{n}}} \qquad \forall v \in \{0,1\} \tag{6.18}$$

where $n = |\mathcal{A}|$ and $p(0), p(1)$ are the probabilities of a coin giving head or tail respectively (the same for all coins). If n is even then this is also a necessary condition.

Proof. We fix a user a_i and observable $\vec{o}$. Since $m = n$ there are exactly 2 solutions to the system of equations $B\vec{x} = \vec{o} \oplus \vec{r_i}$. Moreover, all vertices have exactly 2 adjacent edges so by symmetry inverting all coins doesn't affect the output, thus the solutions come in pairs $(\vec{x}, \vec{x} \oplus \vec{1})$. To simplify the notation, let $h = p(0), t = p(1)$. Let $\vec{x}$ be a solution of the system above, we have

$$p_c(\vec{o}|a_i) = \prod_{i=1}^{n} p(x_i) + \prod_{i=1}^{n} p(x_i \oplus 1) = h^k t^{n-k} + h^{n-k} t^k$$

where $k \geq n/2$ is the number of 0s in $\vec{x}$ (if $k < n/2$ we take $k' = n - k \geq n/2$ and we obtain the same form). Probable innocence requires that

$$\begin{gathered}(n-1) p_c(\vec{o}|a_i) \geq p_c(\vec{o}|a_j) \Leftrightarrow \\ (n-1)(h^k t^{n-k} + h^{n-k} t^k) \geq h^l t^{n-l} + h^{n-l} t^l\end{gathered}$$

where $k, l \geq n/2$ are the number of 0s in the solution of the system for a_i, a_j respectively. Without loss of generality we assume $h \geq t$ (the other case can be treated symmetrically) and let $\alpha = t/h \leq 1$. We divide both sides by h^n:

$$\begin{gathered}(n-1)(h^{k-n} t^{n-k} + h^{-k} t^k) \geq h^{l-n} t^{n-l} + h^{-l} t^l \Leftrightarrow \\ (n-1)(\alpha^k + \alpha^{n-k}) \geq \alpha^l + \alpha^{n-l}\end{gathered} \tag{6.19}$$

We can show that

$$\alpha^n + 1 \geq \alpha^{n-1} + \alpha \geq \ldots \geq \alpha^{n/2} + \alpha^{n/2} \tag{6.20}$$

so (6.19) is satisfied for any $k, l \in \{\frac{n}{2}, \ldots, n\}$ if and only if

$$\begin{gathered}(n-1)(\alpha^{n/2} + \alpha^{n/2}) \geq \alpha^n + 1 \Leftrightarrow \\ (n-1)\, 2\, z \geq z^2 + 1\end{gathered}$$

where $z = \alpha^{n/2}$. Solving $z^2 - 2(n-1) z + 1 = 0, z \leq 1$ we get $z_o = n - 1 - \sqrt{n(n-2)}$. So the inequality above holds iff $z \geq z_o$ that is $\alpha \geq z_o^{2/n}$. Finally

$$t = 1 - \frac{1}{1+\alpha} \geq 1 - \frac{1}{1 + z_o^{2/n}} = 1 - \frac{1}{1 + \left(n - 1 - \sqrt{n(n-2)}\right)^{\frac{2}{n}}}$$

and the same for h since $h \geq t$. If n is even then there will be some a_i, a_j such that k, l are exactly $n/2, n$ respectively, so condition (6.18) is necessary. If n is odd then (6.20) still holds so condition (6.18) is sufficient, even though probable innocence could hold with even more biased coins. □

Theorem 6.5.4 provides a more relaxed condition for probable innocence that the more general Theorem 6.5.2. For example, for a cycle of 4 vertices, the latter requires $p(0) \geq 0.43$ while the former requires $p(0) \geq 0.29$. So we see that even with strongly biased coins the dining cryptographers offers non-trivial anonymity guarantees, namely probable innocence.

It is also worth noting that since $m \geq n - 1$ (the graph is connected) the right-hand side in all the above conditions converges to $1/2$ as $n \to \infty$. This means that as the number of users increases, the condition for the dining cryptographers to satisfy probable innocence approximates the requirement of fairness of the coins, which is the condition for strong anonymity to be satisfied.

Part II

Information Theory and Hypothesis Testing

Seven

An information-theoretic definition of anonymity

In the previous chapters we have formalized two properties of anonymity systems, namely strong anonymity and probable innocence. The first offers "perfect" anonymity guarantees in the sense that it allows no information about the anonymous events to be leaked. The second is weaker, it allows the attacker to obtain some knowledge about the anonymous events but still allows a user to plead "not guilty" in the sense that it appears less probable that he performed the action of interest than that any of the other users together did. Strong anonymity is satisfied by the dining cryptographers protocol with fair coins while probable innocence is satisfied by Crowds, under a condition on the number of corrupted users, and by the dining cryptographers with biased coins, under a condition on the probability distribution of the coins.

These properties are very useful for the analysis of anonymity systems, however they have an important disadvantage: they are "black or white" in the sense than they can either be satisfied or violated by a particular protocol, but they do not provide an indication of "how much" they are satisfied or violated. Consider for example probable innocence in the case of the dining cryptographers. For 4 users the property could hold for $p(heads) \geq 0.29$, that is an instance with $p(heads) = 0.29$ and an instance with $p(heads) = 0.49$ both satisfy probable innocence, however intuitively the second offers much stronger anonymity than the first. The same happens for protocols with $p(heads) < 0.29$, they all violate probable innocence but clearly an instance with $p(heads) = 0$ is much worse than one with $p(heads) = 0.28$.

Due to this issue and since both protocols examined so far have parameters that affect their anonymity, it seems reasonable to search for a definition of anonymity that maps protocols to a continuous scale and which is sensitive to the value of these parameters. Such a definition would give us a better understanding of the behavior of anonymity protocols and would allow us to compare protocols of the same "family", for example protocols that both satisfy or violate probable innocence. Moreover, from an engineering point of view, such a definition would allow us to balance the trade-off between anonymity and other features of the protocols, such as performance or availability, and fine-tune the protocol's parameters to obtain the best overall result.

To obtain such a quantitative definition of anonymity, we consider a framework in which anonymity systems are interpreted as noisy channels in the information-theoretic sense, and we explore the idea of using the notion of capacity as a measure of the loss of anonymity. This idea was already suggested by Moskowitz, Newman and Syverson, in their analysis of the covert channel that can be created as a result of imperfect anonymity [MNCM03, MNS03].

Contribution The contribution of this chapter consists of the following:

- We define a more general notion of capacity, that we call *conditional capacity*, which models the case in which some loss of anonymity is allowed by design.
- We discuss how to compute capacity and conditional capacity when the anonymity protocol satisfies certain symmetries.
- We compare the new definition with various probabilistic notions of anonymity given in literature, in particular strong anonymity and probable innocence. Moreover, we show that the definition of probable innocence introduced in Chapter 6 corresponds to a certain information-theoretic bound.
- We use the new definition to compare different network configurations for the dining cryptographers protocol. More precisely we show if we add a new edge (coin) to any connection graph with arbitrary coins, the anonymity degree of the corresponding system increases or remains the same. Using this property, we give a stronger version of Chaum's result, namely we prove that to achieve strong anonymity in a connected component it suffices that the component have a spanning tree consisting of fair coins.
- We show how to compute the matrix of a protocol using model checking tools. We demonstrate our ideas in the dining cryptographers and crowds protocols, where we show how the parameters of each protocol affect its anonymity.

Plan of the chapter In Section 7.1 we justify our view of protocols as channels and (loss of) anonymity as capacity and conditional capacity, and we give a method to compute these quantities in special symmetry cases. In Section 7.3, we relate our framework to other probabilistic approaches to anonymity. In Section 7.4 we discuss the operation of adding a coin to a dining cryptographers system, and we prove the monotonicity of the degree of anonymity with respect to this operation, which allows us to strengthen Chaum's result. In Section 7.5, we illustrate with specific examples (the dining cryptographers and Crowds) how to compute the channel matrix and the degree of anonymity for a given protocol, possibly using automated tools. Finally, in section 7.6 we discuss related work.

7.1 Loss of Anonymity as Channel Capacity

Let $S = (\mathcal{A}, \mathcal{O}, p_c)$ be an anonymity system. S together with a distribution $p_{\mathcal{A}}$ on $\mathcal{A}$ define an anonymity instance and induce a discrete probability distribution p on $\mathcal{A} \times \mathcal{O}$ as $p(a, o) = p_{\mathcal{A}}(a)p_c(o|a)$. In the rest of this chapter we will use p to denote this induced distribution, where the anonymity system and the distribution $p_{\mathcal{A}}$ should be clear from the context. For simplicity we will use $p(a)$ for $p([a]) = p_{\mathcal{A}}(a)$ and $p(o|a)$ for $p([o]|[a]) = p_c(o|a)$. We define two random variables $A : \mathcal{A} \times \mathcal{O} \mapsto \mathcal{A}$ and $O : \mathcal{A} \times \mathcal{O} \mapsto \mathcal{O}$ as $A((a, o)) = a$ and $O((a, o)) = o$. Their probability mass functions will be $P([A = a]) = p(a)$ and $P([O = o])$ respectively.

We can now use tools from information theory to reason about the information that the adversary obtains from the protocol, these concepts were briefly presented in Section 2.2. The entropy $H(A)$ of A gives the amount of uncertainty about the anonymous events, before executing the protocol. The higher the entropy is the less certain we are about the outcome of A. After the execution, however, we also know the actual value of O. Thus, the conditional entropy $H(A|O)$ gives the uncertainty of the attacker about the anonymous events after performing the observation. To compare these two entropies, we consider the mutual information $I(A; O)$ which measures the information about A that is contained in O. This quantity is exactly what we want to minimize. In the best case it is 0, meaning that we can learn nothing about A by observing O (in other words $H(A|O)$ is equal to $H(A)$). In the worst case it is equal to $H(A)$ meaning that all the uncertainty about A is lost after the observation, thus we can completely deduce the value of A ($H(A|O)$ is 0).

To compute $I(A; O)$ we need the joint distribution $p(a, o)$ which depends on $p_{\mathcal{A}}(a)$ and $p_c(o|a)$. Similarly to strong anonymity and probable innocence, we want our definition to depend only on the conditional probabilities $p_c(o|a)$ and not on the distribution $p_{\mathcal{A}}$ of the anonymous events, since we only consider the former to be a characteristic of the protocol while the latter models the users' intentions during the execution. This view of the system in isolation from the users brings us to consider the protocol as a device that, given $a \in \mathcal{A}$ as input, it produces an output in $\mathcal{O}$ according to a probability distribution $p_c(\cdot|a)$. This concept is well investigated in information theory, where such a device is called a *channel*, and it is described by the matrix whose rows represent the elements of $\mathcal{A}$, the columns the elements of $\mathcal{O}$, and the value in position (a, o) is the conditional probability $p_c(o|a)$. An anonymity channel is shown in Figure 7.1. Note that this is not a "real" channel, in the sense that a is not data that is transmitted from a sender to a receiver, but a modeling tool to define our degree of anonymity. Since we are interested in the worst possible case, we adopt the definition of the *loss of anonymity* as the maximum value of $I(A; O)$ over all possible input distributions, that is the capacity of the corresponding channel.

Definition 7.1.1. *Let $S = (\mathcal{A}, \mathcal{O}, p_c)$ be an anonymity protocol. The* loss of anonymity *$C(S)$ of the protocol is defined as*

$$C(S) = \max_{p_{\mathcal{A}}(a)} I(A; O)$$

where the maximum is taken over all possible input distributions.

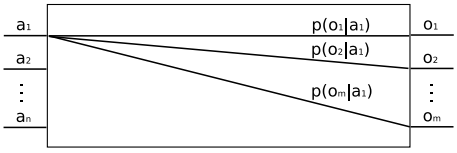


Figure 7.1: An anonymity channel

Figure 7.2: A simple elections protocol

The loss of anonymity measures the amount of information about A that can be learned by observing O in the worst possible distribution of anonymous events. If it is 0 then, no matter what is the distribution of A, the attacker can learn nothing more by observing the protocol. In fact, as we will see in section 7.3.1, this corresponds exactly to strong anonymity. However, as we discuss in section 7.3.3, our framework also captures weaker notions of anonymity.

As with entropy, channel capacity is measured in bits. Roughly speaking, 1 bit of capacity means that after the observation A will have one bit less of entropy, in another words the attacker will have reduced the set of possible anonymous events by a factor of 2, assuming a uniform distribution.

7.1.1 Relative Anonymity

So far, we have assumed that ideally no information about the anonymous events should be leaked. However, there are cases where *some* information about the anonymous events is allowed to be revealed by design, without this leak being considered a flaw of the protocol. Consider, for example, the case of a simple elections protocol, displayed in figure 7.2. For simplicity we assume that there are only two candidates c and d, and that each user always votes for one of them, so an anonymous event can be represented by the subset of users who voted for candidate c. In other words, $\mathcal{A} = 2^V$ where V is the set of voters. The output of the protocol is the list of votes of all users, however, in order to achieve anonymity, the list is randomly reordered, using for example some MIX technique[1]. As a consequence, the attacker can see the number of votes for each candidate, although he should not be able to find out who voted for whom. Indeed, determining the number of votes for candidate c (the cardinality of a), while concealing the vote expressed by each individual (the elements that constitute a), is the purpose of the protocol.

So it is clear that after the observation only a fraction of the anonymous events remains possible. Every event $a \in \mathcal{A}$ with $|a| \neq n$ where n is the number

[1]In MIX protocols an agent waits until it has received requests from multiple users and then forwards the requests in random order to hide the link between the sender and the receiver of each request.

of votes for candidate c can be ruled out. As a consequence $H(A|O)$ will be smaller than $H(A)$ and the capacity of the corresponding channel will be non-zero, meaning that some anonymity is lost. In addition, there might be a loss of anonymity due to other factors, for instance, if the reordering technique is not uniform. However, it is undesirable to confuse these two kinds of anonymity losses, since the first is by design and thus acceptable. We would like a notion of anonymity that factors out the *intended* loss and measures only the loss that we want to minimize.

Let $\mathcal{R}$ be a set of "revealed" values, that is suppose that in each execution exactly one value $r \in \mathcal{R}$ is revealed to the attacker. In the example of the elections protocol, the revealed value is the cardinality of a so $\mathcal{R} = \{0, \ldots, |V|\}$. Also let $p_{\mathcal{R}}(\cdot|a, o)$ be a collection of distributions on $\mathcal{R}$, for each $a \in \mathcal{A}, o \in \mathcal{O}$. Now given an anonymity system $(\mathcal{A}, \mathcal{O}, p_c)$ and a distribution $p_{\mathcal{A}}$ on $\mathcal{A}$ we can define a probability distribution p on $\mathcal{A} \times \mathcal{O} \times \mathcal{R}$ as

$$p(a, o, r) = p_{\mathcal{A}}(a)\, p_c(o|a)\, p_{\mathcal{R}}(r|o, a)$$

As usual we write $p(r)$ for $p([r])$ where $[r] = \mathcal{A} \times \mathcal{O} \times \{r\}$. We also define a random variable $R : \mathcal{A} \times \mathcal{O} \times \mathcal{R} \mapsto \mathcal{R}$ as $R(a, o, r) = r$, with probability mass function $P([R = r]) = p(r)$.

Then we use R to cope with the intended anonymity loss as follows. Since we allow the value of R to be revealed by design, we can consider that it is known even before executing the protocol. So, $H(A|R)$ gives the uncertainty about A given that we know R and $H(A|R, O)$ gives the uncertainty after the execution of the protocol, when we know both R and O. By comparing the two we retrieve the notion of *conditional mutual information* $I(A; O|R)$ defined as

$$I(A; O|R) = H(A|R) - H(A|R, O)$$

So, $I(A; O|R)$ is the amount of uncertainty on A that we lose by observing O, given that R is known. Now we can define the notion of *conditional capacity* $C|R$ which will give us the *relative loss of anonymity* of a protocol.

Definition 7.1.2. *Let $(\mathcal{A}, \mathcal{O}, p_c)$ be an anonymity system, $\mathcal{R}$ a set of revealed values and $p_R(\cdot|a, o)$ a collection of probability distributions on $\mathcal{R}$. The* relative loss of anonymity *of the protocol with respect to R is defined as*

$$C|R = \max_{p_{\mathcal{A}}} I(A; O|R)$$

where the maximum is taken over all possible input distributions.

Partitions: a special case of relative anonymity

An interesting special case of relative anonymity is when the knowledge of either an anonymous event or an observable event totally determines the value of R. In other words, both $\mathcal{A}$ and $\mathcal{O}$ are partitioned into subsets, one for each possible value of R. The elections protocol of the previous section is an example of this case. In this protocol, the value r of R is the number of votes for candidate c. This is totally determined by both anonymous events a (r is the cardinality of a) and observable events o (r is the number of c's in o). So we can partition $\mathcal{A}$ in subsets $\mathcal{A}_0, \ldots, \mathcal{A}_n$ such that $|a| = n$ for each $a \in \mathcal{A}_n$,

and similarly for $\mathcal{O}$. Notice that an anonymous event $a \in \mathcal{A}_i$ produces only observables in $\mathcal{O}_i$, and vice versa.

In this section we show that such systems can be viewed as the composition of smaller, independent sub-systems, one for each value of R.

We say that R is a deterministic function of X if $p(r|x)$ is 0 or 1 for all $r \in \mathcal{R}$ and $x \in \mathcal{X}$. In this case we can partition $\mathcal{X}$ as follows

$$\mathcal{X}_r = \{x \in \mathcal{X} \mid p(r|x) = 1\}$$

Clearly the above sets are disjoint and their union is $\mathcal{X}$.

Theorem 7.1.3. *Let $(\mathcal{A}, \mathcal{O}, p_c)$ be an anonymity system, $\mathcal{R}$ a set of revealed values and $p_R(\cdot|a, o)$ a collection of probability distributions on $\mathcal{R}$. If R is a deterministic function of both A and O, under some non-zero input distribution $p_{\mathcal{A}}$[2], then the transition matrix of the protocol is of the form*

	$\mathcal{O}_{r_1}$	$\mathcal{O}_{r_2}$	$\cdots$	$\mathcal{O}_{r_l}$
$\mathcal{A}_{r_1}$	M_{r_1}	0	$\ldots$	0
$\mathcal{A}_{r_2}$	0	M_{r_2}	$\ldots$	0
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\mathcal{A}_{r_l}$	0	0	$\ldots$	M_{r_l}

and

$$C|R \leq d \quad \Leftrightarrow \quad C_i \leq d, \forall i \in 1..l$$

where C_i is the capacity of the channel with matrix M_{r_i}.

Proof. First we show that the protocol matrix has the above form, that is $p(o|a) = 0$ if $a \in \mathcal{A}_r, o \in \mathcal{O}_{r'}$ with $r \neq r'$. If $p(o) = 0$ then (since $p_{\mathcal{A}}$ is non-zero) then whole column of o is zero and we are finished. Otherwise, since R is a deterministic function of A, O we have $p(r|a) = 1$ and $p(r|o) = 0$. Then (we use the $[\cdot]$ notation to make set operations clearer)

$$p([r] \cap [a]|o) = 0 \Rightarrow p([r] \cap [o]|a)\frac{p(a)}{p(o)} = 0 \Rightarrow p([r] \cap [o]|a) = 0$$

Finally

$$p([r] \cup [o]|a) = p(r|a) + p(o|a) - p([r] \cap [o]|a) = 1 + p(o|a)$$

so $p(o|a) = 0$ otherwise $p([r] \cup [o]|a)$ would be greater than 1.

Now we show that $C|R \leq d$ iff $C_i \leq d, \forall i \in 1..l$ where C_i is the capacity of the channel with matrix M_{r_i}, constructed by taking only the rows in $\mathcal{A}_{r_i}$ and the columns in $\mathcal{O}_{r_i}$.

($\Rightarrow$) Assume that $C|R \leq d$ but $\exists i : C_i > d$. Then there exists a distribution p_i over $\mathcal{A}_{r_i}$ such that $I(A_{r_i}; O_{r_i}) > d$ where A_{r_i}, O_{r_i} are the input and output random variables of channel M_{r_i}. We construct a distribution over $\mathcal{A}$ as follows

$$p(a) = \begin{cases} p_i(a) & \text{if } a \in \mathcal{A}_{r_i} \\ 0 & \text{otherwise} \end{cases}$$

[2]We require $p_{\mathcal{A}}$ to assign non-zero probability to all users so that $p(r|o)$ can be defined, unless the whole column is zero. Note that if R is a deterministic function of O under some non-zero distribution, it is also under all distributions.

It is easy to see that under this distribution, $I(A;O|R) = I(A_{r_i}|O_{r_i})$ which is a contradiction since $I(A;O|R) \leq C|R \leq d < I(A_{r_i}|O_{r_i})$.

($\Leftarrow$) The idea is that for each input distribution $p(a)$ we can construct an input distribution $p_r(a)$ for each sub-channel M_r and express $I(A;O|R)$ in terms of the mutual information of all sub-channels. We write $I(A;O|R)$ as:

$$\begin{aligned}
&I(A;O|R)\\
&= H(A|R) - H(A|R,O)\\
&= -\sum_{r\in\mathcal{R}} p(r) \sum_{a\in\mathcal{A}} p(a|r)\log p(a|r) + \sum_{\substack{r\in\mathcal{R}\\ o\in\mathcal{O}}} p(r,o) \sum_{a\in\mathcal{A}} p(a|r,o)\log p(a|r,o)\\
&= -\sum_{r\in\mathcal{R}} p(r)\Big[\sum_{a\in\mathcal{A}} p(a|r)\log p(a|r) - \sum_{o\in\mathcal{O}} p(o|r) \sum_{a\in\mathcal{A}} p(a|r,o)\log p(a|r,o)\Big]
\end{aligned}$$

Moreover, we have

$$p(a|r) = \begin{cases} \frac{p(a)}{p(r)} & \text{if } a \in \mathcal{A}_r\\ 0 & \text{otherwise}\end{cases}$$

$$p(o|r) = \begin{cases} \frac{p(o)}{p(r)} & \text{if } o \in \mathcal{O}_r\\ 0 & \text{otherwise}\end{cases}$$

Also $p(a|r,o) = p(a|o)$ if $o \in \mathcal{O}_r$ and $p(a|r,o) = 0$ if $a \notin \mathcal{A}_r$. Thus in the above sums the values that do not correspond to each r can be eliminated and the rest can be simplified as follows:

$$I(A;O|R) = -\sum_{r\in\mathcal{R}} p(r)\Big[\sum_{a\in\mathcal{A}_r} \frac{p(a)}{p(r)}\log\frac{p(a)}{p(r)} - \sum_{o\in\mathcal{O}_r}\frac{p(o)}{p(r)}\sum_{a\in\mathcal{A}_r} p(a|o)\log p(a|o)\Big] \tag{7.1}$$

Now for each $r \in \mathcal{R}$ we define a distribution p_r over $\mathcal{A}_r$ as follows:

$$p_r(a) = \frac{p(a)}{p(r)}$$

It is easy to verify that this is indeed a probability distribution. We use p_r as the input distribution in channel M_r and since, by construction of M_r, $p_r(o|a) = p(o|a)$ we have

$$p_r(o) = \sum_{a\in\mathcal{A}_r} p_r(a)p_r(a|o) = \sum_{a\in\mathcal{A}_r}\frac{p(a)}{p(r)}p(a|o) = \frac{p(o)}{p(r)}$$

Now equation (7.1) can be written:

$$\begin{aligned}
&I(A;O|R)\\
&= \sum_{r\in\mathcal{R}} p(r)\Big[-\sum_{a\in\mathcal{A}_r} p_r(a)\log p_r(a) + \sum_{o\in\mathcal{O}_r} p_r(o)\sum_{a\in\mathcal{A}_r} p_r(a|o)\log p_r(a|o)\Big]\\
&= \sum_{r\in\mathcal{R}} p(r)\Big[H(A_r) - H(A_r|O_r)\Big]
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{r \in \mathcal{R}} p(r) I(A_r; O_r) \\
&\leq \sum_{r \in \mathcal{R}} p(r) d \\
&= d
\end{aligned}
$$

Where A_r, O_r are the input and output random variables of channel M_r. Finally, since $I(A; O|R) \leq d$ for all input distributions we have $C|R \leq d$. □

7.2 Computing the channel's capacity

For arbitrary channels, there is no analytic formula to compute their capacity. In the general case we can only use numerical algorithms that converge to the capacity, as we discuss in the end of this section. In practice, however, channels have symmetry properties that can be exploited to compute the capacity in an easy way. In this section we define classes of symmetry and discuss how to compute the capacity for each class. Two classic cases are the *symmetric* and *weakly symmetric* channels.

Definition 7.2.1. *A matrix is* symmetric *if all rows are permutations of each other and all columns are also permutations of each other. A matrix is* weakly *symmetric if all rows are permutations of each other and the column sums are equal.*

The following result is from the literature:

Theorem 7.2.2 ([CT91], page 189)**.** *Let $(\mathcal{A}, \mathcal{O}, p_c)$ be a channel. If p_c is weakly symmetric then the channel's capacity is given by a uniform input distribution and is equal to*

$$C = \log |\mathcal{O}| - H(\mathbf{r})$$

where $\mathbf{r}$ is a row of the matrix and $H(\mathbf{r})$ is the entropy of $\mathbf{r}$.

Note that symmetric channels are also weakly symmetric so Theorem 7.2.2 holds for both classes.

In anonymity protocols, users usually execute exactly the same protocol, with the only difference being the names of the agents to whom they communicate. So if a user a_1 produces an observable o_1 with probability p, it is reasonable to assume that a_2 will produce some observable o_2 with the same probability. In other words we expect all rows of the protocol's matrix to be permutations of each other. On the other hand, the columns are not necessarily permutations of each other, as we will see in the example of Section 7.5. The problem is that o_1 and o_2 above need not be necessarily different, that is we can have the same observable produced with equal probability by all users. Clearly, these "constant" columns cannot be the permutation of non-constant ones so the resulting channel matrix will not be symmetric (and not even weakly symmetric).

To cope with this kind of channel we define a more relaxed kind of symmetry called *partial symmetry*. In this class we allow some columns to be constant and we require the sub-matrix, composed only of the non-constant columns, to be symmetric. A weak version of this symmetry can also be defined.

Definition 7.2.3. *A matrix is* partially symmetric *(resp. weakly partially symmetric) if some columns are constant (possibly with different values in each column) and the rest of the matrix is symmetric (resp. weakly symmetric).*

Now we can extend Theorem 7.2.2 to the case of partial symmetry.

Theorem 7.2.4. *Let $(\mathcal{A}, \mathcal{O}, p_c)$ be a channel. If p_c is weakly partially symmetric then the channel's capacity is given by*

$$C = p_s \log \frac{|\mathcal{O}_s|}{p_s} - H(\mathbf{r}_s)$$

where $\mathcal{O}_s$ is the set of symmetric output values, $\mathbf{r}_s$ is the symmetric part of a row of the matrix and p_s is the sum of $\mathbf{r}_s$.

Proof. Let $\mathcal{O}_s$ by the set of symmetric output values (the ones that correspond to the symmetric columns) and $\mathcal{O}_n$ the set of the non-symmetric ones. Also let $\mathbf{r}$ be a row of the matrix and $\mathbf{r}_s$ the symmetric part of $\mathbf{r}$. Since the matrix is partially symmetric all rows are permutations of each other. As a consequence:

$$H(O|A) = -\sum_o p(o) \sum_a p(o|a) \log p(o|a) = H(\mathbf{r})$$

Moreover the columns in $\mathcal{O}_n$ are constant so for all $o \in \mathcal{O}_n$, $p(o)$ is independent of the input distribution: $p(o) = \sum_a p(a)p(o|a) = p(o|a')$ for some fixed a'. We have

$$\begin{aligned}
I(A;O) &= H(O) - H(O|A) \\
&= -\sum_{o\in\mathcal{O}} p(o) \log p(o) - H(\mathbf{r}) \\
&= -\sum_{o\in\mathcal{O}_s} p(o) \log p(o) - \sum_{o\in\mathcal{O}_n} p(o|a') \log p(o|a') - H(\mathbf{r}) \\
&= -\sum_{o\in\mathcal{O}_s} p(o) \log p(o) - H(\mathbf{r}_s) \\
&\leq -\sum_{o\in\mathcal{O}_s} \frac{p_s}{|\mathcal{O}_s|} \log \frac{p_s}{|\mathcal{O}_s|} - H(\mathbf{r}_s) \qquad (7.2) \\
&= p_s \log \frac{|\mathcal{O}_s|}{p_s} - H(\mathbf{r}_s) \qquad (7.3)
\end{aligned}$$

We constructed inequality (7.2) by taking a uniform distribution $p(o) = \frac{p_s}{|\mathcal{O}_s|}$ of symmetric outputs (the non-symmetric outputs have constant probabilities). p_s is the total probability of having an output among those in $\mathcal{O}_s$. Now if we take a uniform input distribution $p(a) = \frac{1}{|\mathcal{A}|}$ then for all $o \in \mathcal{O}_s : p(o) = \sum_a p(a)p(o|a) = \frac{c}{|\mathcal{A}|}$ where c is the sum of the corresponding column which is the same for all symmetric output values. So a uniform input distribution produces a uniform distribution of the symmetric output values, thus the bound (7.3) is achieved and it is the actual capacity of the channel. □

Note that Theorem 7.2.4 is a generalization of Theorem 7.2.2. A (weakly) symmetric channel can be considered as (weakly) partially symmetric with no

constant columns. In this case $O_s = O, \mathbf{r}_s = \mathbf{r}, p_s = 1$ and we retrieve Theorem 7.2.2 from Theorem 7.2.4.

In all cases of symmetry discussed above, computing the capacity is a simple operation involving only one row of the matrix and can be performed in $O(|\mathcal{O}|)$ time.

In the general case of no symmetry we must use a numerical algorithm, like the Arimoto-Blahut algorithm (see for instance [CT91]) which can compute the capacity to any desired accuracy. However the convergence rate is slow (linear) and the coefficient of the convergence speed gets smaller when the number of input values increases.

7.3 Relation with existing anonymity notions

In this section we consider some particular channels, and we illustrate the relation with probabilistic (non information-theoretic) notions of anonymity existing in literature.

7.3.1 Capacity 0: strong anonymity

The case in which the capacity of the anonymity protocol is 0 is by definition obtained when $I(A;O) = 0$ for all possible input distributions of $\mathcal{A}$. From information theory we know that this is the case iff A and O are independent (cfr. [CT91], page 27). Hence we have the following characterization:

Proposition 7.3.1. *Given an anonymity system $(\mathcal{A}, \mathcal{O}, p_c)$, the capacity of the corresponding channel is 0 iff the system satisfies strong anonymity, that is if all the rows of the channel matrix are the same.*

Proof. The channel capacity is zero if and only if A and O are independent, that is $p(a,o) = p(a)p(o) \Leftrightarrow p(a) = p(a|o)$ for all $o \in \mathcal{O}, a \in \mathcal{A}$. The latter condition is known as conditional anonymity (Def. 5.1.2) and by Theorem 5.1.3 it is equivalent to strong anonymity. □

An example of a protocol with capacity 0 is the dining cryptographers in a connected graph under the assumption of fair coins and considering only the case where one of the cryptographers (and never the master) pays.

7.3.2 Conditional capacity 0: strong anonymity "within a group"

In some anonymity protocols, the users are divided in groups and the protocol allows the adversary to figure out to which group the culprit belongs, although it tries to conceal which user in the group is the culprit. This is the case, for example, of the dining cryptographers in a generic (non-connected) graph, where the groups correspond to the connected components of the graph.

Such a situation corresponds to having a partition on $\mathcal{A}$ and $\mathcal{O}$, see Section 7.1.1. The case of conditional capacity 0 is obtained when each M_{r_i} has capacity 0, namely when in each group r_i the rows are identical.

Proposition 7.3.2. *The dining cryptographers in a generic graph has conditional capacity 0, under the assumption that the coins are fair.*

Proof. We consider the model of the protocol described in Section 5.2. Let G be the graph of the protocol, consisting of l connected components $G_1, \ldots, G_l$. The attacker is allowed to know which connected component the user belongs to, so we define the set of revealed values as $\mathcal{R} = \{1, \ldots, l\}$. We first show that R is a deterministic function of both A, O. Since a user can belong to only one connected component we have $p(r|a) = 1$ if $a \in G_r$ and 0 otherwise. Concerning the observables, in the connected component of the payer the sum of all announcements will have odd parity while in all other components the parity will be even. So $p(r|\vec{o}) = 1$ iff $\sum_{a_i \in G_r} o_i = 1$ and $p(r|\vec{o}) = 0$ otherwise.

So from Theorem 7.1.3 the matrix of the channel consists of smaller sub-matrices, one for each connected component. In each component the protocol is strongly anonymous (Theorem 5.2.1) so the corresponding sub-channel has capacity 0. Since all sub-channels have capacity zero from Theorem 7.1.3 we have $C|R = 0$ for the whole channel. □

One of the authors of [SS00], David Sands, has suggested to us that the notion of strong anonymity "within a group" seems related to the notion of equivalence classes in his work. Exploring this connection is left for future work.

7.3.3 Probable innocence: weaker bounds on capacity

Probable innocence is a weak notion of anonymity introduced by Reiter and Rubin [RR98] for Crowds. Probable innocence was verbally defined as "from the attacker's point of view, the sender appears no more likely to be the originator of the message than to not be the originator". As we discussed in Chapter 6, there are three different definitions that try to formally express this notion, two from the literature and one described in Section 6.2. In this section we discuss the relation between these definitions and the channel capacity.

Definition of Reiter and Rubin

In [RR98] Reiter and Rubin gave a verbal definition of probable innocence and then formalized it and proved it for the Crowds protocol. Their formalization considers the probability that the originator forwards a message directly to a corrupted member (the attacker) and requires this probability to be at most one half. As explained in Section 6.1.1, this definition could be expressed in the framework of Chapter 4 as follows: an anonymity system $(\mathcal{A}, \mathcal{O}, p_c)$ satisfies RR-probable innocence if

$$p_c(o|a) \leq \frac{1}{2} \quad \forall o \in \mathcal{O}, \forall a \in \mathcal{A}$$

In Section 6.1.1 it is argued that this definition makes sense for Crowds due to certain properties that Crowds satisfies, however it is not suitable for arbitrary protocols.

We now show that RR-probable innocence imposes no bound on the capacity of the corresponding channel. Consider, for example, the protocol shown in figure 7.3. The protocols satisfies RR-probable innocence since all values of the matrix are less than or equal to one half. However the channel capacity is (the matrix is symmetric) $C = \log |\mathcal{O}| - H(\mathbf{r}) = \log(2n) - \log 2 = \log n$ which

	o_1	o_2	o_3	o_4	$\cdots$	o_{2n-1}	o_{2n}
a_1	1/2	1/2	0	0	...	0	0
a_1	0	0	1/2	1/2	...	0	0
$\vdots$	$\vdots$				$\ddots$		$\vdots$
a_n	0	0	0	0	...	1/2	1/2

Figure 7.3: A maximum-capacity channel which satisfies RR-probable innocence

is the maximum possible capacity, equal to the entropy of A. Indeed, users can be perfectly identified by the output since each observable is produced by exactly one user.

Note, however, that in Crowds a bound on the capacity can be obtained due to the special symmetries that it satisfies which make RR-probable innocence equivalent to the new definition of probable innocence.

Definition of Halpern and O'Neill

In [HO05] Halpern and O'Neill give a definition of probable innocence that focuses on the attacker's confidence that a particular anonymous event happened, after performing an observation. It requires that the probability of an anonymous event should be at most one half, under any observation. According to the Definition 6.1.2, an anonymity instance satisfies HO-probable innocence if

$$p(a|o) \leq \frac{1}{2} \ \forall o \in \mathcal{O}, \forall a \in \mathcal{A}$$

This definition looks like the one of Reiter and Rubin but its meaning is very different. It does not limit the probability of observing o. Instead, it limits the probability of an anonymous event a given the observation of o.

As discussed in Section 6.1.2, the problem with this definition is that it depends on the probabilities of the anonymous events which are not part of the protocol. As a consequence, HO-probable innocence cannot hold for all input distributions. If we consider a distribution where $p(a)$ is very close to 1, then $p(a|o)$ cannot possibly be less than 1/2. So we cannot speak about the bound that HO-probable innocence imposes to the capacity, since to compute the capacity we quantify over all possible input distributions and HO-probable innocence cannot hold for all of them. However, if we limit ourselves to the input distributions where HO-probable innocence actually holds, then we can prove the following proposition.

Proposition 7.3.3. *Let $(\mathcal{A}, \mathcal{O}, p_c)$ be an anonymity system and $p_{\mathcal{A}}$ a fixed distribution over $\mathcal{A}$. If the channel is symmetric and satisfies HO-probable innocence for this input distribution then $I(A;O) \leq H(A) - 1$.*

Proof. If X is a random variable and f a function on $\mathcal{X}$, we will denote by $Ef(X)$ the expected value of $f(X)$. Note that $H(X) = -E \log p(X)$ and $H(X|Y) = -E \log p(X|Y)$.

We have

$$I(A;O) = H(A) - H(A|O) = H(A) + E\log p(A|O)$$

And since $p(A|O) \leq 1/2$ and both log and E are monotonic

$$I(A;O) \leq H(A) + E\log\frac{1}{2} = H(A) - 1$$

□

Note that we consider the mutual information for a specific input distribution, not the capacity, for the reasons explained above.

New definition of probable innocence

The new definition of probable innocence presented in the previous chapter (Def. 6.2.2) tries to combine the other two by considering both the probability of producing some observable and the attacker's confidence after the observation. This definition considers the probability of two anonymous events a, a' producing the same observable o and does not allow $p_c(o|a)$ to be too high or too low compared to $p_c(o|a')$. A protocol satisfies probable innocence if

$$(n-1)p_c(o|a') \geq p_c(o|a) \quad \forall o \in \mathcal{O}, \forall a, a' \in \mathcal{A}$$

where $n = |\mathcal{A}|$. In Section 6.2 it is shown that this definition overcomes some drawbacks of the other two definitions of probable innocence and it is argued that it is more suitable for general protocols. In this section we show that the new definition imposes a bound on the capacity of the corresponding channel, which strengthens our belief that it is a good definition of anonymity.

Since the purpose of this definition is to limit the fraction $\frac{p(o|a)}{p(o|a')}$ we could generalize it by requiring this fraction to be less than or equal to a constant γ.

Definition 7.3.4. *An anonymity protocol* $(\mathcal{A}, \mathcal{O}, p_c)$ *satisfies* partial anonymity *if there is a constant* γ *such that*

$$\gamma\, p_c(o|a') \geq p_c(o|a) \quad \forall o \in \mathcal{O}, \forall a, a' \in \mathcal{A}$$

A similar notion is called *weak probabilistic anonymity* in [DPP06].

Note that partial anonymity generalizes both probable innocence ($\gamma = n-1$) and strong anonymity ($\gamma = 1$). The following theorem shows that partial anonymity imposes a bound to the channel capacity:

Theorem 7.3.5. *Let* $S = (\mathcal{A}, \mathcal{O}, p_c)$ *be an anonymity system. If* S *satisfies partial anonymity with* $\gamma > 1$ *and the matrix* p_c *is symmetric then*

$$C(S) \leq \frac{\log\gamma}{\gamma - 1} - \log\frac{\log\gamma}{\gamma - 1} - \log\ln 2 - \frac{1}{\ln 2}$$

Proof. Since the channel is symmetric, by Theorem 7.2.2 its capacity is given by $\log|\mathcal{O}| - H(\mathbf{r})$ where $\mathbf{r}$ is a row of the matrix. We consider the first row which contains values of the form $p_c(o|a_1), o \in \mathcal{O}$. Since the columns are permutations of each other, we have $\forall o \exists a : p_c(o|a_1) = p_c(o_1|a)$. And since the

protocol satisfies partial anonymity we have $\forall a, a' \in \mathcal{A} : \gamma\, p_c(o_1|a') \geq p_c(o_1|a)$, thus

$$\gamma\, p_c(o'|a_1) \geq p_c(o|a_1) \qquad \forall o, o' \in \mathcal{O} \tag{7.4}$$

First we show that when we decrease the distance between the probabilities in a distribution then the entropy increases (this is a standard result from information theory). Let $\vec{x} = (x_1, x_2, \ldots, x_n)$ such that $x_1 < x_2$ and let $\vec{x_o} = (x_1 + d, x_2 - d, \ldots, x_n)$ with $d \leq x_2 - x_1$. We can write $\vec{x_o}$ as a convex combination $t\vec{x} + (1-t)\vec{x_p}$ where $t = 1 - \frac{d}{x_2 - x_1}$ and $\vec{x_p} = (x_2, x_1, \ldots, x_n)$. Since $H(\vec{x}) = H(\vec{x_p})$ and $H(\vec{x})$ is a concave function of $\vec{x}$ we have

$$H(\vec{x_o}) = H(t\vec{x} + (1-t)\vec{x_p}) \geq tH(\vec{x}) + (1-t)H(\vec{x_p}) = H(\vec{x})$$

Let p be the minimum value of the row $\mathbf{r}$. By (7.4) the maximum value of $\mathbf{r}$ will be at most γp. To maximize the capacity we want to minimize $H(\mathbf{r})$ so we will construct the row which gives the minimum possible entropy without violating (7.4). If there are any values of the row between p and γp we could subtract some probability from one and add it to another value. Since this operation increases the distance between the values, it decreases the entropy of the row as we showed before (in the inverse direction). So for a fixed p the lowest entropy is given by the row whose values are either p or γp. After that we can no longer separate the values without violating (7.4). However, this is a local optimum. If we take a new p' and construct a new row with values p' and $\gamma p'$ then we might find an even lower entropy.

Let x be the number of elements with value γp. Also let $m = |\mathcal{O}|$. We have

$$(m - x)p + x\gamma p = 1 \Rightarrow p = \frac{1}{A} \quad \text{with} \quad A = x(\gamma - 1) + m$$

And the entropy of $\mathbf{r}$ will be

$$\begin{aligned} H(\mathbf{r}) &= -(m-x)\frac{1}{A}\log\frac{1}{A} - x\frac{\gamma}{A}\log\frac{\gamma}{A} \\ &= (-x(\gamma-1) - m)\frac{1}{A}\log\frac{1}{A} - x\frac{\gamma}{A}\log\gamma \\ &= \log A - x\frac{\gamma}{A}\log\gamma \end{aligned}$$

So $H(\mathbf{r})$ is a function $h(x)$ of only one variable x. We want to find the value x_0 which minimizes $h(x)$. First we differentiate $h(x)$

$$h'(x) = \frac{1}{\ln 2}\frac{\gamma - 1}{A} - \gamma\log\gamma\frac{m}{A^2}$$

And x_0 will be the value for which

$$\begin{aligned} h(x_0) = 0 \Rightarrow \\ \frac{1}{\ln 2}\frac{\gamma-1}{x_0(\gamma-1)+m} = \frac{m\gamma\log\gamma}{(x_0(\gamma-1)+m)^2} \Rightarrow \\ x_0 = \frac{A_0 - m}{\gamma - 1} \quad \text{with} \\ A_0 = \frac{m\gamma\log\gamma\ln 2}{\gamma - 1} \end{aligned}$$

Finally the minimum entropy of $\mathbf{r}$ will be equal to

$$\begin{aligned} h(x_0) &= \log \frac{m\gamma \log \gamma \ln 2}{\gamma - 1} - \frac{\gamma \log \gamma}{\gamma - 1} + \frac{1}{\ln 2} \\ &= \log m - \frac{\log \gamma}{\gamma - 1} + \log \log \gamma - \log(\gamma - 1) + \log \ln 2 + \frac{1}{\ln 2} \end{aligned}$$

And the maximum capacity will be

$$\begin{aligned} C_{\max} &= \log m - h(x_0) \\ &= \frac{\log \gamma}{\gamma - 1} - \log \frac{\log \gamma}{\gamma - 1} - \log \ln 2 - \frac{1}{\ln 2} \end{aligned}$$

□

This bound has two interesting properties. First, it depends only on γ and not on the number of input or output values or on other properties of the channel matrix. Second, the bound converges to 0 as $\gamma \to 1$. As a consequence, due to the continuity of the capacity as a function of the channel matrix, we can retrieve Proposition 7.3.1 about strong anonymity ($\gamma = 1$) from Theorem 7.3.5. A bound for probable innocence can be obtained by taking $\gamma = n - 1$, so Theorem 7.3.5 treats strong anonymity and probable innocence in a uniform way. Note that this bound is proved for the special case of symmetric channels, we plan to examine the general case in the future.

7.4 Adding edges to a dining cryptographers network

We turn our attention again to the dining cryptographers protocol where we use the new definition of anonymity to compare different cryptographer networks. Consider a dining cryptographers instance with an arbitrary network graph G_c and possibly biased coins. The anonymity guarantees of the protocol come from the fact that the unknown values of the coins add noise to the output of the cryptographers. Now imagine that we add a new edge to the graph, that is a new coin shared between two cryptographers, obtaining a new graph G'_c. If the new coin is fair then intuitively we would expect the new graph to have at least the same anonymity guarantees as the old one, if not better. If the new coin is biased the intuition is not so clear, but it is still true that we add more noise to the system so we could expect the same behavior. In this section we explore this idea and prove various results about the anonymity of the resulting system.

This section is somewhat transversal in the sense that some of its results are about topics which belong to the scope of other chapters, but we decided to keep them together because they are strictly interconnected. Let us explain how they are articulated. The main result of this section is Theorem 7.4.3, which states that the capacity of the system decreases monotonically with the insertion of a new edge. In order to prove the main result, we start by showing that the conditional probabilities of the new instance are convex combinations of conditional probabilities of the old one, where the coefficients are the probabilities of the added coin (Proposition 7.4.1). As a side result of this proposition, we prove that if the old system satisfies probable innocence, then also the new system does (Corollary 7.4.2). As a consequence of the main

theorem, we are able to strengthen Chaum's result, namely we prove that in order for a component to be strongly anonymous it is sufficient to have a spanning tree consisting of fair coins (Corollary 7.4.4). Finally, we prove that this condition is also necessary (Theorem 7.4.5).

It is important to note that when we add an edge to the graph, the number of observables remains the same, but the conditional probabilities $p_c(\vec{o}|a)$ change. The following proposition states how the new probabilities can be expressed in terms of the old ones.

Proposition 7.4.1. *Let G_c be a connected component of a dining cryptographers graph and $S(G_c) = (\mathcal{A}, \mathcal{O}, p_c)$ the corresponding anonymity system. Let G'_c be the graph produced by adding an edge (coin) to G_c and let h, t be the probability of heads/tails of the added coin. Then $S(G'_c) = (\mathcal{A}, \mathcal{O}, p'_c)$ where*

$$p'_c(\vec{o}|a) = h\, p_c(\vec{o}|a) + t\, p_c(\vec{o} \oplus \vec{w}|a) \quad \forall \vec{o} \in \mathcal{O}, a \in \mathcal{A}$$

where $\vec{w}$ is a fixed vector of even parity (depending only on G'_c).

Proof. Let n, m be the number of vertices and edges of G_c. Also let B, B' be the incidence matrices of G_c, G'_c respectively. B' is be the same as B with an extra column corresponding to the added edge. We fix an $\vec{o} \in \mathcal{O}$ and $a \in \mathcal{A}$. The coin configurations that produce $\vec{o}$ in the output of G'_c will be the solutions of the system of equations $B'\vec{x} = \vec{o} \oplus \vec{r}$ where $\vec{r}$ is the inversion vector corresponding to the cryptographer a. As already discussed in the proof of Theorem 5.2.1, B' has rank $n-1$ and the system is solvable with 2^{n-m} solutions.

Let $\mathcal{C} \subseteq \mathrm{GF}(2)^{m+1}$ be the set of its solutions. We split $\mathcal{C}$ in two subsets $\mathcal{C}_0, \mathcal{C}_1$ based on the $(m+1)$-th coin (the added one) where its value in all elements of $\mathcal{C}_0, \mathcal{C}_1$ is $0, 1$ respectively. Let $p_i(0), p_i(1)$ be the probabilities of the i-th coin giving heads, tails respectively, thus $h = p_{m+1}(0), t = p_{m+1}(1)$. The probability $p'(\vec{o}|a)$ is

$$\begin{aligned} p'_c(\vec{o}|a) &= \sum_{\vec{c} \in \mathcal{C}} \prod_{i=1}^{m+1} p_i(c_i) \\ &= \sum_{\vec{c} \in \mathcal{C}_0} \prod_{i=1}^{m+1} p_i(c_i) + \sum_{\vec{c} \in \mathcal{C}_1} \prod_{i=1}^{m+1} p_i(c_i) \\ &= h \left(\sum_{\vec{c} \in \mathcal{C}_0} \prod_{i=1}^{m} p_i(c_i) \right) + t \left(\sum_{\vec{c} \in \mathcal{C}_1} \prod_{i=1}^{m} p_i(c_i) \right) \end{aligned} \tag{7.5}$$

If $\vec{x}$ is a vector in a n-dimensional space we will denote by $\vec{y} = (\vec{x}, v)$ the vector in a $(n+1)$-dimensional space such that $y_i = x_i, 1 \leq i \leq n$ and $y_{n+1} = v$. Consider a vector $(\vec{c}, 0) \in \mathcal{C}_0$. Since its last element is 0 then $B\vec{c} = B'(\vec{c}, 0)$. So $\vec{c}$ is a solution to the system $B\vec{x} = \vec{o} \oplus \vec{r}$, that is $\vec{c}$ is a coin configuration that produces $\vec{o}$ in the output of G_c (intuitively, this means that adding a zero coin to a configuration does not change the output). So

$$p_c(\vec{o}|a) = \sum_{\vec{c} \in \mathcal{C}_0} \prod_{i=1}^{m} p_i(c_i) \tag{7.6}$$

The most interesting case however are the vectors $(\vec{c}, 1) \in \mathcal{C}_1$ since now $\vec{c}$ is not a solution to $B\vec{x} = \vec{o} \oplus \vec{r}$. We write $(\vec{c}, 1)$ as $(\vec{c}, 0) \oplus \vec{I}$ where $\vec{I}$ is a vector having 1 as its $(m+1)$-th element and 0 everywhere else. Then we have

$$\begin{aligned} B'(\vec{c}, 1) &= \vec{o} \oplus \vec{r} \Leftrightarrow \\ B'((\vec{c}, 0) \oplus \vec{I}) &= \vec{o} \oplus \vec{r} \Leftrightarrow \\ B'(\vec{c}, 0) &= \vec{o} \oplus B'\vec{I} \oplus r \end{aligned}$$

so $(\vec{c}, 0)$ is a solution to $B'\vec{x} = \vec{o} \oplus \vec{w} \oplus r$, where $\vec{w} = B'\vec{I}$, and as discussed above, $\vec{c}$ is a solution to $B\vec{x} = \vec{o} \oplus \vec{w} \oplus r$. Note that $\vec{w}$ has even parity, so $\vec{o} \oplus \vec{w}$ has even parity so it is itself an observable. Thus

$$p_c(\vec{o} \oplus \vec{w}|a) = \sum_{\vec{c} \in \mathcal{C}_1} \prod_{i=1}^{m} p_i(c_i) \tag{7.7}$$

Also note that $\vec{w}$ is a fixed vector, it does not depend either on $\vec{o}$ or on a. In fact, $\vec{w}$ is a vector containing 1 in the positions of the cryptographers joined by the added edge and 0 everywhere else. Finally, (7.5) using (7.6),(7.7) becomes

$$p'_c(\vec{o}|a) = h\, p_c(\vec{o}|a) + t\, p_c(\vec{o} \oplus \vec{w}|a)$$

□

Previous proposition allows us to show, as a side results, that adding an edge to a dining cryptographers graph preserves probable innocence.

Corollary 7.4.2. *Let G_c be a connected component of a dining cryptographers graph and G'_c the graph produced by adding an edge. Also let $S(G_c) = (\mathcal{A}, \mathcal{O}, p_c)$ and $S(G'_c) = (\mathcal{A}, \mathcal{O}, p'_c)$ be the corresponding anonymity systems. If $S(G_c)$ satisfies probable innocence then $S(G'_c)$ also satisfies it.*

Proof. Since $S(G_c)$ satisfies probable innocence then $(n-1)p(\vec{o}|a) \geq p(\vec{o}|a')$ for all $\vec{o} \in \mathcal{O}, a, a' \in \mathcal{A}$. For $S(G'_c)$ we have

$$\begin{aligned} (n-1)p'_c(\vec{o}|a) &= (n-1)h\, p_c(\vec{o}|a) + (n-1)t\, p_c(\vec{o} \oplus \vec{w}|a) && \text{Prop. 7.4.1} \\ &\geq h\, p_c(\vec{o}|a') + t\, p_c(\vec{o} \oplus \vec{w}|a') && \text{Probable Inn.} \\ &= p'_c(\vec{o}|a') && \text{Prop. 7.4.1} \end{aligned}$$

□

The above result conforms to our intuition that we cannot make the protocol less anonymous by adding new coins. However, by saying that both systems satisfy probable innocence we don't actually compare them. Either of the two could be "worse" that the other, while still satisfying the property. The inability to compare protocols of the same family was one of the reasons that led us to a quantitative definition of anonymity. Using the new definition we can show that the degree of anonymity of the protocol after the addition of the edge is at least as good and that of the original protocol.

To show this result we use the fact that capacity is a convex function of the channel's matrix, that is $C(t_1\, M_1 + t_2\, M_2) \leq t_1\, C(M_1) + t_2\, C(M_2)$ where t_1, t_2 are positive coefficients such that $t_1 + t_2 = 1$ and M_1, M_2 are matrices of the

same size. This is an important property of capacity that leads to many useful results. We will give a proof of it in the next chapter (Theorem 8.1.3) since it fits better there, for the moment we take it for granted.

Theorem 7.4.3. *Let G_c be a connected component of a dining cryptographers graph and G'_c the graph produced by adding an edge. Also let $S(G_c) = (\mathcal{A}, \mathcal{O}, p_c)$ and $S(G'_c) = (\mathcal{A}, \mathcal{O}, p'_c)$ be the corresponding anonymity systems. Then*

$$C(G'_c) \leq C(G_c)$$

Proof. From Proposition 7.4.1 we have $p'_c(\vec{o}|a) = h\, p_c(\vec{o}|a) + t\, p_c(\vec{o} \oplus \vec{w}|a)$ for a fixed vector $\vec{w}$. Let M be the channel matrix of $S(G_c)$[3], we create a matrix M_p by permuting the columns of M so that the column $\vec{o} \oplus \vec{w}$ is placed at the position of $\vec{o}$. Since we only permuted the columns, $C(M) = C(M_p)$. Now we can write the matrix M' of $S(G'_c)$ as a convex combination of M and M_p

$$M' = h\, M + t\, M_p$$

and finally, because of the convexity of capacity as a function of the matrix, we get

$$\begin{aligned} C(M') &= C(h\, M + t\, M_p) & \\ &\geq h\, C(M) + t\, C(M_p) & \text{by convexity} \\ &= h\, C(M) + t\, C(M) & C(M) = C(M_p) \\ &= C(M) & \end{aligned}$$

□

As a consequence of the above theorem we are able to prove an interesting and somehow counter-intuitive result about strong anonymity. Chaum's proof (Theorem 5.2.1) says that dining cryptographers on an arbitrary connected graph G_c is strongly anonymous if all the coins are fair. However, it states this condition as sufficient, not necessary for strong anonymity. Indeed, not all coins need to be fair. We show that having a spanning tree of fair coins is enough, even if the rest of the coins are biased.

Corollary 7.4.4. *A dining cryptographers instance is strongly anonymous with respect to a connected component G_c if G_c has a spanning tree consisting only of fair coins.*

Proof. Let G_t be the spanning tree of C_c. Since G_t is connected and all coins are fair then $S(G_t)$ is strongly anonymous so $C(S(G_t)) = 0$. We can reconstruct G_c from G_t by adding the remaining edges, so by Theorem 7.4.3 $C(S(G_c)) \leq C(S(G_t)) = 0$. Hence G_c is strongly anonymous. □

Finally we show that the above is also a necessary condition, namely a dining cryptographers instance is strongly anonymous with respect to a connected component if and only if the component has a spanning tree consisting of only fair coins.

In order to understand this result, let us remind the reader that we assume that the matrix of the protocol is known to the adversary. This implies that (in general) the adversary knows whether a coin is biased, and how it is biased.

[3]Note that M is not the incidence matrix of G_c but the matrix of conditional probabilities of the channel.

Theorem 7.4.5. *A dining cryptographers instance is strongly anonymous with respect to a connected component G_c only if G_c has a spanning tree consisting only of fair coins.*

Proof. By contradiction. Let n be the number of vertices in G_c. Assume that G_c is strongly anonymous without having a spanning tree consisting only of fair coins. Then it is possible to split G_c in two non-empty subgraphs, G_1 and G_2, such that all the edges between G_1 and G_2 are unfair. Let $\vec{c} = (c_1, c_2, \ldots, c_m)$ be the vector of coins corresponding to these edges. Since G_c is connected, we have that $m \geq 1$.

Let a_1 be a vertex in G_1 and a_2 be a vertex in G_2. By strong anonymity, for every observable $\vec{o}$ we have

$$p(\vec{o} \mid a_1) = p(\vec{o} \mid a_2) \tag{7.8}$$

Observe now that $p(\vec{o} \mid a_1) = p(\vec{o} \oplus \vec{w} \mid a_2)$ where $\vec{w}$ is a vector in $\mathrm{GF}(2)^n$ containing 1 exactly twice, in correspondence of a_1 and a_2. Hence (7.8) becomes

$$p(\vec{o} \oplus \vec{w} \mid a_2) = p(\vec{o} \mid a_2) \tag{7.9}$$

Let d be the binary sum of all the elements of $\vec{o}$ in G_1, and d' be the binary sum of all the elements of $\vec{o} \oplus \vec{w}$ in G_1. Since in G_1 $\vec{w}$ contains 1 exactly once, we have $d' = d \oplus 1$. Hence (7.9), being valid for all $\vec{o}$'s, implies

$$p(d \oplus 1 \mid a_2) = p(d \mid a_2) \tag{7.10}$$

Because of the way o, and hence d, are calculated, and since the contribution of the edges internal to G_1 is 0, and a_2 (the payer) is not in G_1, we have that

$$d = \sum_{i=1}^{m} c_i$$

from which, together with (7.10), and the fact that the coins are independent from the choice of the payer, we derive

$$p(\sum_{i=1}^{m} c_i = 0) = p(\sum_{i=1}^{m} c_i = 1) = 1/2 \tag{7.11}$$

The last step is to prove that $p(\sum_{i=1}^{m} c_i = 0) = 1/2$ implies that one of the c_i's is fair, which will give us a contradiction. We prove this by induction on m. The property obviously holds for $m = 1$. Let us now assume that we have proved it for the vector $(c_1, c_2, \ldots, c_{m-1})$. Observe that $p(\sum_{i=1}^{m} c_i = 0) = p(\sum_{i=1}^{m-1} c_i = 0)p(c_m = 0) + p(\sum_{i=1}^{m-1} c_i = 1)p(c_m = 1)$. From (7.11) we derive

$$p(\sum_{i=1}^{m-1} c_i = 0)p(c_m = 0) + p(\sum_{i=1}^{m-1} c_i = 1)p(c_m = 1) = 1/2 \tag{7.12}$$

Now, it is easy to see that (7.12) has only two solutions: one in which $p(c_m = 0) = 1/2$, and one in which $p(\sum_{i=1}^{m-1} c_i = 1) = 1/2$. In the first case we are done, in the second case we apply the induction hypothesis. □

7.5 Computing the degree of anonymity of a protocol

In this section we discuss how to compute the channel matrix and the degree of anonymity for a given protocol, possibly using automated tools. We illustrate our ideas on the dining cryptographers protocol, where we measure the degree of anonymity when modifying the probability of the coins, and on crowds where we measure the degree of anonymity as a function of the probability of forwarding a message.

7.5.1 Dining cryptographers

To measure the degree of anonymity of a system, we start by identifying the set of anonymous events, which depend on what the system is trying to hide. In the dining cryptographers, we take $\mathcal{A} = \{c_1, c_2, c_3, m\}$ where c_i means that cryptographer i is paying and m that the master is paying. Then the set of observable events should also be defined, based on the visible actions of the protocol and on the various assumptions made about the attacker. In the dining cryptographers, we consider for simplicity the case where all the cryptographers are honest and the attacker is an external observer (the case of corrupted cryptographers can be treated similarly). Since the coins are only visible to the cryptographers, the only observables of the protocol are the announcements of *agree*/*disagree*. So the set of observable events will contain all possible combinations of announcements, that is $\mathcal{O} = \{aaa, aad, \ldots, ddd\}$ where a means *agree* and d means *disagree*.

If some information about the anonymous events is revealed intentionally then we should consider using relative anonymity (see Section 7.1.1). In the dining cryptographers, the information about whether the payer is a cryptographer or not is revealed by design (this is the purpose of the protocol). If, for example, the attacker observes aaa then he concludes that the anonymous event that happened is m since the number of *disagree* is even. To model this fact we use the conditional capacity and we take $\mathcal{R} = \{m, c\}$ where m means that the master is paying and c that one of the cryptographers is paying.

After defining $\mathcal{A}, \mathcal{O}, \mathcal{R}$ we should model the protocol in some formal probabilistic language. In our example, we modeled the dining cryptographers in the language of the PRISM model-checker, which is essentially a formalism to describe Markov Decision Processes. Then the channel matrix of conditional probabilities $p_c(o|a)$ must be computed, either by hand or using an automated tool like PRISM. In the case of relative anonymity, the probabilities $p_c(o|a)$ and $p_{\mathcal{R}}(r|a, o)$ are needed for all a, o, r. However, in our example, R is a deterministic function of both A and O, so by Theorem 7.1.3 we can compute the conditional capacity as the maximum capacity of the sub-channels for each value of R individually. For $R = m$ the sub-channel has only one input value, hence its capacity is 0. Therefore the only interesting case is when $R = c$. In our experiments, we use PRISM to compute the channel matrix, while varying the probability p of each coin yielding heads. PRISM can compute the probability of reaching a specific state starting from a given one. Thus, each conditional probability $p_c(o|a)$ is computed as the probability of reaching a state where the cryptographers have announced o, starting from the state where a is chosen. In Fig. 7.4 the channel matrix is displayed for $p = 0.5$ and $p = 0.7$.

Finally, from the matrix, the capacity can be computed in two different

	daa	*ada*	*aad*	*ddd*	*aaa*	*dda*	*dad*	*add*
c_1	0.25	0.25	0.25	0.25	0	0	0	0
c_2	0.25	0.25	0.25	0.25	0	0	0	0
c_3	0.25	0.25	0.25	0.25	0	0	0	0
m	0	0	0	0	0.25	0.25	0.25	0.25

	daa	*ada*	*aad*	*ddd*	*aaa*	*dda*	*dad*	*add*
c_1	0.37	0.21	0.21	0.21	0	0	0	0
c_2	0.21	0.37	0.21	0.21	0	0	0	0
c_3	0.21	0.21	0.37	0.21	0	0	0	0
m	0	0	0	0	0.37	0.21	0.21	0.21

Figure 7.4: The channel matrices for probability of heads $p = 0.5$ (left) and $p = 0.7$ (right)

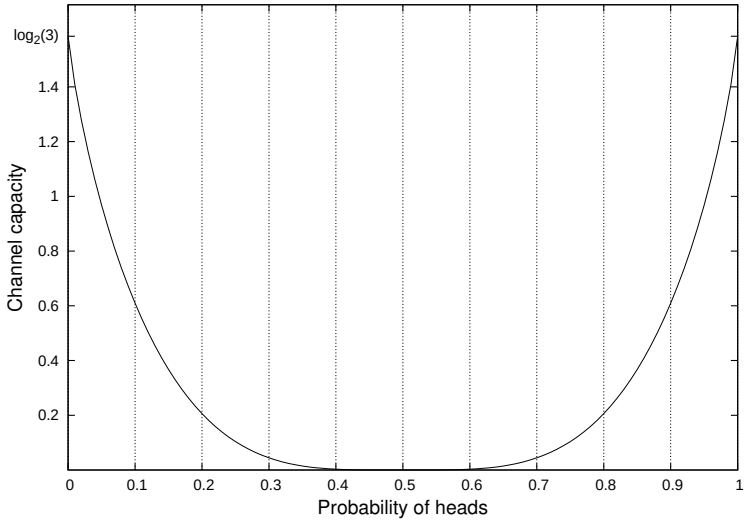


Figure 7.5: The degree of anonymity in the Dining Cryptographers as a function of the coins' probability to yield heads.

ways. Either by using the general Arimoto-Blahut algorithm, or by using Theorem 7.2.4 which can be applied because the matrix is partially symmetric. The resulting graph is displayed in Fig. 7.5. As expected, when $p = 0.5$ the protocol is strongly anonymous and the relative loss of anonymity is 0. When p approaches 0 or 1, the attacker can deduce the identity of the payer with increasingly high probability, so the capacity increases. In the extreme case where the coins are totally biased the attacker can be sure about the payer, and the capacity takes its maximum value of $\log 3$.

In this example, we see how the various results of this chapter fit together when we analyze an anonymity protocol. We model the protocol by considering

the anonymous events $\mathcal{A}$, the observable events $\mathcal{O}$, the revealed information $\mathcal{R}$ and the matrices $p_c(o|a), p_{\mathcal{R}}(r|a,o)$. In this framework, the relative loss of anonymity (Definition 7.1.2) gives an intuitive measure of the anonymity degree of the protocol. Theorem 7.1.3 greatly reduces the size of the problem since we need to consider only the submatrices of $p_c(o|a)$. Partial symmetry simplifies our work even more, we only need to compute one row for each sub-matrix and the computation of the capacity is a very simple operation on this row. Finally, the actual computation of the conditional probabilities that we need can be fully automated using a model-checking tool like PRISM.

7.5.2 Crowds

In this section we do a similar analysis of Crowds and show how to compute its degree of anonymity. Consider a Crowds instance of m users of which n are honest and $c = m - n$ are corrupted. Since anonymity makes sense only for honest users we define $\mathcal{A} = \{a_1, \ldots, a_n\}$ where a_i means that user i is the initiator of the message. The set of observables $\mathcal{O}$ depends on the attacker model, we could measure sender anonymity wrt the end server or wrt the corrupted users of the protocol, here we only consider the latter which is more interesting. The only thing that a corrupted user can observe is a request to forward a message, coming from another user of the protocol. Moreover, as is usually the case in the analysis of Crowds ([Shm02, WALS02]), we assume that a corrupted user will never forward a message sent to him since by doing so he cannot learn more information about the actual initiator. Thus, there is at most one observed user (the one who sent the message to the corrupted user) and it is always an honest one. So we define $\mathcal{O} = \{o_1, \ldots, o_n\}$ where o_i means that the user i forwarded a message to a corrupted user.

The channel matrix $p_c(o|a)$ can be computed either analytically or by means of a model-checking tool like PRISM. The advantage of the second approach is that with minimal changes we could compute the matrix for any network topology, not only for the usual clique network, which is much more difficult to do analytically. In fact, in Chapter 9 we use PRISM to compute the matrix of Crowds in a grid network. Since PRISM can only check finite-state models, we need to model Crowds as a finite-state system, even though its executions are infinite. We use a model similar to the one in [Shm02] where a state is defined by the user who currently possesses the message, independently form the path that the message followed to arrive there, so the number of states is finite. In order for $p_c(\cdot|a)$ to be a distribution over $\mathcal{A}$, we normalize all elements by dividing with the total probability of observing any user. This corresponds to computing all probabilities conditioned on the event that some user has been observed, which is reasonable since if no user is observed at all then anonymity is not an issue.

From the matrix we can compute the capacity, for the case of a clique network, using Theorem 7.2.2 since the matrix is symmetric. As a consequence we only need one row of the matrix, so we can only compute a single one to speed up model-checking. For non-clique networks we can still compute the capacity using the Arimoto-Blahut algorithm.

The resulting graph is displayed in Fig. 7.5.2. We have plotted the capacity of three Crowds instances while varying the probability p_f of forwarding a message in the protocol. All instances have 50 honest users while the num-

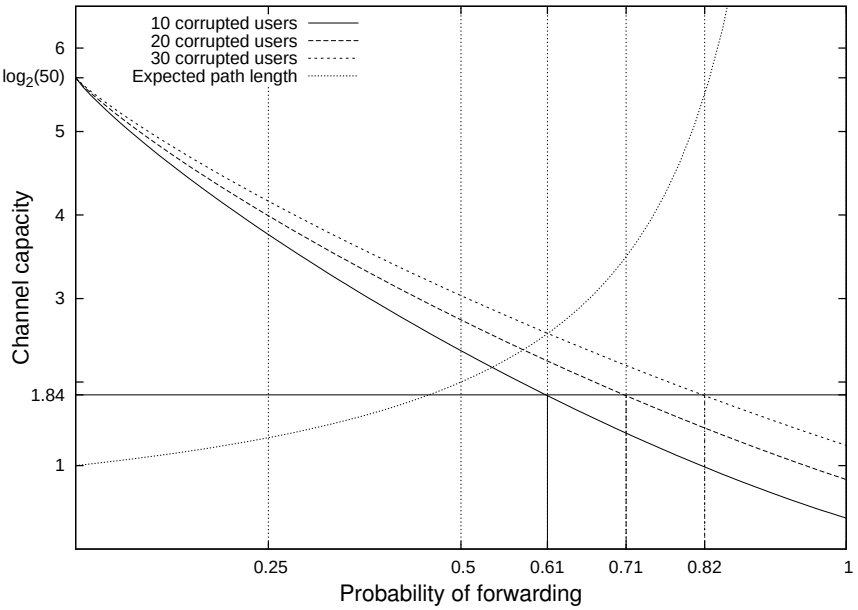


Figure 7.6: The degree of anonymity for Crowds as a function of the probability p_f of forwarding a message. Three instances are displayed, with 50 honest users and 10, 20 and 30 corrupted ones. The expected path length is also displayed as a function of p_f.

ber of corrupted ones is 10, 20 and 30 respectively. Firstly, we see that the whole graph of the capacity is smaller when the number of corrupted users is smaller, which is expected since more corrupted users means higher probability of getting detected in the first round. When $p_f = 0$ then all instances have maximum capacity $\log_2 50$, meaning no anonymity at all, since, if forwarding never happens then the detected user is always the initiator.

For each instance we also indicate the minimum value of p_f required to satisfy probable innocence, given by the equation $m = \frac{p_f}{p_f - \frac{1}{2}}(c+1)$. This value is different for each instance (since m, c are different) however at this value all instances have the same capacity $C = H(p_u) - H(p_{1/2}) \approx 1.8365$ where p_u is a uniform distribution over $\mathcal{A}$ and $p_{1/2}$ is a distribution that assigns probability $1/2$ to one user, and uniform to all the others.

Finally, the expected length of the path to the server, equal to $\frac{1}{1-p_f}$ (as shown in [RR98]) is displayed. As we can see from the graph there is a trade-off between performance (expected path length) and anonymity (capacity) when selecting a value for p_f. Given the maximum number of corrupted users that we want to consider, we can use the graph to find a value for p_f that offers acceptable capacity with a reasonable expected path length. The quantitative aspect of the capacity is important in this case, since it provides more detail about the connection between the degree of anonymity and p_f, even in areas where probable innocence is always satisfied or violated.

7.6 Related work

A recent line of work has been dedicated to exploring the notion of anonymity from an information-theoretic point of view [SD02, DSCP02]. The main difference with our approach is that in those works the anonymity degree is expressed in terms of entropy, rather than mutual information. More precisely, the emphasis is on the lack of information that an attacker has about the distribution of the users, rather than on the capability of the protocol to conceal this information despite the observables that are made available to the attacker. Moreover, a uniform user distribution is assumed, while in our definition we try to abstract from the user distribution and make no assumptions about it.

Channel capacity has been already used in an anonymity context in [MNCM03, MNS03], where the ability to have covert communication as a result of non-perfect anonymity is examined. The difference with our approach is that in those works the channels are constructed by the users of the protocol using the protocol mechanisms, to transfer information, and capacity is used to measure the amount of information that can be transferred through these channels. In our work, we consider the channel to be an abstraction of the protocol itself, and we use the capacity to measure the anonymity degree of the protocol. However in [MNS03] the authors also suggest that the channel's capacity can be used as an asymptotic measure of the worst-case loss of anonymity, which is the idea that we explore in this chapter. Note that in [MNS03] the authors warn that in certain cases the notion of capacity might be too strong a measure to compare systems with, because the holes in the anonymity of a system might not behave like text book discrete memoryless channels.

Zhu and Bettati proposed in [ZB05] a definition of anonymity based on mutual information. The notion we consider is based on capacity, which is an abstraction of mutual information obtained by maximizing over the possible input distributions. As a consequence, we get a measure that depends only on the protocol (i.e. the channel) and not on the users (i.e. the input distribution), which is an advantage because in general we don't know the input distribution, and it also depends on the users, and even with the same users, it may change over time. Of course, in case we know a priori the input distribution, then the definition of Zhu and Bettati is more precise because it gives the exact loss of anonymity for the specific situation.

Another approach close in spirit to ours is the one of [DPW06]. In this work, the authors use the notion of relative entropy to perform a metric analysis of anonymity. In our work, we use the notion of mutual information, which is a special case of relative entropy. However, the specific application of relative entropy in [DPW06] is radically different from ours. We use it to compare the entropy of the input of an anonymity protocol before and after the observation. They use it to establish a sort of distance between the traces of an anonymity system.

In the field of information flow and non-interference there is a line of research which is closely related to ours. There have been various works [McL90, Gra91, CHM01, CHM05, Low02] in which the *high information* and the *low information* are seen as the input and output respectively of a channel. From an abstract point of view, the setting is very similar; technically it does not matter what kind of information we are trying to conceal, what is relevant for the analysis is only the probabilistic relation between the input and the

output information. We believe that part of our framework and of our results are applicable more or less directly also to the field of non-interference. Some of the results however, for instance those based on the hypotheses of symmetry or weak symmetry of the protocol's matrix, seem to be specific to the anonymity setting, in the sense that the assumptions would be too restrictive for the non-interference case.

Eight

A monotonicity principle and its implications for binary channels

In the previous chapter we saw that we can view anonymity systems as noisy channels in the information theoretic sense, and measure the loss of anonymity of a protocol as the capacity of the corresponding channel. As a consequence, the study of channels can provide us with new insight and results about anonymity protocols. In particular we would like to compare channels and define orders with respect to which the capacity is monotone. This would allow us to compare different instances of protocols or a protocol and its specification. Moreover, since capacity is usually difficult to compute and reason about, we would like to have bounds based on easily computable functions.

In this chapter we establish a monotonicity principle for convex functions: a convex function decreases on a line segment iff it assumes its minimum value at the end of that line segment. Though quite simple, this single idea has an unusual number of important consequences for information theory, since the capacity has the important property of being *convex* as a function of the channel matrix. We have already seen a use of this property in Section 7.4, in this chapter we use it extensively, together with the monotonicity principle, to obtain a number of general results.

In the rest of the chapter we show various implications of the monotonicity principle for binary channels. The first of these is that it offers a significant extension of algebraic information theory [MMA06]: a new partial order is introduced on binary channels with respect to which capacity is monotone. This new order is much larger than the interval order considered in [MMA06], and can be characterized in at least three different ways, each of which has its own value: by means of a simple formula, which makes it easy to apply in practice; geometrically, which makes it easy to understand and reason about; and algebraically, which establishes its canonical nature, mathematically speaking.

Another use of the monotonicity principle is in establishing inequalities relating different measurements on the domain of channels. These inequalities can be used to provide bounds for the capacity of a channel in cases where only partial information is known about the channel, or where the channel matrix depends on run-time parameters of the protocol. These results also provide graphical methods for reasoning about the capacity of channels. There is a

"geometry of binary channels", in which, roughly speaking, a line of channels either hits the diagonal, or is parallel to it. We determine the behavior of capacity in both these cases, which allows one to answer most (but not all) questions when it comes to comparing channel behavior.

The results in this chapter are from joint work with Keye Martin and will appear in the forthcoming paper ([CM07]) which contains additional results such as an explanation of the relation between capacity and Euclidean distance and the solution of an open problem in quantum steganography.

8.1 The monotonicity principle

The monotonicity principle introduced in this section is based on the property of convexity (see Section 2.3 for a brief discussion on convexity). A function $f : S \to \mathbb{R}$ defined on a convex set S is *convex* iff

$$tf(x_1) + \bar{t}f(x_2) \geq f(tx_1 + \bar{t}x_2) \qquad \forall x_1, x_2 \in S, \forall t \in [0,1]$$

where $\bar{t} = 1 - t$. A function f is *strictly convex* if $tf(x_1) + \bar{t}f(x_2) = f(tx_1 + \bar{t}x_2)$ for $x_1 \neq x_2$ implies $t = 0$ or $t = 1$.

We now come to *the monotonicity principle*: a convex function decreases along a line segment iff it assumes its minimum value at the end of that line segment.

Theorem 8.1.1. *If S is a set of vectors, $x, y \in S$, $\pi(t) = ty + \bar{t}x$ is the line from x to y and $c : S \to \mathbb{R}$ is a function (strictly) convex on $\pi[0,1]$, then the following are equivalent:*

(i) *The function $c \circ \pi : [0,1] \to \mathbb{R}$ is (strictly) monotone decreasing,*

(ii) *The minimum value of $c \circ \pi$ on $[0,1]$ is $c(\pi(1)) = c(y)$.*

Proof. (ii) $\Rightarrow$ (i). The function $f : [0,1] \to \mathbb{R} :: f(t) = c(\pi(t))$ is convex, since c is convex on $\pi[0,1]$ and π satisfies $\pi(px + \bar{p}y) = p\pi(x) + \bar{p}\pi(y)$, $p \in [0,1]$. Let $0 \leq s < t \leq 1$. We prove $f(s) \geq f(t)$. Since t is between s and 1, we have $t = p \cdot s + \bar{p} \cdot 1$, where $p = \bar{t}/\bar{s} \in [0,1)$. Then

$$\begin{aligned} f(t) &\leq pf(s) + \bar{p}f(1) && \text{(convexity of } f\text{)} && (8.1)\\ &\leq pf(s) + \bar{p}f(s) && (f(1) \leq f(s)) && (8.2)\\ &= f(s) \end{aligned}$$

Then suppose that c is strictly convex on $\pi[0,1]$, so f is also strictly convex on $[0,1]$. We want to show that f is strictly monotone decreasing, that is $f(s) > f(t)$ (since $s < t$). Assuming $f(s) = f(t)$ then we have equality in (8.1) and from strict convexity this implies $p = 0$ (since $p < 1$). Then the equality in (8.2) implies $f(s) = f(1)$. Then we take any point $r \in (s,1)$ which can be written as $r = q \cdot s + \bar{q} \cdot 1$, $q = \bar{r}/\bar{s} \in (0,1)$ and by strict convexity we have:

$$f(r) < qf(s) + \bar{q}f(1) = f(1)$$

which is a contradiction since $f(1)$ is the minimum of f. □

It is by no means obvious that the monotonicity principle is of any value in problem solving. However, as we will see shortly, there are many situations in information theory where it is far easier to establish a minimum value along a line than it is to establish monotonicity itself. Then the monotonicity principle can be applied since many of the functions involved in information theory turn out to be convex.

Let $(\mathcal{A}, \mathcal{O}, \mathbf{m})$ be a discrete channel and p a distribution over $\mathcal{A}$. We denote by $I_p(\mathbf{m})$ the mutual information between the input and the output of the channel, for the given p. The next result appears as Theorem 2.7.4 in the book by Cover and Thomas ([CT91]):

Theorem 8.1.2. *The mutual information $I_p(\mathbf{m})$ is a convex function of $\mathbf{m}$ for a fixed p.*

An important consequence of the last result, first observed by Shannon in [Sha93], though not particularly well-known, is that capacity itself is convex:

Theorem 8.1.3. *The capacity $c(\mathbf{m})$ is a convex function of $\mathbf{m}$.*

Proof. Let p_1, p_2, p be the capacity achieving distributions of the channels $\mathbf{m}_1, \mathbf{m}_2$ and $t\mathbf{m}_1 + \bar{t}\mathbf{m}_2$ respectively. We have

$$\begin{array}{rcll} tc(\mathbf{m}_1) + \bar{t}c(\mathbf{m}_2) & = & tI_{p_1}(\mathbf{m}_1) + \bar{t}I_{p_2}(\mathbf{m}_2) & \text{definition of } c(\mathbf{m}) \\ & \geq & tI_p(\mathbf{m}_1) + \bar{t}I_p(\mathbf{m}_2) & p_i \text{ gives the best } I_{p_i}(\mathbf{m}_i) \\ & \geq & I_p(t\mathbf{m}_1 + \bar{t}\mathbf{m}_2) & \text{Theorem 8.1.2} \\ & = & c(t\mathbf{m}_1 + \bar{t}\mathbf{m}_2) & \text{definition of } c(\mathbf{m}) \end{array}$$

□

Because Theorem 8.1.1 can be applied to *any* line that ends on a minimum capacity channel, it provides a powerful technique for comparing the capacity of channels. One immediate application of it is that we can solve the *capacity reduction problem* for arbitrary $m \times n$ channels. In the capacity reduction problem, we have an $m \times n$ channel x[1] and would like to systematically obtain a channel whose capacity is smaller by some pre-specified amount. The monotonicity principle offers a solution:

Proposition 8.1.4. *Let x be any $m \times n$ channel, y be any $m \times n$ channel with zero capacity and π denote the line from x to y. Then $c(\pi[0,1]) = [0, c(x)]$ and the function $c \circ \pi$ is monotone decreasing.*

Proof. By the continuity of capacity [Mar07], $c(\pi[0,1])$ is an interval that contains 0 and $c(x)$, which means $[0, c(x)] \subseteq c(\pi[0,1])$. Since $c(y) = 0 = \min c \circ \pi$, Theorem 8.1.1 implies that $c \circ \pi$ is decreasing, so $c(\pi[0,1]) \subseteq [0, c(x)]$. □

Thus, given any $0 < r < c(x)$, we need only solve the equation $c(\pi(t)) = r$ for t. This equation can be solved numerically since $c \circ \pi - r$ changes sign on $[0,1]$. Notice that this enables us to systematically solve a problem that otherwise would have $m(n-1)$ unknowns but only a single equation. Moreover,

[1] We will represent discrete channels by their probability matrices, here x is a $m \times n$ matrix.

the channel obtained is a linear degradation of the original. Similarly, we can systematically increase the capacity using the line from x to a maximum capacity channel.

In the rest of this chapter we will see many more implications of the monotonicity property for the family of binary channels.

8.2 Binary channels

A binary channel is a discrete channel with two inputs ("0" and "1") and two outputs ("0" and "1"). An input is sent through the channel to a receiver. Because of noise in the channel, what arrives may not necessarily be what the sender intended. The effect of noise on input data is modeled by a noise matrix u. If data is sent through the channel according to the distribution x, then the output is distributed as $y = x \cdot u$. The noise matrix u is given by

$$u = \begin{pmatrix} a & \bar{a} \\ b & \bar{b} \end{pmatrix}$$

where $a = P(0|0)$ is the probability of receiving 0 when 0 is sent and $b = P(0|1)$ is the probability of receiving 0 when 1 is sent.

Thus, the noise matrix of a binary channel can be represented by a point (a, b) in the unit square $[0, 1]^2$ and all points in the unit square represent the noise matrix of some binary channel.

Definition 8.2.1. The set of *binary channels* is $[0, 1]^2$.

The *composition* of two binary channels x and y is the channel whose noise matrix is the usual product of matrices $x \cdot y = xy$. The multiplication of two noise matrices $x = (a, b)$ and $y = (c, d)$ in the unit square representation is

$$(a, b) \cdot (c, d) = (\ a(c - d) + d,\ b(c - d) + d\ ) = c(a, b) + d(\bar{a}, \bar{b})$$

where the expression to the right uses scalar multiplication and addition of vectors. By contrast, the representation for a convex sum of noise matrices is simply the convex sum of each representing vector.

A *monoid* is a set with an associative binary operation that has an identity. The set of binary channels is a monoid under the operation of multiplication whose identity is the noiseless channel $1 := (1, 0)$. A binary channel can be classified according to the sign of its determinant, $\det(a, b) = a - b$, which defines a homomorphism $\det : ([0, 1]^2, \cdot) \to ([-1, 1], \cdot)$ between monoids.

Definition 8.2.2. A binary channel x is called *positive* when $\det(x) > 0$, *negative* when $\det(x) < 0$ and a *zero channel* when $\det(x) = 0$. A channel is *non-negative* if it is either positive or zero.

Also, a channel (a, b) is called a *Z-channel* if $a \in \{0, 1\}$ or $b \in \{0, 1\}$.

Notice that $\det(x) \in (0, 1]$ for positive channels, and that $\det(x) \in [-1, 0)$ for negative channels. Thus, the set of positive channels is a submonoid of $[0, 1]^2$ as is the set of non-negative channels; the determinant is a homomorphism from the non-negative channels into $([0, 1], \cdot)$.

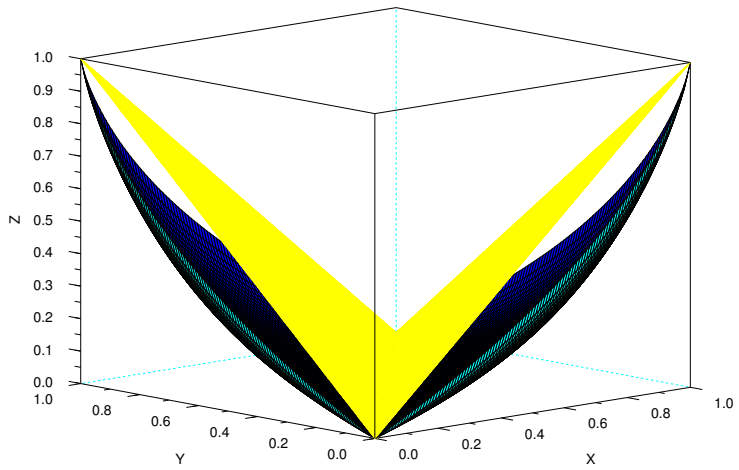


Figure 8.1: The capacity (lower graph) and the determinant (upper graph) for binary channels.

Definition 8.2.3. The set of non-negative binary channels is denoted $\mathbb{N}$. The set of positive binary channels is denoted $\mathbb{P}$.

A nice property of positive binary channels is that composition can be inverted (even though the "inverse" of a channel is not a channel).

Lemma 8.2.4. *For $a \in \mathbb{P}$, $x, y \in \mathbb{N}$ we have $ax = ay$ iff $x = y$ iff $xa = ya$.*

Proof. Seeing a, x, y as matrices, $det(a) > 0$ so a can be inverted (note that a^{-1} is not a channel) so $ax = ay \Leftrightarrow a^{-1}ax = a^{-1}ay \Leftrightarrow x = y$. Similarly for $xa = ya$. □

The amount of information that may be sent through a channel (a, b) is given by its capacity

$$c(a,b) = \sup_{x \in [0,1]} H((a-b)x + b) - xH(a) - (1-x)H(b)$$

This defines a continuous function on the unit square [Mar07], given by

$$c(a,b) \quad = \log_2\left(2^{\frac{\bar{a}H(b) - \bar{b}H(a)}{a-b}} + 2^{\frac{bH(a) - aH(b)}{a-b}}\right)$$

where $c(a,a) := 0$ and $H(x) = -x\log_2(x) - (1-x)\log_2(1-x)$ is the base two entropy. A graph of the capacity and the determinant for binary channels is displayed in Figure 8.1.

Another interesting property of binary channels is that the capacity is *strictly* convex everywhere, except on the zero channels. To show this we will adjust the proof of convexity from [CT91], focusing on equality conditions. We start by the log sum inequality.

Theorem 8.2.5 (Log sum inequality)**.** *For non-negative numbers* $a_1, \dots, a_n$ *and* $b_1, \dots, b_n$

$$\sum_{i=1}^{n} a_i \log \frac{a_i}{b_i} \geq \big(\sum_{i=1}^{n} a_i\big) \log \frac{\sum_{i=1}^{n} a_i}{\sum_{i=1}^{n} b_i}$$

with equality if and only if $\frac{a_i}{b_i}$ *is constant.*

We refer to [CT91] for the proof. We use the conventions $0 \log 0 = 0, a \log \frac{a}{0} = \infty,\ a > 0$ and $0 \log \frac{0}{0} = 0$ that are justified by continuity.

Theorem 8.2.6. *Capacity on binary channels is strictly convex everywhere except on the zero channels. That is, given* $u_1, u_2 \in [0,1]^2$, $u_1 \neq u_2$ *and* $t \in (0,1)$, *we have*

$$c(tu_1 + \bar{t}u_2) \leq tc(u_1) + \bar{t}c(u_2)$$

with equality if and only if both u_1, u_2 *are zero channels.*

Proof. We already know that c is convex from Theorem 8.1.3, we will only focus on the equality condition. First we show that the mutual information $I_p(u)$ is strictly convex everywhere except on the zero channels, for a fixed nonzero p. Let $u_1 = (a_1, b_1), u_2 = (a_2, b_2)$ and $u_t = tu_1 + \bar{t}u_t = (a_t, b_t)$. We denote by $p_1(y|x), p_2(y|x), p_t(y|x)$ the conditional distributions of u_1, u_2, u_t where

$$p_1(0|0) = a_1 \qquad p_1(0|1) = b_1 \qquad p_1(1|0) = \bar{a}_1 \qquad p_1(1|1) = \bar{b}_1$$

and similarly for the others. Given a nonzero input distribution $p(x)$, we denote by $p_1(x,y), p_2(x,y), p_t(x,y)$ the corresponding joint distributions and $p_1(y), p_2(y), p_t(y)$ the corresponding marginals. It is easy to see that

$$\begin{aligned} p_t(x,y) &= tp_1(x,y) + \bar{t}p_2(x,y) \text{ and} \\ p_t(y) &= tp_1(y) + \bar{t}p_2(y) \end{aligned}$$

From the log sum inequality we have

$$\begin{aligned} p_t(x,y) \log \frac{p_t(x,y)}{p(x)p_t(y)} = \\ \big(tp_1(x,y) + \bar{t}p_2(x,y)\big) \log \frac{tp_1(x,y) + \bar{t}p_2(x,y)}{tp(x)p_1(y) + \bar{t}p(x)p_2(y)} \leq \\ tp_1(x,y) \log \frac{tp_1(x,y)}{tp(x)p_1(y)} + \bar{t}p_2(x,y) \log \frac{\bar{t}p_2(x,y)}{\bar{t}p(x)p_2(y)} \end{aligned}$$

By summing over all $x, y \in \{0,1\}$ and by definition of mutual information (2.1) we get

$$I_p(u_t) \leq tI_p(u_1) + \bar{t}I_p(u_2)$$

with equality iff

$$\frac{tp_1(x,y)}{tp(x)p_1(y)} = \frac{\bar{t}p_2(x,y)}{\bar{t}p(x)p_2(y)} \overset{p(x)\neq 0}{\Rightarrow} \frac{p_1(y|x)}{p_1(y)} = \frac{p_2(y|x)}{p_2(y)} \tag{8.3}$$

for all $x, y \in \{0,1\}$. Assuming that all the elements of u_1, u_2 are nonzero, we have:

$$\frac{a_1}{b_1} = \frac{a_2}{b_2} \quad \text{and} \quad \frac{\bar{a}_1}{\bar{b}_1} = \frac{\bar{a}_2}{\bar{b}_2} \tag{8.4}$$

Letting $k = \frac{a_1}{b_1}$, we get from the right-hand side:

$$\begin{aligned}(1-a_1)(1-b_2) = (1-a_2)(1-b_1) \Rightarrow \\ kb_1 + b_2 = kb_2 + b_1 \Rightarrow \\ b_1(k-1) = b_2(k-1)\end{aligned}$$

from which we get $b_1 = b_2$ or $k = 1$ and since we assumed $u_1 \neq u_2$ we have $k = 1$ and as a consequence $a_1 = b_1$ and $a_2 = b_2$.

Now consider the case $b_1 = 0$. If $a_1 > 0$ then $p_1(0) > 0$ and from (8.3) we get $b_2 = 0$ and from the right side of (8.4) we get $a_1 = a_2 = 1$, which is impossible since we assumed $u_1 \neq u_2$. If $a_1 = 0$ then from (8.4) we get $a_2 = b_2$ so the statement holds. Similarly for $b_1 = 1$ and the other extreme cases.

Finally let p_1, p_2, p be the capacity achieving distributions of the channels u_1, u_2, u_t respectively, we have

$$\begin{array}{rcll} tc(u_1) + \bar{t}c(u_2) & = & tI_{p_1}(u_1) + \bar{t}I_{p_2}(u_2) & \text{definition of } c(u) \\ & \geq & tI_p(u_1) + \bar{t}I_p(u_2) & p_i \text{ gives the best } I_{p_i}(u_i) \\ & \geq & I_p(tu_1 + \bar{t}u_2) & \text{Theorem 8.1.2} \\ & = & c(tu_1 + \bar{t}u_2) & \text{definition of } c(u) \end{array}$$

Suppose that equality holds. This means that $I_{p_i}(u_i) = I_p(u_i)$, that is p is a capacity achieving distribution for both u_1, u_2. Also it means that $I_p(u_t) = tI_p(u_1) + \bar{t}I_p(u_2)$ which (assuming that p is nonzero) implies that u_1, u_2 are zero channels. If $p(0)$ or $p(1)$ is zero, and since p is the capacity achieving distribution, then the capacity of all u_1, u_2, u_t is zero, in other words they are zero channels. □

The equality $c(tu_1 + \bar{t}u_2) = tc(u_1) + \bar{t}c(u_2)$ essentially means that the capacity is linear between u_1 and u_2. The above theorem states that this only happens along the line of zero channels, as can be clearly seen in the graph of Figure 8.1.

8.3 Relations between channels

In this section, we consider partial orders on binary channels with respect to which capacity is monotone. Their importance stems from the fact that a statement like "$x \leq y$" is much easier to verify than a statement like "$c(x) \leq c(y)$". This is particularly useful in situations where the noise matrix of a channel depends on some parameter of the protocol, like the distribution of the coins in the Dining Cryptographers or the probability of forwarding in Crowds, allowing us to provide bounds for large classes of protocol instances.

8.3.1 Algebraic information theory

Algebraic information theory uses the interplay of order, algebra and topology to study communication. In [MMA06] it is shown that the interval domain with the inclusion order can be used to fruitfully reason about binary channels.

Recall that a *partial order* on a set is a relation which is reflexive, transitive and antisymmetric.

Definition 8.3.1. The *interval domain* is the set of non-negative binary channels $(\mathbb{N}, \sqsubseteq)$ together with the partial order $\sqsubseteq$ defined by

$$x \sqsubseteq y \quad \text{iff} \quad b \leq d \;\; \& \;\; c \leq a,$$

for $x = (a, b) \in \mathbb{N}$ and $y = (c, d) \in \mathbb{N}$. The natural measurement $\mu : \mathbb{N} \to [0, 1]^*$ is given by

$$\mu x = \det(x) = a - b$$

where $x = (a, b) \in \mathbb{N}$.

This is not the usual notation in domain theory for the interval domain, but experience has taught us that this is the simplest way of handling things in the context of information theory. The following result is proved in [MMA06]:

Theorem 8.3.2. *Let $(\mathbb{N}, \cdot)$ denote the monoid of non-negative channels.*

- *The right zero elements of $\mathbb{N}$ are precisely the zero channels,*
- *The maximally commutative submonoids of $\mathbb{N}$ are precisely the lines which join the identity to a zero channel,*
- *For any maximally commutative submonoid $\pi \subseteq \mathbb{N}$,*

$$(\forall x, y \in \pi)\; x \sqsubseteq y \;\Leftrightarrow\; \mu x \geq \mu y \;\Leftrightarrow\; cx \geq cy$$

- *Capacity $c : \mathbb{N} \to [0, 1]^*$ is monotone: if $x \sqsubseteq y$, then $c(x) \geq c(y)$.*

We will now see that the monotonicity principle offers a new order $\leq$ on channels that leads to a clear and significant extension of algebraic information theory.

8.3.2 A new partial order on binary channels

By the monotonicity principle, capacity decreases along any line that ends on a zero capacity channel. This suggests a new way of ordering positive channels:

Definition 8.3.3. For two positive channels $x = (a, b)$ and $y = (c, d)$,

$$x \leq y \;\equiv\; c \cdot \mu x \geq a \cdot \mu y \quad \text{and} \quad \bar{c} \cdot \mu x \geq \bar{a} \cdot \mu y$$

Proposition 8.3.4.

(i) *The relation $\leq$ is a partial order on the set $\mathbb{P}$ of positive channels,*

(ii) *For $x, y \in \mathbb{P}$, if $x \sqsubseteq y$, then $x \leq y$. In particular, the least element of $(\mathbb{P}, \leq)$ is the identity channel $\perp = (1, 0)$,*

(iii) *For $x, y \in \mathbb{P}$, we have $x \leq y$ iff there is a line segment that begins at x, passes through y and ends at some point of $\{(t, t) : t \in [0, 1]\}$,*

(iv) *Capacity $c : \mathbb{P} \to [0, 1]^*$ is strictly monotone: if $x \leq y$, then $c(x) \geq c(y)$ with equality iff $x = y$.*

Proof. (i) Reflexivity is immediate from the definition of $\leq$. For antisymmetry we first notice that

$$x \leq y \wedge \mu x = \mu y \Rightarrow x = y \tag{8.5}$$

Then assuming $x \leq y$ and $y \leq x$ we have $c \cdot \mu x = a \cdot \mu y$ and $\bar{c} \cdot \mu x = \bar{a} \cdot \mu y$ from which we conclude that $\mu x = \mu y$ thus $x = y$ (by (8.5)).

For transitivity, assume $x \leq y$ and $y \leq z$. Write $x = (a, b)$, $y = (c, d)$ and $z = (e, f)$. Then we have

$$\left.\begin{array}{l} c \cdot \mu x \geq a \cdot \mu y \\ e \cdot \mu y \geq c \cdot \mu z \end{array}\right\} \Rightarrow\ c \cdot \mu x \geq \frac{ac}{e} \mu z \ \Rightarrow\ e \cdot \mu x \geq a \cdot \mu z$$

(if $e = 0$ or $c = 0$ we can easily get the same result). Similarly we can show that $\bar{e} \cdot \mu x \geq \bar{a} \cdot \mu z$ thus $x \leq z$, establishing transitivity.

(ii) Write $x = (a, b)$ and $y = (c, d)$. Assume $x \sqsubseteq y$ thus $c \leq a$ and $b \leq d$. By subtracting the two we get $\mu x \geq \mu y$. Then we have $c \leq a$ thus $\bar{c} \geq \bar{a}$, which gives $\bar{c} \cdot \mu x \geq \bar{a} \cdot \mu y$. Finally from $c \leq a$, $b \leq d$ we get

$$\begin{aligned} ad &\geq cb \Rightarrow \\ ca - cb &\geq ca - ad \Rightarrow \\ c(a - b) &\geq a(c - d) \end{aligned}$$

thus $c \cdot \mu x \geq a \cdot \mu y$ which gives $x \leq y$.

(iii) First, assume $x \leq y$ thus

$$c \cdot \mu x \geq a \cdot \mu y \tag{8.6}$$

$$\bar{c} \cdot \mu x \geq \bar{a} \cdot \mu y \tag{8.7}$$

The case $x = y$ is trivial. If $x \neq y$ then by adding (8.6) and (8.7) we get $\mu x \geq \mu y$ and by (8.5) we get $\mu x > \mu y$. so the expression

$$\alpha := \frac{c \cdot \mu x - a \cdot \mu y}{\mu x - \mu y}$$

is well-defined. By (8.6) we have $\alpha \geq 0$ and by (8.7) we get $\alpha \leq 1$. Thus (α, α) is a zero channel. The line from x to (α, α), given by $\pi(t) = (1 - t)x + t(\alpha, \alpha)$ for $t \in [0, 1]$, passes through y since

$$\pi\left(\frac{\mu x - \mu y}{\mu x}\right) = y$$

which finishes the proof in this direction.

Conversely, suppose there is a line $\pi(t) = (1 - t)x + t(\alpha, \alpha)$ from x to a zero channel $(\alpha, \alpha) \in [0, 1]^2$ that passes through y. Then for some $s \in [0, 1]$, $\pi(s) = y$. Writing $y = (c, d)$, this value of s satisfies

$$s\alpha + (1 - s)a = c \ \ \& \ \ s\alpha + (1 - s)b = d$$

Subtracting the second equation from the first, $(1 - s) = \mu y / \mu x \in [0, 1]$, so $\mu x \geq \mu y$. If $\mu x = \mu y$, then $s = 0$, which gives $\pi(s) = y = \pi(0) = x$ and hence $x \leq y$. Otherwise, $\mu x > \mu y$, and from the first equation relating s and α,

$$\alpha = \frac{c \cdot \mu x - a \cdot \mu y}{\mu x - \mu y}$$

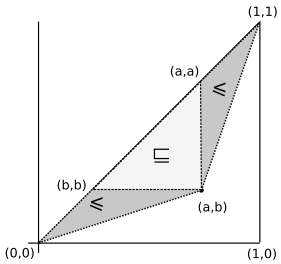


Figure 8.2: Geometric representation of $\sqsubseteq, \leq$.

Since $0 \leq \alpha \leq 1$ we have that (8.6),(8.7) both hold and hence $x \leq y$.

(iv) Strict monotonicity follows from (iii) and Theorems 8.1.1 and 8.2.6. □

Notice that the monotonicity of capacity on $(\mathbb{N}, \sqsubseteq)$, given in Theorem 8.3.2, is now a trivial consequence of (ii) and (iv) in Proposition 8.3.4, showing also that capacity is *strictly* monotone wrt $\sqsubseteq$.

8.3.3 The coincidence of algebra, order and geometry

Each order is given by a simple formula that is easy to verify in practice: for $x = (a, b) \in \mathbb{P}$ and $y = (c, d) \in \mathbb{P}$,

- $x \sqsubseteq y$ iff $b \leq d$ and $c \leq a$,
- $x \leq y$ iff $c \cdot \mu x \geq a \cdot \mu y$ and $\bar{c} \cdot \mu x \geq \bar{a} \cdot \mu y$.

Each also has a clear geometric significance which makes it easy to reason about: for $x = (a, b) \in \mathbb{P}$ and $y \in \mathbb{P}$,

- $x \sqsubseteq y$ iff y is contained in the triangle with vertices $\{(a, a), x, (b, b)\}$ iff there is a line segment from x to a point of $\{(t, t) : t \in [b, a]\}$ that passes through y.
- $x \leq y$ iff y is contained in the triangle with vertices $\{(0, 0), x, (1, 1)\}$ iff there is a line segment from x to a point of $\{(t, t) : t \in [0, 1]\}$ that passes through y.

A geometric interpretation of these orders is shown in Figure 8.2. Remarkably, each of these orders can also be characterized algebraically:

Lemma 8.3.5. *For $x, y \in \mathbb{P}$,*

(i) $x \sqsubseteq y$ *iff* $(\exists z \in \mathbb{P})\, zx = y$,

(ii) $x \leq y$ *iff* $(\exists z \in \mathbb{P})\, xz = y$.

Proof. (i) Write $x = (a, b)$. If $x \sqsubseteq y$, then $x \leq y$, so by Prop. 8.3.4(ii), there is a line segment $\pi : [0, 1] \to \mathbb{N} :: \pi(s) = (1 - s)x + s(\alpha, \alpha)$ with $\pi(t) = y$ for some $t \in [0, 1]$. Define $z = (c, d)$ by

$$c := 1 + t \cdot \frac{\alpha - a}{a - b} \quad \& \quad d := t \cdot \frac{\alpha - b}{a - b}$$

Because $x \sqsubseteq y$, $b \leq \alpha \leq a$, which ensures that $c, d \in [0,1]$, so that z is a channel. Moreover, z is positive since $\det(z) = c - d = 1 - t > 0$, which holds since $y = \pi(t) \in \mathbb{P}$ and $\pi(1) = (\alpha, \alpha) \notin \mathbb{P}$. Finally,

$$zx = (\bar{t}a + t\alpha, \bar{t}b + t\alpha) = \pi(t) = y$$

which finishes this direction. Conversely, if there is $z \in \mathbb{P}$ with $zx = y$, then it is straightforward to verify that $x \sqsubseteq zx$, so $x \sqsubseteq y$.

(ii) Write $x = (a, b)$. If $x \leq y$, then by Prop. 8.3.4(ii), there is a line segment $\pi : [0,1] \to \mathbb{N} :: \pi(s) = (1-s)x + s(\alpha, \alpha)$ with $\pi(t) = y$ for some $t \in [0,1]$. First notice that $t < 1$ since $\pi(t) = y \in \mathbb{P}$ and $\pi(1) \notin \mathbb{P}$. Define $z = (c, d)$ by

$$c := (1-t) + \alpha t \quad \& \quad d := \alpha t$$

We have $c, d \in [0,1]$ because $\alpha, t \in [0,1]$ and $\det(z) = 1 - t > 0$ since $t < 1$, so z is a positive channel. Finally,

$$xz = (c-d)x + d(1,1) = (1-t)x + t(\alpha, \alpha) = y$$

which finishes this direction. Conversely, suppose there is $z \in \mathbb{P}$ with $xz = y$. Write $z = (c, d)$. If $z = (1, 0)$, then $x = zx = y$ and we are done, so we can assume $\det(z) = c - d < 1$, which lets us define

$$\alpha := \frac{d}{1 - \det(z)} \in [0,1] \quad \& \quad t := 1 - \det(z) \in [0,1]$$

Because $y = xz = (1-t)x + t(\alpha, \alpha)$, we know that y lies on the line segment from x to (α, α), which by Prop. 8.3.4(ii) implies $x \leq y$. □

Thus, despite the somewhat awkward formulation of $\leq$ given in Definition 8.3.3, we see that $\leq$ is nevertheless quite natural. In fact, from the point of view of information theory, it is more natural than $\sqsubseteq$:

Theorem 8.3.6. *Let $(\mathbb{P}, \cdot, 1)$ denote the monoid of positive binary channels.*

(i) *The relation*

$$x \leq y \equiv (\exists z \in \mathbb{P})\, xz = y$$

defines a partial order on $\mathbb{P}$ with respect to which capacity $c : \mathbb{P} \to [0,1]^*$ is strictly monotone,

(ii) *The operator $l_x : \mathbb{P} \to \mathbb{P} :: l_x(y) = xy$ is monotone with respect to $\leq$,*

(iii) *The operator $r_x : \mathbb{P} \to \mathbb{P} :: r_x(y) = yx$ is monotone with respect to $\leq$.*

Proof. (i) Follows from Lemma 8.3.5(ii) and Prop. 8.3.4. For (ii), let $a \leq b$. By Lemma 8.3.5(ii), there is $c \in \mathbb{P}$ with $b = ac$. Then

$$l_x(b) = xb = x(ac) = (xa)c = l_x(a)c$$

so by Lemma 8.3.5(ii), we have $l_x(a) \leq l_x(b)$.

(iii) Let $a \leq b$. By Lemma 8.3.5(ii), there is $c \in \mathbb{P}$ with $b = ac$. Proceeding as in the proof of (ii),

$$r_x(b) = bx = (ac)x$$

which does not appear to help much. However, by Lemma 8.3.5(i), $x \sqsubseteq cx$, and hence $x \leq cx$ by Proposition 8.3.4. Thus, by Lemma 8.3.5(ii), there is $z \in \mathbb{P}$ with $xz = cx$, so

$$r_x(b) = (ac)x = a(cx) = a(xz) = (ax)z = r_x(a)z$$

which means that $r_x(a) \leq r_x(b)$ by Lemma 8.3.5(ii). □

By contrast, r_x is monotone with respect to $\sqsubseteq$, but l_x is not. The reason for this difference is that $\mathbb{P} \cdot x \subseteq x \cdot \mathbb{P}$ holds for all $x \in \mathbb{P}$, and this inclusion is strict. So even though $\mathbb{P}$ is not commutative, it has a special property commutative monoids have which ensures that both l_x and r_x are monotone with respect to $\leq$. The monotonicity of l_x and r_x implies that

$$(\forall\, a, b, x, y \in \mathbb{P})\; x \leq y \Rightarrow c(axb) \geq c(ayb)$$

with equality iff $x = y$ since $axb \leq ayb$ and $c(axb) = c(ayb)$ implies $axb = ayb$ which from Lemma 8.2.4 implies that $x = y$. The above inequality, in turn, has an important and new consequence for information theory:

Corollary 8.3.7. *For all $a, b, x, y \in \mathbb{P}$,*

$$c(axyb) \leq \min\{c(axb), c(ayb)\}$$

with equality iff $x = 1$ or $y = 1$.

Proof. Since $1 \leq x$, we can multiply on the right by y to get $y \leq xy$. Similarly, $x \leq xy$. Since $x, y \leq xy$, we can multiply on the left by a to get $ax \leq axy$ and $ay \leq axy$, and then multiply on the right by b to get $axb \leq axyb$ and $ayb \leq axyb$. The result now follows from the monotonicity of capacity. From strict monotonicity, if $c(axyb) = c(axb)$ then $axyb = axb$ and from Lemma 8.2.4 $y = 1$. Similarly $c(axyb) = c(ayb) \Leftrightarrow x = 1$. □

In particular, for $a = b = 1$, the well-known inequality $c(xy) \leq \min\{c(x), c(y)\}$ follows. It is interesting indeed that it may be derived from an order which itself may be derived from algebraic structure. This illustrates the value of knowing about the coincidence of algebra, order and geometry.

8.4 Relations between monotone mappings on channels

Having just considered relations between binary channels, we now turn to relations between monotone mappings on binary channels. Of particular interest is the fascinating relationship between capacity and Euclidean distance.

8.4.1 Algebraic relations

Both capacity and Euclidean distance are invariant under multiplication by the idempotent $e = (0, 1)$:

Lemma 8.4.1. *Let $e := (0, 1)$.*

(i) *For any $(a, b) \in [0, 1]^2$,*

$$e \cdot (a, b) = (b, a) \quad \& \quad (a, b) \cdot e = (\bar{a}, \bar{b}),$$

(ii) *For any* $x \in [0,1]^2$, $c(ex) = c(xe) = c(x)$, *and*

(iii) *For any* $x \in [0,1]^2$, $|\det(ex)| = |\det(xe)| = |\det(x)|$.

We now establish our first result which relates capacity to distance:

Theorem 8.4.2. *For two binary channels* $x, y \in [0,1]^2$,

$$c(xy) \leq \min\{\ c(x)|\det(y)|, |\det(x)|c(y)\ \}.$$

with equality iff x *(or* y*) is* 1, e *or a zero channel.*

Proof. First assume that $x, y \in \mathbb{N}$. Write $x = (a, b)$ and $y = (c, d)$. The product xy can be written as a convex sum in two different ways:

$$xy = (c-d)(a,b) + d(1,1) + c(0,0) = \det(y)(a,b) + d(1,1) + \bar{c}(0,0) \tag{8.8}$$

and

$$xy = (a-b)(c,d) + b(c,c) + \bar{a}(d,d) = \det(x)(c,d) + b(c,c) + \bar{a}(d,d) \tag{8.9}$$

The inequality now follows for $x, y \in \mathbb{N}$ by applying the convexity of capacity to each expression for xy. By Theorem 8.2.6 the equality in (8.8) holds iff one of the convex coefficients is 1 or all convexly added channels are zero channels. That is, iff $det(y) = 1 \Rightarrow y = 1$ or $(a, b) = x$ is a zero channel (note that $d, \bar{c}$ cannot be 1). Similarly, we have equality in (8.9) iff $x = 1$ or y is a zero channel.

To finish the proof, we now consider the three remaining cases: (1) $x \notin \mathbb{N}, y \in \mathbb{N}$, (2) $x \in \mathbb{N}, y \notin \mathbb{N}$, (3) $x \notin \mathbb{N}, y \notin \mathbb{N}$. For (1), we use Lemma 8.4.1(ii) and associativity of channel multiplication to get

$$c(xy) = c(e(xy)) = c((ex)y)$$

But the channels ex and y are non-negative, so

$$\begin{aligned} c(xy) &= c((ex)y) \\ &\leq \min\{\ c(ex)|\det(y)|, |\det(ex)|c(y)\ \} \\ &= \min\{\ c(x)|\det(y)|, |\det(x)|c(y)\ \} \end{aligned}$$

where the last equality holds by Lemma 8.4.1(ii) and Lemma 8.4.1(iii). The equality holds if x or y are zero channels, $y = 1$ or $ex = 1 \Rightarrow x = e$. For (2), we write $c(xy)$ as

$$c(xy) = c((xy)e) = c(x(ye))$$

and just as with (1), we see that the desired inequality holds. For (3), we use

$$c(xy) = c(e(xy)) = c((ex)y)$$

which reduces the problem to the case just settled in (2), finishing the proof. □

The last result extends to *any* convex function on $\mathbb{N}$. It gives a new proof of a well-known result in information theory.

Corollary 8.4.3. *For* $x, y \in [0,1]^2$, $c(xy) \leq \min\{c(x), c(y)\}$ *with equality iff* x *(or* y*) is* 1, e *or a zero channel.*

Proof. Simply use the fact that $|\det(x)| \leq 1$. □

It also sheds light on the relation between Euclidean distance and capacity:

Corollary 8.4.4. *For a binary channel $x \in [0,1]^2$, $c(x) \leq |\det(x)|$ with equality iff x is 1, e or a zero channel.*

Proof. By replacing x with ex if necessary, we can assume that $x \in \mathbb{N}$. Now take y to be the identity channel, which has capacity $c(y) = \det(y) = 1$. □

Intuitively, the Euclidean distance $|\det|$ is a canonical upper bound on capacity. Our goal now is to prove this. First, $|\det|$ is determined by its value on the set $\mathbb{N}$ of non-negative channels. Next, as a function on $\mathbb{N}$, it preserves multiplication, convex sum and identity. There are only two functions like this in existence:

Theorem 8.4.5. *If $f : \mathbb{N} \to [0,1]$ is a function such that*

- $f(1) = 1$
- $f(xy) = f(x)f(y)$
- $f(px + \bar{p}y) = pf(x) + \bar{p}f(y)$

then either $f \equiv 1$ or $f = \det$.

Proof. Assume that f is not a constant function, so that $f(x) \neq 1$ for some $x \in \mathbb{N}$. We can now calculate the value of f at a zero channel (α, α):

$$f(\alpha, \alpha) = f(x \cdot (\alpha, \alpha)) = f(x)f(\alpha, \alpha)$$

and since $f(x) < 1$, $(1 - f(x))f(\alpha, \alpha) = 0$ implies that $f(\alpha, \alpha) = 0$. This allows us to determine the value of f along the x-axis since

$$f(a, 0) = f(a \cdot (1, 0) + \bar{a} \cdot (0, 0)) = af(1, 0) + \bar{a}f(0, 0) = a \cdot 1 + \bar{a} \cdot 0 = a$$

and also along the line $a = 1$,

$$f(1, b) = f(\bar{b} \cdot (1, 0) + b \cdot (1, 1)) = \bar{b}f(1, 0) + bf(1, 1) = \bar{b} \cdot 1 + b \cdot 0 = 1 - b$$

Since any non-negative channel $(a, b) \neq (0, 0)$ can be written as a product of Z-channels,

$$(a, b) = (1, b/a) \cdot (a, 0)$$

we have

$$f(a, b) = f((1, b/a) \cdot (a, 0)) = f(1, b/a) \cdot f(a, 0) = (1 - b/a)a = a - b = \det(a, b)$$

and are finished. □

Thus, there is only one nontrivial convex-linear homomorphism above capacity: the determinant. This raises the question of how close in value the two are.

8.4.2 Inequalities

In the formulation of $\leq$ given in Definition 8.3.3, the case $\mu x = \mu y$ is specifically excluded i.e. channels that lie on a line of constant determinant do not compare with respect to $\leq$ unless they are equal. The behavior of capacity on such lines is more involved than it is for lines that hit the diagonal. We now turn to this important special case, and once again, find the monotonicity principle indispensable.

Consider a line in $\mathbb{N}$ of fixed determinant, that is, a line joining the Z-channels $(d, 0)$ and $(1, 1-d)$:

$$\pi_d(t) = t(1, 1-d) + \bar{t}(d, 0)$$

Let $c(t)$ denote the capacity of the channel $\pi_d(t)$.

Theorem 8.4.6. *The function $c \circ \pi_d$ for $d > 0$ is strictly monotonically decreasing on $[0, \frac{1}{2}]$ and strictly monotonically increasing on $[\frac{1}{2}, 1]$. For $d = 0$ it is constant and equal to 0.*

Proof. First we prove that $c \circ \pi_d$ is symmetric about $1/2$. The line π_d is given by $\pi_d(t) = (a(t), b(t))$, where $a(t) = t(1-d) + d$ and $b(t) = t(1-d)$. Notice that $a(\bar{t}) = \overline{b(t)}$ and $b(\bar{t}) = \overline{a(t)}$. Using these equations and Lemma 8.4.1(ii), we have

$$c(\bar{t}) = c(a(\bar{t}), b(\bar{t})) = c(\overline{b(t)}, \overline{a(t)}) = c(a(t), b(t)) = c(t)$$

However, because $c \circ \pi_d$ is convex, as the composition of a convex function and a line, this implies that its absolute minimum value is assumed at $t = 1/2$: for any $t \in [0, 1]$,

$$\begin{aligned} c(t) &= \tfrac{1}{2}c(t) + \tfrac{1}{2}c(t) \\ &= \tfrac{1}{2}c(t) + \tfrac{1}{2}c(\bar{t}) && \text{symmetry of } c(t) \\ &\geq c(\tfrac{1}{2}t + \tfrac{1}{2}\bar{t}) && \text{convexity of } c(t) \\ &= c(\tfrac{1}{2}) \end{aligned}$$

For $d > 0$ the capacity is strictly convex on π_d. By Theorem 8.1.1, then, capacity is strictly decreasing along the line $\pi_d : [0, 1/2] \to \mathbb{N}$. Again by Theorem 8.1.1, capacity is strictly decreasing along the line $\pi : [0, 1] \to \mathbb{N}$ given by $\pi(t) = \pi_d(1 - t/2)$, which means it is strictly increasing along the line $\pi_d : [1/2, 1] \to \mathbb{N}$. The line π_0 is the line of zero channels where the capacity is always 0. □

We have derived the following lower and upper bounds on the capacity:

Corollary 8.4.7. *For any binary channel $x \in [0, 1]^2$,*

$$1 - H\left(\frac{1 - |\det(x)|}{2}\right) \leq c(x) \leq \log_2\left(1 + 2^{\frac{-H(|\det(x)|)}{|\det(x)|}}\right)$$

with the understanding that the expression on the right is zero when $\det(x) = 0$.

Proof. By the symmetry of capacity, we know that $c(x)$ is bounded from below by the capacity of the binary symmetric channel $((1 + |\det(x)|)/2, (1 - |\det(x)|)/2)$, which is the expression on the left, and bounded from above by the capacity of the Z channel $(|\det(x)|, 0)$, which is the expression on the right. □

The bounds in Corollary 8.4.7 are canonical:

Definition 8.4.8. A function $f : [0,1]^2 \to \mathbb{R}$ is called *det-invariant* if

$$|\det(x)| = |\det(y)| \Rightarrow f(x) = f(y)$$

for all $x, y \in \mathbb{N}$.

Thus, a det-invariant function is one whose value depends only on the magnitude of the channel's determinant – in particular, such functions are symmetric.

Corollary 8.4.9.

- *The supremum of all det-invariant lower bounds on capacity is*

$$a(x) = 1 - H\left(\frac{1 - |\det(x)|}{2}\right)$$

- *The infimum of all det-invariant upper bounds on capacity is*

$$b(x) = \log_2\left(1 + 2^{\frac{-H(|\det(x)|)}{|\det(x)|}}\right)$$

Proof. Each $x \in \mathbb{N}$ lies on a line π of constant determinant which joins $p = (\det(x), 0)$ to $q = ((1 + \det(x))/2, (1 - \det(x))/2)$. If f is a det-invariant lower bound on capacity,

$$f(x) = f(q) \leq c(q) = 1 - H\left(\frac{1 - |\det(x)|}{2}\right)$$

while for any det-invariant upper bound g we have

$$\log_2\left(1 + 2^{\frac{-H(|\det(x)|)}{|\det(x)|}}\right) = c(p) \leq g(p) = g(x).$$

The argument above applies if $x \notin \mathbb{N}$ since all functions involved are symmetric. Finally, a and b are themselves det-invariant, so the proof is finished. □

The best det-invariant lower bound in Corollary 8.4.7 is the key idea in determining how close in value that $|\det|$ is to c:

Theorem 8.4.10.

$$\sup_{(a,b)\in[0,1]^2} |\det(a,b)| - c(a,b) = \log_2(5/4)$$

This supremum is attained by the channels $(4/5, 1/5)$ *and* $(1/5, 4/5)$.

Proof. The expression we are maximizing is symmetric, so for the purposes of calculation, we can take this supremum over the set of nonnegative channels $\mathbb{N}$. Let $(a, b) \in \mathbb{N}$. Then $\det(a, b) = a - b \geq 0$. Let $y \in \mathbb{N}$ be a binary symmetric channel with $\det(y) = \det(a, b)$. Then

$$|\det(a, b)| - c(a, b) = \det(a, b) - c(a, b) = \det(y) - c(a, b) \leq \det(y) - c(y)$$

Then we can calculate our supremum by considering only nonnegative binary symmetric channels, which can be parametrized by $\{(1 - p, p) : p \in [0, 1/2]\}$. Thus, we need only maximize the function

$$f(p) = \det(1 - p, p) - c(1 - p, p) = (1 - 2p) - (1 - H(p))$$

over the interval $[0, 1/2]$, where H is the base two entropy. The derivative of f on $(0, 1/2)$ is

$$f'(p) = -2 + \log_2(\bar{p}/p)$$

Then $f'(p) > 0$ iff $p \in (0, 1/5)$, $f'(p) < 0$ iff $p \in (1/5, 1/2)$, and $f'(1/5) = 0$. Thus, f has a maximum value at $p = 1/5$, given by

$$f(1/5) = H(1/5) - 2/5 = \log_2(5/4),$$

which finishes the proof. □

The number $\log_2(5/4)$ is approximately equal to 0.3219. Because $|\det|$ itself is a det-invariant upper bound on capacity, $b(x) \leq |\det(x)|$ by Corollary 8.4.9, and we have the following chain of inequalities:

$$a(x) \leq c(x) \leq b(x) \leq |\det(x)| \leq c(x) + \log_2\left(\frac{5}{4}\right)$$

Nine

Hypothesis testing and the probability of error

As we saw in Chapter 7, probabilistic anonymity systems can be fruitfully regarded as information-theoretic channels, where the inputs are the anonymous events, the outputs are the observables and the channel matrix represents the correlation between the anonymous and observed events, in terms of conditional probabilities. An adversary can try to infer the anonymous event that took place from his observations using the Bayesian method, which is based on the principle of assuming an *a priori probability distribution* on the anonymous events (*hypotheses*), and deriving from that (and from the matrix) the *a posteriori probability distribution* after a certain event has been observed. It is well known that the best strategy for the adversary is to apply the MAP (Maximum Aposteriori Probability) criterion, which, as the name says, dictates that one should choose the hypothesis with the maximum a posteriori probability given the observation. "Best" means that this strategy induces the smallest probability of guessing the wrong hypothesis. The probability of error, in this case, is also called *Bayes risk*.

Even if the adversary does not know the *a priori* distribution, the method is still valid asymptotically, under the condition that the matrix's rows are all pairwise distinguished. By repeating the experiment, the contribution of the *a priori* probability becomes less and less relevant for the computation of the a posteriori probability, and it "washes out" in the limit [CT91]. Furthermore, the probability of error converges to 0 in the limit. If the rows are all equal, namely if the channel has capacity 0, then the Bayes risk is maximal and does not converge to 0. This is the ideal situation, from the point of view of information-hiding protocols. In practice, however, it is difficult to achieve such degree of anonymity.

In general we are interested in maximizing the Bayes risk. The main purpose of this chapter is to investigate the Bayes risk, in relation to the channel's matrix, and to produce bounds on it.

There are many bounds known in literature for the Bayes risk. An interesting class of such bounds is based on relations with the conditional entropy of the channel's input given the output (*equivocation*). The first result of this kind, found by Rényi [Rén66], established that the probability of error is bounded

by the equivocation. Later, Hellman and Raviv improved this bound by half [HR07]. Recently, Santhi and Vardy have proposed a new bound, that depends exponentially on the (opposite of the) equivocation, and which considerably improves the Hellman-Raviv bound in the case of multi-hypothesis testing [SV06]. The Hellman-Raviv bound, however, is better than the Santhi-Vardy bound in the case of two hypotheses.

Contribution The contribution of this chapter consists of the following:

- We consider what we call "the corner points" of a piecewise linear function, and we propose criteria to compute the maximum of the function, and to identify concave functions that are upper bounds for the given piecewise linear function, based on the analysis of its corner points only.
- We develop a technique that allows us to prove that a certain set of points is a set of corner points of a given function. By using the notion of corner points, we are able to give alternative proofs of the Hellman-Raviv and the Santhi-Vardy bounds, much simpler than the original proofs.
- We show that the probability of error associated to the MAP rule is piecewise linear, and we give a constructive characterization of a set of corner points, which turns out to be finite. This characterization is the central and most substantial result of this chapter.
- Using the above results, we establish methods (a) to compute the maximum probability of error over all the input distributions, and (b) to improve on the Hellman-Raviv and the Santhi-Vardy bounds. In particular, our improved bounds always are tight at least at one point, while the others are tight at some points only in case of channels of capacity 0.
- We show how to apply the above results to randomized protocols for anonymity. In particular, we work out in detail the application to Crowds, and derive the maximum probability of error for an adversary who tries to break anonymity, and bounds on this probability in terms of conditional entropy, for any input distribution.
- We explore the consequences of protocol repetition for hypothesis testing. If the rows of the matrix are pairwise different, then the MAP rule can be approximated by a rule called *Maximum Likelihood*, which does not require the knowledge of the *a priori* distribution. Furthermore, the probability of error converges to 0 as the number of repetitions increases. We also show the converse, namely if two or more rows are identical, then the probability of error of any decision rule has a positive lower bound. The first result is an elaborations of a remark we found in [CT91] and the second is an easy consequence of the theorem of the central limit. The latter is, to the best of our knowledge, our contribution, in the sense that we were not able to find it in literature.

Plan of the chapter Next section recalls some basic notions in information theory, and about hypothesis testing and the probability of error. Section 9.2 proposes some methods to identify bounds for a function that is generated by a

set of corner points; these bounds are tight on at least one corner point. Using the notion of corner points we show an alternative proof of the Hellman-Raviv and the Santhi-Vardy bounds. Section 9.3 presents the main result of this chapter, namely a constructive characterization of the corner points of Bayes risk. Section 9.4 illustrates an application of our results to Crowds. Section sec:hyp:repetition considers the case of protocol repetition. Finally Section 9.6 discusses related work.

9.1 Hypothesis testing and the probability of error

In this section we briefly review some basic notions on hypothesis testing.

We consider discrete channels $(\mathcal{A}, \mathcal{O}, p_c)$ where the sets of input values $\mathcal{A}$ and output values $\mathcal{O}$ are finite with cardinality n and m respectively. We will also sometimes use indices to represent their elements: $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ and $\mathcal{O} = \{o_1, o_2, \ldots, o_m\}$. The *matrix* p_c of the channel gives the conditional probability of observing an output given a certain input, the usual convention is to arrange the a's by rows and the o's by columns.

The set of input values can also be regarded as a set of *mutually exclusive* (hidden) *facts* or *hypotheses*. A probability distribution $p_{\mathcal{A}}$ over $\mathcal{A}$ is called *a priori probability*, and together with the channel it induces a joint probability distribution p over $\mathcal{A} \times \mathcal{O}$ as $p(a, o) = p_{\mathcal{A}}(a)\, p_c(o|a)$ such that $p([o]|[a]) = p_c(o|a)$, where $[a] = \{a\} \times \mathcal{O}, [o] = \mathcal{A} \times \{o\}$. As usual, we often write $p(a), p(o), p(o|a)$ instead of $p([a]), p([o]), p([o]|[a])$ for simplicity. The probability

$$p([o]) = \sum_a p(a, o) = \sum_a p_{\mathcal{A}}(a)\, p_c(o|a)$$

is called the *marginal probability* of $o \in \mathcal{O}$.

When we observe an output o, the probability that the corresponding input has been a certain a is given by the conditional probability $p(a|o)$, also called *a posteriori probability* of a given o, which in general is different from $p(a)$. This difference can be interpreted as the fact that observing o gives us evidence that changes our degree of belief in the hypothesis a. The *a priori* and the a posteriori probabilities of a are related by Bayes' theorem:

$$p(a|o) = \frac{p(o|a)\, p(a)}{p(o)}$$

In hypothesis testing we try to infer the *true* hypothesis (i.e. the input fact that really took place) from the observed output. In general, it is not possible to determine the right hypothesis with certainty. We are interested, then, in minimizing the *probability of error*, i.e. the probability of making the wrong guess. Formally, the probability of error is defined as follows. Given the *decision function* $f : \mathcal{O} \to \mathcal{A}$ adopted by the observer to infer the hypothesis, let $E_f : \mathcal{A} \to 2^{\mathcal{O}}$ be the function that gives the *error region* of f when $a \in \mathcal{A}$ has occurred, namely:

$$E_f(a) = \{o \in \mathcal{O} \mid f(o) \neq a\}$$

Let $\eta_f : \mathcal{A} \to [0,1]$ be the function that associates to each $a \in \mathcal{A}$ the probability that f gives the the wrong input fact when $a \in \mathcal{A}$ has occurred, namely:

$$\eta_f(a) = \sum_{o \in E_f(a)} p(o|a)$$

The probability of error for f is then obtained as the sum of the probability of error for each possible input, averaged over the probability of the input:

$$P_f = \sum_a p(a)\, \eta_f(a)$$

In the Bayesian framework, the best possible decision function f_B, namely the decision function that minimizes the probability of error, is obtained by applying the MAP (*Maximum Aposteriori Probability*) criterion, that chooses an input a with a maximal $p(a|o)$. Formally:

$$f_B(o) = a \quad \Rightarrow \quad \forall a' \;\; p(a|o) \geq p(a'|o)$$

A decision function that satisfies the above condition will be called *MAP decision function*. The probability of error associated to f_B, also called the *Bayes risk*, is then given by

$$P_e = 1 - \sum_o p(o) \max_a p(a|o) = 1 - \sum_o \max_a p(o|a)\, p(a)$$

Note that f_B, and the Bayes risk, depend on the inputs' *a priori* probability. The input distributions can be represented as the elements $\vec{x} = (x_1, x_2, \ldots, x_n)$ of a domain $D^{(n)}$ defined as

$$D^{(n)} = \{\vec{x} \mid \sum_i x_i = 1 \text{ and } \forall i \; x_i \geq 0\}$$

where the correspondence is given by $\forall i \; x_i = p(a_i)$. In the rest of the chapter we will assume the MAP rule and view the Bayes risk as a function $P_e : D^{(n)} \to [0,1]$ defined by

$$P_e(\vec{x}) = 1 - \sum_i \max_j p(o_i|a_j) x_j \tag{9.1}$$

There are some notable results in the literature relating the Bayes risk to the information-theoretic notion of *conditional entropy*, also called *equivocation*. A brief discussion about entropy is made in Section 2.2. We recall that the entropy $H(A)$ measures the uncertainty of a random variable A. It takes its maximum value $\log n$ when A's distribution is uniform and its minimum value 0 when A is constant. The conditional entropy $H(A|O)$ measures the amount of uncertainty of A when O is known. It can be shown that $0 \leq H(A|O) \leq H(A)$. It takes its maximum value $H(A)$ when O reveals no information about A, i.e. when A and O are independent, and its minimum value 0 when O completely determines the value of A.

Given a channel, let $\vec{x}$ be the *a priori* distribution on the inputs. Recall that $\vec{x}$ also determines a probability distribution on the outputs. Let A and

O be the random variables associated to the inputs and outputs respectively. The Bayes risk is related to $H(A|O)$ by the Hellman-Raviv bound [HR07]:

$$P_e(\vec{x}) \leq \frac{1}{2}H(A|O) \tag{9.2}$$

and by the Santhi-Vardy bound [SV06]:

$$P_e(\vec{x}) \leq 1 - 2^{-H(A|O)} \tag{9.3}$$

We remark that, while the bound (9.2) is tighter than (9.3) in case of binary hypothesis testing, i.e. when $n = 2$, (9.3) gives a much better bound when n becomes larger. In particular the bound in (9.3) is always limited by 1, which is not the case for (9.2).

9.2 Convexly generated functions and their bounds

In this section we characterize a special class of functions on probability distributions, and we present various results regarding their bounds which lead to methods to compute their maximum, to prove that a concave function is an upper bound, and to derive an upper bound from a concave function. The interest of this study is that the probability of error will turn out to be a function in this class.

We recall than a subset S of a vector space is called *convex* if it is closed under convex combination (see Section 2.3 for a brief discussion about convexity). It is easy to see that for any n the domain $D^{(n)}$ of probability distributions of dimension n (that is a $(n-1)$-simplex) is convex. The *convex hull* of S, denoted by $ch(S)$ is the smallest convex set containing S. An interesting case is when we can generate all elements of a set S from a smaller set U using convex combinations. This brings us to the concept of *convex base*:

Definition 9.2.1. *Given the vector sets S, U, we say that U is a convex base for S if and only if $U \subseteq S$ and $S \subseteq ch(U)$.*

In the following, given a vector $\vec{x} = (x_1, x_2, \ldots, x_n)$, and a function f from n-dimensional vectors to reals, we will use the notation $(\vec{x}, f(\vec{x}))$ to denote the vector (in a space with one additional dimension) $(x_1, x_2, \ldots, x_n, f(\vec{x}))$. Similarly, given a vector set S in a n-dimensional space, we will use the notation $(S, f(S))$ to represent the set of vectors $\{(\vec{x}, f(\vec{x})) \mid \vec{x} \in S\}$ in a $(n+1)$-dimensional space. The notation $f(S)$ represents the image of S under f, i.e. $f(S) = \{f(\vec{x}) \mid \vec{x} \in S\}$.

We are now ready to introduce the class of functions that we mentioned at the beginning of this section:

Definition 9.2.2. *Given a vector set S, a convex base U of S, and a function $f : S \to \mathbb{R}$, we say that $(U, f(U))$ is a set of* corner points *of f if and only if $(U, f(U))$ is a convex base for $(S, f(S))$. We also say that f is* convexly generated *by $f(U)$*[1].

[1] To be more precise we should say that f is convexly generated by $(U, f(U))$.

Of particular interest are the functions that are convexly generated by a finite number of corner points. This is true for *piecewise linear functions* in which S can be decomposed into finitely many convex polytopes (n-dimensional polygons) and f is equal to a linear function on each of them. Such functions are convexly generated by the finite set of vertices of these polytopes.

We now give a criterion for computing the maximum of a convexly generated function.

Proposition 9.2.3. *Let $f : S \to \mathbb{R}$ be convexly generated by $f(U)$. If $f(U)$ has a maximum element b, then b is the maximum value of f on S.*

Proof. Let b be the maximum of $f(U)$. Then for every $u \in U$ we have that $f(u) \leq b$. Consider now a vector $\vec{x} \in S$. Since f is convexly generated by $f(U)$, there exist $\vec{u}^1$, $\vec{u}^2$, ..., $\vec{u}^k$ in U such that $f(\vec{x})$ is obtained by convex combination from $f(\vec{u}^1)$, $f(\vec{u}^2)$, ..., $f(\vec{u}^k)$ via some convex coefficients λ_1, λ_2, ..., λ_k. Hence:

$$\begin{array}{rcll} f(\vec{x}) & = & \sum_i \lambda_i f(\vec{u}^i) & \\ & \leq & \sum_i \lambda_i b & \text{since } f(\vec{u}^i) \leq b \\ & = & b & \lambda_i\text{'s being convex combinators} \end{array}$$

□

Note that if U is finite then $f(U)$ always has a maximum element.

Next, we propose a method for establishing functional upper bounds for f, when they are in the form of *concave* functions (see Section 2.3 for a definition of concave functions).

Proposition 9.2.4. *Let $f : S \to \mathbb{R}$ be convexly generated by $f(U)$ and let $g : S \to \mathbb{R}$ be concave. Assume that for all $\vec{u} \in U$ $f(\vec{u}) \leq g(\vec{u})$ holds. Then we have that g is an upper bound for f, i.e.*

$$\forall \vec{x} \in S \;\; f(\vec{x}) \leq g(\vec{x})$$

Proof. Let $\vec{x}$ be an element of S. Since f is convexly generated, there exist $\vec{u}^1$, $\vec{u}^2$, ..., $\vec{u}^k$ in U such that $(\vec{x}, f(\vec{x}))$ is obtained by convex combination from $(\vec{u}^1, f(\vec{u}^1))$, $(\vec{u}^2, f(\vec{u}^2))$, ..., $(\vec{u}^k, f(\vec{u}^k))$ via some convex coefficients λ_1, λ_2, ..., λ_k. Hence:

$$\begin{array}{rcll} f(\vec{x}) & = & \sum_i \lambda_i f(\vec{u}^i) & \\ & \leq & \sum_i \lambda_i g(\vec{u}^i) & \text{since } f(\vec{u}^i) \leq g(\vec{u}^i) \\ & \leq & g(\sum_i \lambda_i \vec{u}^i) & \text{by the concavity of } g \\ & = & g(\vec{x}) & \end{array}$$

□

We also give a method to obtain functional upper bounds, that are tight on at least one corner point, from concave functions.

Proposition 9.2.5. *Let $f : S \to \mathbb{R}$ be convexly generated by $f(U)$ and let $g : S \to \mathbb{R}$ be concave and non-negative. Let $R = \{c \mid \exists \vec{u} \in U : f(\vec{u}) \geq c\, g(\vec{u})\}$ and assume that R has an upper bound. Then the function $c_o\, g$ is a functional upper bound for f satisfying*

$$\forall \vec{x} \in S \;\; f(\vec{x}) \leq c_o\, g(\vec{x})$$

where $c_o = \sup R$. If $c_o \in R$ then f and $c_o\, g$ coincide at least at one point.

Proof. We first show that $f(\vec{u}) \leq c_o\, g(\vec{u})$ for all $\vec{u} \in U$. Suppose the opposite, then there exists $\vec{u} \in U$ such that $f(\vec{u}) > c_o\, g(\vec{u})$. If $g(\vec{u}) = 0$ then for all $c \in \mathbb{R} : f(\vec{u}) > c\, g(\vec{u}) = 0$ so the set R is not bounded, which is a contradiction. If $g(\vec{u}) > 0$ (we assumed that g is non-negative) then let $c = \frac{f(\vec{u})}{g(\vec{u})}$ so $c > c_o$ but also $c \in R$ which is also a contradiction since $c = \sup R$.

Hence by Proposition 9.2.4 we have that $c_o\, g$ is an upper bound for f. Furthermore, if $c_o \in R$ then there exists $\vec{u} \in U$ such that $f(\vec{u}) \geq c_o\, g(\vec{u})$ so $f(\vec{u}) = c_o\, g(\vec{u})$ and the bound is tight as this point. □

Note that, if U is finite and $\forall \vec{u} \in U : g(\vec{u}) = 0 \Rightarrow f(\vec{u}) \leq 0$, then the maximum element of R always exists and is equal to

$$\max_{\vec{u} \in U, g(\vec{u}) > 0} \frac{f(\vec{u})}{g(\vec{u})}$$

Finally, we develop a proof technique that will allow us to prove that a certain set is a set of corner points of a function f. Let S be a set of vectors. The *extreme points* of S, denoted by $extr(S)$, is the set of points of S that cannot be expressed as the convex combination of two distinct elements of S. An subset of $\mathbb{R}^n$ is called *compact* if it is closed and bounded. Our proof technique uses the Krein-Milman theorem which relates a compact convex set to its extreme points.

Theorem 9.2.6 (Krein-Milman)**.** *A compact and convex vector set is equal to the convex hull of its extreme points.*

We refer to [Roy88] for the proof. Now since the extreme points of S are enough to generate S, to show that a given set $(U, f(U))$ is a set of corner points, it is sufficient to show that all extreme points are included in it.

Proposition 9.2.7. *Let S be a compact vector set, U be a convex base of S and $f : S \to \mathbb{R}$ be a continuous function. Let $T = S \setminus U$. If all elements of $(T, f(T))$ can be written as the convex combination of two distinct elements of $(S, f(S))$ then $(U, f(U))$ is a set of corner points of f.*

Proof. Let $S_f = (S, f(S))$ and $U_f = (U, f(U))$. Since S is compact and continuous maps preserve compactness then S_f is also compact, and since the convex hull of a compact set is compact then $ch(S_f)$ is also compact (note that we didn't require S to be convex). Then $ch(S_f)$ satisfies the requirements of the Krein-Milman theorem, and since the extreme points of $ch(S_f)$ are clearly the same as those of S_f we have

$$\begin{aligned} ch(extr(ch(S_f))) = ch(S_f) \Rightarrow \\ ch(extr(S_f)) = ch(S_f) \end{aligned} \tag{9.4}$$

Now all points in $S_f \setminus U_f$ can be written as convex combinations of other (distinct) points, so they are not extreme. Thus all extreme points are contained in U_f, that is $extr(S_f) \subseteq U_f$, and since $ch(\cdot)$ is monotone with respect to set inclusion, we have

$$\begin{aligned} &ch(extr(S_f)) \subseteq ch(U_f) \Rightarrow \\ &S_f \subseteq ch(S_f) \subseteq ch(U_f) \qquad \text{by (9.4)} \end{aligned}$$

which means that U_f is a set of corner points of f. □

The big advantage of the above proposition is that we need to express points outside U as convex combinations of any other points, not necessarily of points in U (as a direct application of the definition of corner points would require).

9.2.1 An alternative proof for the Hellman-Raviv and Santhi-Vardy bounds

Using Proposition 9.2.4 we can give an alternative, simpler proof for the bounds in (9.2) and (9.3). Let $f : D^{(n)} \to \mathbb{R}$ be the function $f(\vec{y}) = 1 - \max_j y_j$. We start by identifying a set of corner points of f, using Prop. 9.2.7 to prove that they are indeed corner points.

Proposition 9.2.8. *The function f defined above is convexly generated by $f(U)$ with $U = U_1 \cup U_2 \cup \ldots \cup U_n$ where, for each k, U_k is the set of all vectors that have value $1/k$ in exactly k components, and 0 everywhere else.*

Proof. We have to show that for any point $\vec{x}$ in $S \setminus U$, $(\vec{x}, f(\vec{x}))$ can be written as a convex combination of two points in $(S, f(S))$. Let $w = \max_i x_i$. Since $\vec{x} \notin U$ then there is at least one element of $\vec{x}$ that is neither w nor 0, let x_i be that element. Let k the number of elements equal to w. We create two vectors $\vec{y}, \vec{z} \in S$ as follows

$$y_j = \begin{cases} x_i + \epsilon & \text{if } i = j \\ w - \frac{\epsilon}{k} & \text{if } x_j = w \\ x_j & \text{otherwise} \end{cases} \qquad z_j = \begin{cases} x_i - \epsilon & \text{if } i = j \\ w + \frac{\epsilon}{k} & \text{if } x_j = w \\ x_j & \text{otherwise} \end{cases}$$

where ϵ is a very small positive number, such that $w - \frac{\epsilon}{k}$ is still the maximum element. Clearly $\vec{x} = \frac{1}{2}\vec{y} + \frac{1}{2}\vec{z}$ and since $f(\vec{x}) = 1 - w$, $f(\vec{y}) = 1 - w + \frac{\epsilon}{k}$ and $f(\vec{y}) = 1 + w - \frac{\epsilon}{k}$ we have $f(\vec{x}) = \frac{1}{2}f(\vec{y}) + \frac{1}{2}f(\vec{z})$. Since f is continuous and $D^{(n)}$ is compact, the result follows from Prop. 9.2.7. □

Consider now the functions $g, h : D^{(n)} \to \mathbb{R}$ defined as

$$g(\vec{y}) = \frac{1}{2}H(\vec{y}) \quad \text{and} \quad h(\vec{y}) = 1 - 2^{-H(\vec{y})}$$

where (with a slight abuse of notation) H represents the entropy of the distribution $\vec{y}$, i.e. $H(\vec{y}) = -\sum_j y_j \log y_j$.

We now compare g, h with $f(\vec{y}) = 1 - \max_j y_j$ on the corner points on f. A corner point $\vec{u}^k \in U_k$ (defined in Prop. 9.2.8) has k elements equal to $1/k$ and

the rest equal to 0. So $H(\vec{u}^k) = \log k$ and

$$f(\vec{u}^k) = 1 - \frac{1}{k}$$
$$g(\vec{u}^k) = \frac{1}{2}\log k$$
$$h(\vec{u}) = 1 - 2^{-\log k} = 1 - \frac{1}{k}$$

So $f(\vec{u}^1) = 0 = g(\vec{u}^1)$, $f(\vec{u}^2) = 1/2 = g(\vec{u}^2)$, and for $k > 2$, $f(\vec{u}^k) < g(\vec{u}^k)$. On the other hand, $f(\vec{u}^k) = h(\vec{u}^k)$, for all k.

Thus, both g and h are greater or equal than f on all the corner points so from Proposition 9.2.4 we have

$$\forall \vec{y} \in D^{(n)} \quad f(\vec{y}) \leq g(\vec{y}) \quad \text{and} \quad f(\vec{y}) \leq h(\vec{y}) \tag{9.5}$$

The rest of the proof proceeds as in [HR07] and [SV06]: Let $\vec{x}$ represent an *a priori* distribution on $\mathcal{A}$ and let the above $\vec{y}$ denote the a posteriori probabilities on $\mathcal{A}$ with respect to a certain observable o, i.e. $y_j = p(a_j|o) = (p(o|a_j)/p(o))\, x_j$. Then $P_e(\vec{x}) = \sum_o p(o) f(\vec{y})$, so from (9.5) we obtain

$$P_e(\vec{x}) \leq \sum_o p(o) \frac{1}{2} H(\vec{y}) = \frac{1}{2} H(A|O) \tag{9.6}$$

and

$$P_e(\vec{x}) \leq \sum_o p(o)(1 - 2^{-H(\vec{y})}) \leq 1 - 2^{-H(A|O)} \tag{9.7}$$

where the last step in (9.7) is obtained by applying Jensen's inequality. This concludes the alternative proof of (9.2) and (9.3).

We end this section with two remarks. First, we note that g coincides with f only on the points of U_1 and U_2, whereas h coincides with f on all U. This explains, intuitively, why (9.3) is a better bound than (9.2) for dimensions higher than 2.

Second, we observe that, although h is a good bound for f, when we average h and f on the output probabilities to obtain $\sum_o p(o)(1 - 2^{-H(\vec{y})})$ and $P_e(\vec{x})$ respectively, and then we apply Jensen's inequality, we usually loosen this bound significantly, as we will see in some examples later. The only case in which we do not loosen it is when the channel has capacity 0 (maximally noisy channel), i.e. all the rows of the matrix are the same. In the general case of non-zero capacity, however, this implies that if we want to obtain a better bound we need to follow a different strategy. In particular, we need to find directly the corner points of P_e instead than those of the f defined above. This is what we are going to do in the next section.

9.3 The corner points of the Bayes risk

In this section we present our main contribution, namely we show that P_e is convexly generated by $P_e(U)$ for a finite U, and we give a constructive characterization of U, so that we can apply the results of previous section to compute tight bounds on P_e.

The idea behind the construction of such U is the following: recall that the Bayes risk is given by $P_e(\vec{x}) = 1 - \sum_i \max_j p(o_i|a_j)x_j$. Intuitively, this function is linear as long as, for each i, the j which gives the maximum $p(o_i|a_j)x_j$ remains the same while we vary $\vec{x}$. When, for some i and k, the maximum becomes $p(o_i|a_k)x_k$, the function changes its inclination and then it becomes linear again. The exact point in which the inclination changes is a solution of the equation $p(o_i|a_j)x_j = p(o_i|a_k)x_k$. This equation actually represents a hyperplane (a space in $n-1$ dimensions, where n is the cardinality of $\mathcal{A}$) and the inclination of P_e changes in all its points for which $p(o_i|a_j)x_j$ is maximum, i.e. it satisfies the inequality $p(o_i|a_j)x_j \geq p(o_i|a_\ell)x_\ell$ for each ℓ. The intersection of $n-1$ hyperplanes of this kind, and of the one determined by the equation $\sum_j x_j = 1$, is a vertex $\vec{v}$ such that $(\vec{v}, P_e(\vec{v}))$ is a corner point of P_e.

Definition 9.3.1. *Given a channel $\mathcal{C} = (\mathcal{A}, \mathcal{O}, p_c)$, the family $\mathbb{S}(\mathcal{C})$ of systems generated by $\mathcal{C}$ is the set of all systems of inequalities of the following form:*

$$\begin{array}{rcll}
p_c(o_{i_1}|a_{j_1})x_{j_1} & = & p_c(o_{i_1}|a_{j_2})x_{j_2} & \\
p_c(o_{i_2}|a_{j_3})x_{j_3} & = & p_c(o_{i_2}|a_{j_4})x_{j_4} & \\
 & \vdots & & \\
p_c(o_{i_r}|a_{j_{2r-1}})x_{j_{2r-1}} & = & p_c(o_{i_r}|a_{j_{2r}})x_{j_{2r}} & \\
x_j & = & 0 \quad \textit{for } j \notin \{j_1, j_2, \ldots, j_{2r}\} & \\
x_1 + x_2 + \ldots + x_n & = & 1 & \\
p_c(o_{i_h}|a_{j_{2h}})x_{j_{2h}} & \geq & p_c(o_{i_h}|a_\ell)x_\ell \qquad \textit{for } 1 \leq h \leq r & \\
 & & \qquad\qquad\qquad\quad \textit{and } 1 \leq \ell \leq n &
\end{array}$$

such that all the coefficients $p(o_{i_h}|a_{j_{2h-1}})$, $p(o_{i_h}|a_{j_{2h}})$ are strictly positive ($1 \leq h \leq r$), and the equational part has exactly one solution. Here n is the cardinality of $\mathcal{A}$, and r ranges between 0 and $n-1$.

The variables of the above systems of inequalities are $x_1, \ldots, x_n$. Note that for $r = 0$ the system consists only of $n-1$ equations of the form $x_j = 0$, plus the equation $x_1 + x_2 + \ldots + x_n = 1$. A system is called *solvable* if it has solutions. By definition, a system of the kind considered in the above definition has at most one solution.

The condition on the uniqueness of solution requires to (attempt to) solve more systems than they are actually solvable. Since the number of systems of equations of the form given in Definition 9.3.1 increases very fast with n, it is reasonable to raise the question of the effectiveness of our method. Fortunately, we will see that the uniqueness of solution can be characterized by a simpler condition (cf. Proposition 9.3.7), however still producing a huge number of systems. We will investigate the complexity of our method in Section 9.3.1.

We are now ready to state our main result:

Theorem 9.3.2. *Given a channel $\mathcal{C}$, the Bayes risk P_e associated to $\mathcal{C}$ is convexly generated by $P_e(U)$, where U is constituted by the solutions to all solvable systems in $\mathbb{S}(\mathcal{C})$.*

Proof. We need to prove that, for every $\vec{u} \in D^{(n)}$, there exist $\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_t \in U$, and convex coefficients $\lambda_1, \lambda_2, \ldots, \lambda_t$ such that

$$\vec{u} = \sum_i \lambda_i \vec{u}_i \quad \text{and} \quad P_e(\vec{u}) = \sum_i \lambda_i P_e(\vec{u}_i)$$

Let us consider a particular $\vec{u} \in D^{(n)}$. In the following, for each i, we will use j_i to denote the index j for which $p_c(o_i|a_j)u_j$ is maximum. Hence, we can rewrite $P_e(\vec{u})$ as

$$P_e(\vec{u}) = 1 - \sum_i p_c(o_i|a_{j_i})u_{j_i} \tag{9.8}$$

We proceed by induction on n. All conditional probabilities $p_c(o_i|a_j)$ that appear in the proof are assumed to be strictly positive: we do not need to consider the ones which are zero, because we are interested in maximizing the terms of the form $p_c(o_i|a_j)x_j$.

Base case ($n = 2$) In this case U is the set of solutions of all the systems of the form

$$\{p_c(o_i|a_1)x_1 = p_c(o_i|a_2)x_2 \ , \ x_1 + x_2 = 1\}$$

or

$$\{x_j = 0 \ , \ x_1 + x_2 = 1\}$$

and $\vec{u} \in D^{(2)}$. Let c be the minimum $x \geq 0$ such that

$$p_c(o_i|a_1)(u_1 - x) = p_c(o_i|a_2)(u_2 + x) \quad \text{for some } i$$

or let c be u_1 if such x does not exist. Analogously, let d be the minimum $x \geq 0$ such that

$$p_c(o_i|a_2)(u_2 - x) = p_c(o_i|a_1)(u_1 + x) \quad \text{for some } i$$

or let d be u_2 if such x does not exist.

Note that $p_c(o_i|a_2)(u_2+c) \geq 0$, hence $u_1-c \geq 0$ and consequently $u_2+c \leq 1$. Analogously, $u_2 - d \geq 0$ and $u_1 + d \leq 1$. Let us define $\vec{v}$, $\vec{w}$ (the corner points of interest) as

$$\vec{v} = (u_1 - c, u_2 + c) \qquad \vec{w} = (u_1 + d, u_2 - d)$$

Consider the convex coefficients

$$\lambda = \frac{d}{c+d} \qquad \mu = \frac{c}{c+d}$$

A simple calculation shows that

$$\vec{u} = \lambda\vec{v} + \mu\vec{w}$$

It remains to prove that

$$P_e(\vec{u}) = \lambda P_e(\vec{v}) + \mu P_e(\vec{w}) \tag{9.9}$$

To this end, it is sufficient to show that P_e is defined in $\vec{v}$ and $\vec{w}$ by the same formula as (9.8), i.e. that $P_e(\vec{v})$, $P_e(\vec{w})$ and $P_e(\vec{u})$ are obtained as values, in $\vec{v}$, $\vec{w}$ and $\vec{u}$, respectively, of the same linear function. This amounts to show that the coefficients are the same, i.e. that for each i and k the inequality $p_c(o_i|a_{j_i})v_{j_i} \geq p_c(o_i|a_k)v_k$ holds, and similarly for $\vec{w}$.

Let i and k be given. If $j_i = 1$, and consequently $k = 2$, we have that $p_c(o_i|a_1)u_1 \geq p_c(o_i|a_2)u_2$ holds. Hence for some $x \geq 0$ the equality $p_c(o_i|a_1)(u_1 - x) = p_c(o_i|a_2)(u_2 + x)$ holds. Therefore:

$$\begin{aligned}
p_c(o_i|a_1)v_1 &= p_c(o_i|a_1)(u_1 - c) && \text{by definition of } \vec{v} \\
&\geq p_c(o_i|a_1)(u_1 - x) && \text{since } c \leq x \\
&= p_c(o_i|a_2)(u_2 + x) && \text{by definition of } x \\
&\geq p_c(o_i|a_2)(u_2 + c) && \text{since } c \leq x \\
&= p_c(o_i|a_1)v_2 && \text{by definition of } \vec{v}
\end{aligned}$$

If, on the other hand, $j_i = 2$, and consequently $k = 1$, we have:

$$\begin{aligned}
p_c(o_i|a_2)v_2 &= p_c(o_i|a_2)(u_2 + c) && \text{by definition of } \vec{v} \\
&\geq p_c(o_i|a_2)u_2 && \text{since } c \geq 0 \\
&\geq p_c(o_i|a_1)u_1 && \text{since } j_i = 2 \\
&\geq p_c(o_i|a_1)(u_1 - c) && \text{since } c \geq 0 \\
&= p_c(o_i|a_1)v_1 && \text{by definition of } \vec{v}
\end{aligned}$$

The proof that for each i and k the inequality $p_c(o_i|a_{j_i})w_{j_i} \geq p_c(o_i|a_k)w_k$ holds is analogous.

Hence we have proved that

$$P_e(\vec{v}) = 1 - \sum_i p_c(o_i|a_{j_i})v_{j_i} \quad \text{and} \quad P_e(\vec{w}) = 1 - \sum_i p_c(o_i|a_{j_i})w_{j_i}$$

and a simple calculation shows that (9.9) holds.

Inductive case Let $\vec{u} \in D^{(n)}$. Let c be the minimum $x \geq 0$ such that for some i and k

$$\begin{aligned}
p_c(o_i|a_{j_i})(u_{j_i} - x) &= p_c(o_i|a_n)(u_n + x) \quad j_i = n - 1 \\
&\text{or} \\
p_c(o_i|a_{j_i})(u_{j_i} - x) &= p_c(o_i|a_k)u_k \quad j_i = n - 1 \text{ and } k \neq n \\
&\text{or} \\
p_c(o_i|a_{j_i})u_{j_i} &= p_c(o_i|a_n)(u_n + x) \quad j_i \neq n - 1
\end{aligned}$$

or let c be u_{n-1} if such x does not exist. Analogously, let d be the minimum $x \geq 0$ such that for some i and k

$$\begin{aligned}
p_c(o_i|a_{j_i})(u_{j_i} - x) &= p_c(o_i|a_{n-1})(u_{n-1} + x) \quad j_i = n \\
&\text{or} \\
p_c(o_i|a_{j_i})(u_{j_i} - x) &= p_c(o_i|a_k)u_k \quad j_i = n \text{ and } k \neq n - 1 \\
&\text{or} \\
p_c(o_i|a_{j_i})u_{j_i} &= p_c(o_i|a_{n-1})(u_{n-1} + x) \quad j_i \neq n
\end{aligned}$$

or let d be u_n if such x does not exist. Similarly to the base case, define $\vec{v}$, $\vec{w}$ as

$$\vec{v} = (u_1, u_2, \ldots, u_{n-2}, u_{n-1} - c, u_n + c)$$

and

$$\vec{w} = (u_1, u_2, \ldots, u_{n-2}, u_{n-1} + d, u_n - d)$$

and consider the same convex coefficients

$$\lambda = \frac{d}{c+d} \qquad \mu = \frac{c}{c+d}$$

Again, we have $\vec{u} = \lambda\vec{v} + \mu\vec{w}$.

By case analysis, and following the analogous proof given for $n = 2$, we can prove that for each i and k the inequalities $p_c(o_i|a_{j_i})v_{j_i} \geq p_c(o_i|a_k)v_k$ and $p_c(o_i|a_{j_i})w_{j_i} \geq p_c(o_i|a_k)w_k$ hold, hence, following the same lines as in the base case, we derive

$$P_e(\vec{u}) = \lambda P_e(\vec{v}) + \mu P_e(\vec{w})$$

We now prove that $\vec{v}$ and $\vec{w}$ can be obtained as convex combinations of corner points of P_e in the hyperplanes (instances of $D^{(n-1)}$) defined by the equations that give, respectively, the c and d above. More precisely, if $c = u_{n-1}$ the equation is $x_{n-1} = 0$. Otherwise, the equation is of the form

$$p_c(o_i|a_k)x_k = p_c(o_i|a_\ell)x_\ell$$

and analogously for d. We develop the proof for $\vec{w}$; the case of $\vec{v}$ is analogous.

If $d = u_n$, then the hyperplane is defined by the equation $x_n = 0$, and it consists of the set of vectors of the form $(x_1, x_2, \ldots, x_{n-1})$. The Bayes risk is defined in this hyperplane exactly in the same way as P_e (since the contribution of x_n is null) and therefore the corner points are the same. By inductive hypothesis, those corner points are given by the solutions to the set of inequalities of the form given in Definition 9.3.1. To obtain the corner points in $D^{(n)}$ it is sufficient to add the equation $x_n = 0$.

Assume now that d is given by one of the other equations. Let us consider the first one, the cases of the other two are analogous. Let us consider, therefore, the hyperplane $\mathcal{H}$ (instance of $D^{(n-1)}$) defined by the equation

$$p_c(o_i|a_n)x_n = p_c(o_i|a_{n-1})x_{n-1} \tag{9.10}$$

It is convenient to perform a transformation of coordinates. Namely, represent the elements of $\mathcal{H}$ as vectors $\vec{y}$ with

$$y_j = \begin{cases} x_j & 1 \leq j \leq n-2 \\ x_{n-1} + x_n & j = n-1 \end{cases} \tag{9.11}$$

Consider the channel

$$\mathcal{C}' = \langle \mathcal{A}', \mathcal{O}, p'(\cdot|\cdot) \rangle$$

with $\mathcal{A}' = \{a_1, a_2, \ldots, a_{n-1}\}$, and

$$p'(o_k|a_j) = \begin{cases} p_c(o_k|a_j) & 1 \leq j \leq n-2 \\ \max\{p_1(k), p_2(k)\} & j = n-1 \end{cases}$$

where

$$p_1(k) = p_c(o_k|a_{n-1})\frac{p_c(o_i|a_n)}{p_c(o_i|a_{n-1}) + p_c(o_i|a_n)}$$

($p_c(o_i|a_n)$ and $p_c(o_i|a_{n-1})$ are from (9.10)), and

$$p_2(k) = p_c(o_k|a_n)\frac{p_c(o_i|a_{n-1})}{p_c(o_i|a_{n-1}) + p_c(o_i|a_n)}$$

The Bayes risk in $\mathcal{H}$ is defined by

$$P_e(\vec{y}) = \sum_k \max_{1\leq j\leq n-1} p'(o_k|a_j)y_j$$

and a simple calculation shows that $P_e(\vec{y}) = P_e(\vec{x})$ whenever $\vec{x}$ satisfies (9.10) and $\vec{y}$ and $\vec{x}$ are related by (9.11). Hence the corner points of $P_e(\vec{x})$ over $\mathcal{H}$ can be obtained from those of $P_e(\vec{y})$.

The systems in $\mathbb{S}(\mathcal{C})$ are obtained from those in $\mathbb{S}(\mathcal{C}')$ in the following way. For each system in $\mathbb{S}(\mathcal{C}')$, replace the equation $y_1 + y_2 + \ldots + y_{n-1} = 1$ by $x_1 + x_2 + \ldots + x_{n-1} + x_n = 1$, and replace, in each equation, every occurrence of y_j by x_j, for j from 1 to $n-2$. Furthermore, if y_{n-1} occurs in an equation E of the form $y_{n-1} = 0$, then replace E by the equations $x_{n-1} = 0$ and $x_n = 0$. Otherwise, it must be the case that for some k_1, k_2, $p'(o_{k_1}|a_{n-1})y_{n-1}$ and $p'(o_{k_2}|a_{n-1})y_{n-1}$ occur in two of the other equations. In that case, replace $p'(o_{k_1}|a_{n-1})y_{n-1}$ by $p_c(o_{k_1}|a_{n-1})x_{n-1}$ if $p_1(k_1) \geq p_2(k_1)$, and by $p_c(o_{k_1}|a_n)x_n$ otherwise. Analogously for $p'(o_{k_2}|a_{n-1})y_{n-1}$. Finally, add the equation $p_c(o_i|a_n)x_n = p_c(o_i|a_{n-1})x_{n-1}$. It is easy to see that the uniqueness of solution is preserved by this transformation. The conversions to apply on the inequality part are trivial. □

Note that $\mathbb{S}(\mathcal{C})$ is finite, hence the U in Theorem 9.3.2 is finite as well.

9.3.1 An alternative characterization of the corner points

In this section we give an alternative characterization of the corner points of the Bayes risk. The reason is that the new characterization considers only systems of equations that are guaranteed to have a unique solution (for the equational part). As a consequence, we need to solve much less systems than those of Definition 9.3.1. We characterize these systems in terms of graphs.

Definition 9.3.3. *A* labeled undirected multigraph *is a tuple $G = (V, L, E)$ where V is a set of* vertices*, L is a set of* labels *and $E \subseteq \{(\{v, u\}, l) \mid v, u \in V, l \in L\}$ is a set of* labeled edges *(note that multiple edges are allowed between the same vertices). A graph is* connected *iff there is a path between any two vertices. A* tree *is a connected graph without cycles. We say that a tree $T = (V_T, L_T, E_T)$ is a tree of G iff $V_T \subseteq V, L_T \subseteq L, E_T \subseteq E$.*

Definition 9.3.4. *Let $\mathcal{C} = (\mathcal{A}, \mathcal{O}, p_c)$ be a channel. We define its associated graph $G(\mathcal{C}) = (V, L, E)$ as $V = \mathcal{A}$, $L = \mathcal{O}$ and $(\{a, a'\}, o) \in E$ iff $p_c(o|a)$, $p_c(o|a')$ are both positive.*

Definition 9.3.5. *Let $\mathcal{C} = (\mathcal{A}, \mathcal{O}, p_c)$ be a channel, let $n = |\mathcal{A}|$ and let $T = (V_T, L_T, E_T)$ be a tree of $G(\mathcal{C})$. The system of inequalities generated by T is defined as*

$$\begin{aligned} p_c(o_i|a_j)x_j &= p_c(o_i|a_k)x_k \\ p_c(o_i|a_j)x_j &\geq p_c(o_i|a_l)x_l \qquad\qquad \forall\ 1 \leq l \leq n \end{aligned}$$

for all edges $(\{a_j, a_k\}, o_i) \in E_T$, plus the equalities

$$\begin{aligned} x_j &= 0 \qquad\qquad \forall a_j \notin V_T \\ x_1 + \ldots + x_n &= 1 \end{aligned}$$

Let $\mathbb{T}(\mathcal{C})$ be the set of systems generated by all trees of $G(\mathcal{C})$.

An advantage of this characterization is that it allows an alternative, simpler proof of Theorem 9.3.2. The two proofs differ substantially. Indeed, the new one is non-inductive and uses the proof technique of Proposition 9.2.7.

Theorem 9.3.6. *Given a channel $\mathcal{C}$, the Bayes risk P_e associated to $\mathcal{C}$ is convexly generated by $(U, P_e(U))$, where U is the set of solutions to all solvable systems in $\mathbb{T}(\mathcal{C})$.*

Proof. Let $J = \{1, \ldots, |\mathcal{A}|\}, I = \{1, \ldots, |\mathcal{O}|\}$. We define

$$\begin{aligned} \mathrm{m}(\vec{x}, i) &= \max_{k \in J} p_c(o_i|a_k)x_k && \text{Maximum for column } i \\ \Psi(\vec{x}) &= \{i \in I \mid \mathrm{m}(\vec{x}, i) > 0\} && \text{Columns with non-zero maximum} \\ \Phi(\vec{x}, i) &= \{j \in J \mid p_c(o_i|a_j)x_j = \mathrm{m}(\vec{x}, i)\} && \text{Rows giving the maximum for col. } i \end{aligned}$$

The probability of error can be written as

$$P_e(\vec{x}) = 1 - \sum_{i \in I} p_c(o_i|a_{j(\vec{x},i)})x_{j(\vec{x},i)} \quad \text{where } j(\vec{x}, i) = \min \Phi(\vec{x}, i) \tag{9.12}$$

We now fix a point $\vec{x} \notin U$ and we are going to show that there exist $\vec{y}, \vec{z} \in D^{(n)}$ different than $\vec{x}$ such that $(\vec{x}, P_e(\vec{x})) = t(\vec{y}, P_e(\vec{y})) + \bar{t}(\vec{z}, P_e(\vec{z}))$. Let $M(\vec{x})$ be the indexes of the non-zero elements of $\vec{x}$, that is $M(\vec{x}) = \{j \in J \mid x_j > 0\}$ (we will simply write M if $\vec{x}$ is clear from the context. The idea is that we will "slightly" modify some elements in M without affecting any of the sets $\Phi(\vec{x}, i)$. We first define a relation $\sim$ on the set M as

$$j \sim k \quad \text{iff} \quad \exists i \in \Psi(\vec{x}) : j, k \in \Phi(\vec{x}, i)$$

and take $\approx$ as the reflexive and transitive closure of $\sim$ ($\approx$ is an equivalence relation). Now assume that $\approx$ has only one equivalence class, equal to M. Then we can create a tree T as follows: we start from a single vertex a_j, $j \in M$. At each step, we find a vertex a_j in the current tree such that $j \sim k$ for some $k \in M$ where a_k is not yet in the tree (such a vertex always exist since M is an equivalence class of $\approx$). Then we add a vertex a_k and an edge $(\{a_j, a_k\}, o_i)$ where i is the one from the definition of $\sim$. Note that since $i \in \Psi(\vec{x})$ we have that $p_c(o_i|a_j), p_c(o_i|a_k)$ are positive so this edge also belongs to $G(\mathcal{C})$.

Repeating this procedure creates a tree of $G(\mathcal{C})$ such that $\vec{x}$ is a solution to its corresponding system of inequalities, which is a contradiction since $\vec{x} \notin U$.

So we conclude that $\approx$ has at least two equivalence classes, say C, D. The idea is that we will add/subtract an ϵ from all elements of the class simultaneously, while preserving the relative ratio of the elements. We choose an $\epsilon > 0$ small enough such that $0 < x_j - \epsilon$ and $x_j + \epsilon < 1$ for all $j \in M$ and such that subtracting it from any element does not affect the relative order of the quantities $p_c(o_i|a_j)x_j$, that is

$$p_c(o_i|a_j)x_j > p_c(o_i|a_k)x_k \;\Rightarrow\; p_c(o_i|a_j)(x_j - \epsilon) > p_c(o_i|a_k)(x_k + \epsilon) \tag{9.13}$$

for all $i \in I, j, k \in M$.[2] Then we create two points $\vec{y}, \vec{z} \in D^{(n)}$ as follows:

$$y_j = \begin{cases} x_j - x_j\epsilon_1 & \text{if } j \in C \\ x_j + x_j\epsilon_2 & \text{if } j \in D \\ x_j & \text{otherwise} \end{cases} \qquad z_j = \begin{cases} x_j + x_j\epsilon_1 & \text{if } j \in C \\ x_j - x_j\epsilon_2 & \text{if } j \in D \\ x_j & \text{otherwise} \end{cases}$$

where $\epsilon_1 = \epsilon / \sum_{j \in C} x_j$ and $\epsilon_2 = \epsilon / \sum_{j \in D} x_j$ (note that $x_j\epsilon_1, x_j\epsilon_2 \leq \epsilon$) . It is easy to see that $\vec{x} = \frac{1}{2}\vec{y} + \frac{1}{2}\vec{z}$, it remains to show that $P_e(\vec{x}) = \frac{1}{2}P_e(\vec{y}) + \frac{1}{2}P_e(\vec{z})$.

We notice that $M(\vec{x}) = M(\vec{y}) = M(\vec{z})$ and $\Psi(\vec{x}) = \Psi(\vec{y}) = \Psi(\vec{z})$ since $x_j > 0$ iff $y_j > 0, z_j > 0$. We now compare $\Phi(\vec{x}, i)$ and $\Phi(\vec{y}, i)$. If $i \notin \Psi(\vec{x})$ then $p_c(o_i|a_k) = 0,\ \forall k \in M$ so $\Phi(\vec{x}, i) = \Phi(\vec{y}, i) = J$. Assuming $i \in \Psi(\vec{x})$, we first show that $p_c(o_i|a_j)x_j > p_c(o_i|a_k)x_k$ implies $p_c(o_i|a_j)y_j > p_c(o_i|a_k)y_k$. This follows from (9.13) since

$$p_c(o_i|a_j)y_j \geq p_c(o_i|a_j)(x_j - \epsilon) > p_c(o_i|a_k)(x_k + \epsilon) \geq p_c(o_i|a_k)y_k$$

This means that $k \notin \Phi(\vec{x}, i) \Rightarrow k \notin \Phi(\vec{y}, i)$, in other words

$$\Phi(\vec{x}, i) \supseteq \Phi(\vec{y}, i) \tag{9.14}$$

Now we show that $k \in \Phi(\vec{x}, i) \Rightarrow k \in \Phi(\vec{y}, i)$. Assume $k \in \Phi(\vec{x}, i)$ and let $j \in \Phi(\vec{y}, i)$ (note that $\Phi(\vec{y}, i) \neq \emptyset$). By (9.14) we have $j \in \Phi(\vec{x}, i)$ which means that $p_c(o_i|a_k)x_k = p_c(o_i|a_j)x_j$. Moreover, since $i \in \Psi(\vec{x})$ we have that j, k belong to the same equivalence class of $\approx$. If $j, k \in C$ then

$$\begin{aligned} p_c(o_i|a_k)y_k &= p_c(o_i|a_k)(x_k - x_k\epsilon_1) \\ &= p_c(o_i|a_j)(x_j - x_j\epsilon_1) \qquad\qquad p_c(o_i|a_k)x_k = p_c(o_i|a_j)x_j \\ &= p_c(o_i|a_j)y_j \end{aligned}$$

which means that $k \in \Phi(\vec{y}, i)$. Similarly for $j, k \in D$. If $j, k \notin C \cup D$ then $x_k = y_k$, $x_j = y_j$ and the same result is immediate. So we have $\Phi(\vec{x}, i) = \Phi(\vec{y}, i),\ \forall i \in I$. And symmetrically we can show that $\Phi(\vec{x}, i) = \Phi(\vec{z}, i)$. This

[2] Let $\delta_{i,j,k} = p_c(o_i|a_j)x_j - p_c(o_i|a_k)x_k$. It is sufficient to take

$$\epsilon < \min(\{\frac{\delta_{i,j,k}}{p_c(o_i|a_j) + p_c(o_i|a_j)} \mid \delta_{i,j,k} > 0\} \cup \{x_j \mid j \in M\})$$

implies that $j(\vec{x},i) = j(\vec{y},i) = j(\vec{z},i)$ (see (9.12)) so we finally have

$$\begin{aligned}\frac{1}{2}P_e(\vec{y}) + \frac{1}{2}P_e(\vec{z}) &= \frac{1}{2}\big(1 - \sum_{i\in I} p_c(o_i|a_{j(\vec{y},i)})y_{j(\vec{y},i)} + 1 - \sum_{i\in I} p_c(o_i|a_{j(\vec{z},i)})z_{j(\vec{z},i)}\big)\\ &= 1 - \sum_{i\in I} p_c(o_i|a_{j(\vec{x},i)})(\frac{1}{2}y_{j(\vec{x},i)} + \frac{1}{2}z_{j(\vec{x},i)})\\ &= P_e(\vec{x})\end{aligned}$$

Applying Proposition 9.2.7 completes the proof. □

We now show that both characterizations give the same systems of equations, that is $\mathbb{S}(\mathcal{C}) = \mathbb{T}(\mathcal{C})$.

Proposition 9.3.7. *Consider a system of inequalities of the form given in Definition 9.3.1. Then, the equational part has a unique solution if and only if the system is generated by a tree of $G(\mathcal{C})$.*

Proof. **if)** Assume that the system is generated by a tree of $G(\mathcal{C})$. Consider the variable corresponding to the root, say x_1. Express its children x_2, $\ldots$, x_k in terms of x_1. That is to say that, if the equation is $ax_1 = bx_2$, then we express x_2 as a/bx_1. At the next step, we express the children of x_2 in terms of x_2 an hence in terms of x_1, $\ldots$etc. Finally, we replace all $x_i's$ by their expressions in terms of x_1 in the equation $\sum_i x_i = 1$. This has exactly one solution.

only if) Assume by contradiction that the system is not generated by a tree. Then we we can divide the variables in at least two equivalence classes with respect to the equivalence relation $\approx$ defined in the proof of Theorem 9.3.6, and we can define the same $\vec{y}$ defined a few paragraphs later. This $\vec{y}$ is a different solution of the same system (also for the inequalities). □

The advantage of Definition 9.3.5 is that it constructs directly solvable systems, in contrast to Definition 9.3.1 which would oblige us to solve all systems of the given form and keep only the solvable ones. We finally give the complexity of computing the corner points of P_e using the tree characterization, which involves counting the number of trees of $G(\mathcal{C})$.

Proposition 9.3.8. *Let $\mathcal{C} = (\mathcal{A}, \mathcal{O}, p_c)$ be a channel and let $n = |\mathcal{A}|, m = |\mathcal{O}|$. Computing the set of corner points of P_e for $\mathcal{C}$ can be performed in $O(n(nm)^{n-1})$ time.*

Proof. To compute the set of corner points of P_e we need to solve all the systems of inequalities in $\mathbb{T}(\mathcal{C})$. Each of those is produced by a tree of $G(\mathcal{C})$. In the worst case, the matrix of the channel is non-zero everywhere, in which case $G(\mathcal{C})$ is the complete multigraph K_n^m of n vertices, each pair of which is connected by exactly m edges. Let K_n^1 be the complete graph of n vertices (without multiple edges). Cayley's formula ([Cay89]) gives its number $\sigma(K_n^1)$ of spanning trees:

$$\sigma(K_n^1) = n^{n-2} \tag{9.15}$$

We now want to compute the total number $\tau(K_n^1)$ of trees of K_n^1. To create a tree of k vertices, we have $\binom{n}{k}$ ways to select k out of the n vertices of K_n^1 and $\sigma(K_k^1)$ ways to form a tree with them. Thus

$$\begin{aligned}
\tau(K_n^1) &= \sum_{k=1}^{n} \binom{n}{k} \sigma(K_k^1) \\
&= \sum_{k=1}^{n} \frac{n!}{k!(n-k)!}\, k^{k-2} && (9.15) \\
&= \sum_{k=1}^{n} \frac{1}{(n-k)!}\, (k+1) \cdot \ldots \cdot n \cdot k^{k-2} \\
&\leq \sum_{k=1}^{n} \frac{1}{(n-k)!}\, n^{n-k} \cdot n^{k-2} && k+i \leq n \\
&= n^{n-2} \sum_{l=0}^{n-1} \frac{1}{l!} && \text{set } l = n-k \\
&\leq e \cdot n^{n-2} && \text{since } \textstyle\sum_{l=0}^{\infty} \frac{1}{l!} = e
\end{aligned}$$

thus $\tau(K_n^1) \in O(n^{n-2})$. Each tree of K_n^m can be produced by a tree of K_n^1 by exchanging the edge between two vertices with any of the m available edges in K_n^m. Since a tree of K_n^m has at most $n-1$ edges, for each tree of K_n^1 we can produce at most m^{n-1} trees of K_n^m. Thus

$$\tau(K_n^m) \leq m^{n-1} \tau(K_n^1) \in O(m^{n-1} n^{n-2})$$

Finally, for each tree we have to solve the corresponding system of inequalities. Due to the form of this system, computing the solution can be done in $O(n)$ time by expressing all variables x_i in terms of the root of the tree, and then replace them in the equation $\sum_i x_i = 1$. On the other hand, for each solution we have to verify as many as $n(n-1)$ inequalities, so in total the solution can be found in $O(n^2)$ time. Thus, computing all corner points takes $O(n^2 m^{n-1} n^{n-2}) = O(n(nm)^{n-1})$ time. □

Note that, to improve a bound using Proposition 9.2.5, we need to compute the maximum ratio $f(\vec{u})/g(\vec{u})$ of all corner points $\vec{u}$. Thus, we need only to compute these points, not to store them. Still, as shown in the above proposition, the number of the systems we need to solve in the general case is huge. However, as we will see in Section 9.4.1, in certain cases of symmetric channel matrices the complexity can be severely reduced to even polynomial time.

9.3.2 Examples

Example 9.3.9 (Binary hypothesis testing)**.** *The case $n = 2$ is particularly simple: the systems generated by $\mathcal{C}$ are all those of the form*

$$\{p_c(o_i|a_1)x_1 = p_c(o_i|a_2)x_2 \ , \ x_1 + x_2 = 1\}$$

plus the two systems

$$\{x_1 = 0 \ , \ x_1 + x_2 = 1\}$$
$$\{x_2 = 0 \ , \ x_1 + x_2 = 1\}$$

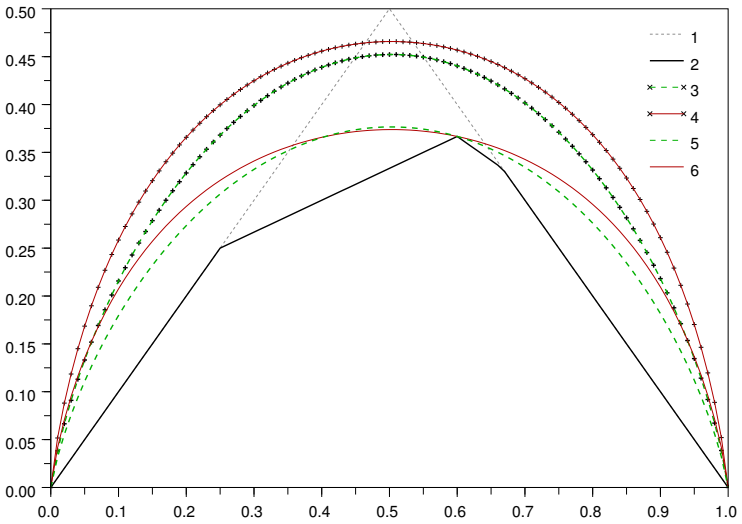


Figure 9.1: The graph of the Bayes risk for the channel in Example 9.3.9 and various bounds for it. Curve 1 represents the probability of error if we ignore the observables, i.e. the function $f(\vec{x}) = 1 - \max_j x_j$. Curve 2 represents the Bayes risk $P_e(\vec{x})$. Curve 3 represents the Hellman-Raviv bound $\frac{1}{2}H(A|O)$. Curve 4 represents the Santhi-Vardy bound $1 - 2^{-H(A|O)}$. Finally, Curves 5 and 6 represent the improvements on 3 and 4, respectively, that we get by applying the method induced by our Proposition 9.2.5.

These systems are always solvable, hence we have $m + 2$ corner points, where we recall that m is the cardinality of O.

Let us illustrate this case with a concrete example: let $\mathcal{C}$ be the channel determined by the following matrix:

	o_1	o_2	o_3
a_1	1/2	1/3	1/6
a_2	1/6	1/2	1/3

The systems generated by $\mathcal{C}$ are:

$$\begin{aligned}
\{x_1 = 0 \quad &, \quad x_1 + x_2 = 1\} \\
\{\tfrac{1}{2}x_1 = \tfrac{1}{6}x_2 \quad &, \quad x_1 + x_2 = 1\} \\
\{\tfrac{1}{3}x_1 = \tfrac{1}{2}x_2 \quad &, \quad x_1 + x_2 = 1\} \\
\{\tfrac{1}{6}x_1 = \tfrac{1}{3}x_2 \quad &, \quad x_1 + x_2 = 1\} \\
\{x_1 = 0 \quad &, \quad x_1 + x_2 = 1\}
\end{aligned}$$

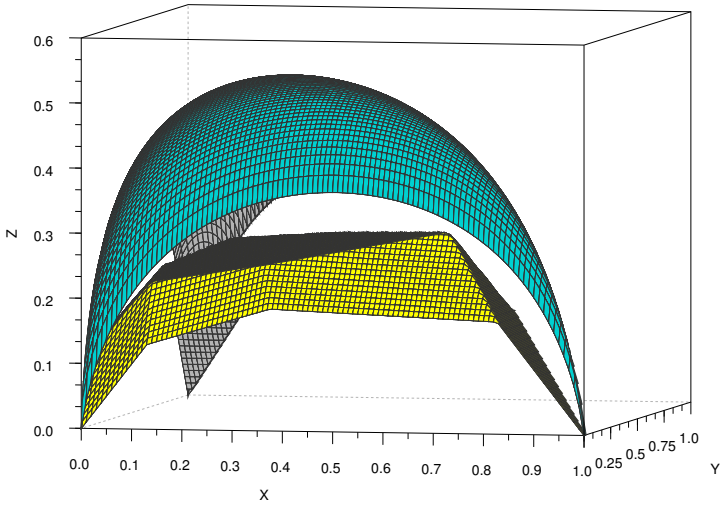


Figure 9.2: Ternary hypothesis testing. The solid curve represents the Bayes risk for the channel in Example 9.3.10, while the dotted curve represents the Santhi-Vardy bound $1 - 2^{-H(A|O)}$.

The solutions of these systems are: $(0, 1)$, $(1/4, 3/4)$, $(3/5, 2/5)$, $(2/3, 1/3)$, *and* $(1, 0)$, *respectively. The value of* P_e *on these points is* 0, $1/4$, $3/10$ *(maximum),* $1/3$, *and* 0 *respectively, and* P_e *is piecewise linear between these points, i.e. it can be generated by convex combination of these points and its value on them. Its graph is illustrated in Figure 9.1, where* x_1 *is represented by* x *and* x_2 *by* $1 - x$.

Example 9.3.10 (Ternary hypothesis testing)**.** *Let us consider now a channel* $\mathcal{C}$ *with three inputs. Assume the channel has the following matrix:*

	o_1	o_2	o_3
a_1	2/3	1/6	1/6
a_2	1/8	3/4	1/8
a_3	1/10	1/10	4/5

The following is an example of a solvable system generated by $\mathcal{C}$*:*

$$\begin{aligned} \tfrac{2}{3}x_1 &= \tfrac{1}{8}x_2 \\ \tfrac{1}{8}x_2 &= \tfrac{4}{5}x_3 \\ x_1 + x_2 + x_3 &= 1 \\ \tfrac{2}{3}x_1 &\geq \tfrac{1}{10}x_3 \\ \tfrac{1}{8}x_2 &\geq \tfrac{1}{6}x_1 \end{aligned}$$

Another example is

$$\begin{aligned} \tfrac{1}{6}x_1 &= \tfrac{3}{4}x_2 \\ x_3 &= 0 \\ x_1 + x_2 + x_3 &= 1 \end{aligned}$$

The graph of P_e is depicted in Figure 9.2, where x_3 is represented by $1 - x_1 - x_2$.

9.4 Application: Crowds

In this section we discuss how to compute the channel matrix for a given protocol using automated tools, and use it to improve the bound for the probability of error. We illustrate our ideas on a variation of Crowds.

In this protocol, described in detail in Section 3.2.2, a user (called the *initiator*) wants to send a message to a web server without revealing its identity. To achieve that, he routes the message through a crowd of users participating in the protocol. The routing is performed using the following protocol: in the beginning, the initiator selects randomly a user (called a *forwarder*), possibly himself, and forwards the request to him. A forwarder, upon receiving a message, performs a probabilistic choice. With probability p_f (a parameter of the protocol) he selects a new user and forwards once again the message. With probability $1 - p_f$ he sends the message directly to the server.

It is easy to see that the initiator is strongly anonymous with respect to the server, as all users have the same probability of being the forwarder who finally delivers the message. However, the more interesting case is when the attacker is one of the users of the protocol (called a *corrupted* user) which uses his information to find out the identity of the initiator. A corrupted user has more information than the server since he sees other users forwarding the message through him. The initiator, being the in first in the path, has greater probability of forwarding the message to the attacker than any other user, so strong anonymity cannot hold. However, as shown in Section 6.5.1, Crowds satisfies probable innocence, under certain conditions on the number of corrupted users.

In our analysis, we consider two network topologies. In the first, used in the original presentation of Crowds, all users are assumed to be able to communicate with any other user, in other words the network graph is a clique. In this case, the channel matrix is symmetric and easy to compute. Moreover, due to the symmetry of the matrix, the corner points of the probability of error are fewer in number and have a simple form.

However, having a clique network is not always feasible in practice, as it is the case for example in distributed systems. As the task of computing the matrix becomes much harder in a non-clique network, we employ model-checking tools to perform it automatically. The set of corner points, being finite, can also be computed automatically by solving the corresponding systems of inequalities.

9.4.1 Crowds in a clique network

We consider an instance of Crowds with m users, of which n are honest and $c = m - n$ are corrupted. To construct the matrix of the protocol, we start by identifying the set of anonymous facts, which depends on what the system is trying to hide. In protocols where one user performs an action of interest (like initiating a message in our example) and we want to protect his identity, the set $\mathcal{A}$ would be the set of the users of the protocol. Note that the corrupted users should not be included in this set, since we cannot expect the attacker's own actions to be hidden from him. So in our case we have $\mathcal{A} = \{u_1, \dots u_n\}$ where u_i means that user i is the initiator.

The set of observables should also be defined, based on the visible actions of the protocol and on the various assumptions made about the attacker. In Crowds we assume that the attacker does not have access to the entire network (such an attacker would be too powerful for this protocol) but only to the messages that pass through a corrupted user. Each time a user i forwards the message to a corrupted user we say that he is *detected* which corresponds to an observable action in the protocol. Along the lines of other studies of Crowds (e.g. [Shm04]) we suppose that an attacker will not forward a message himself, since by doing so he would not gain more information. So at each execution of the protocol there is at most one detected user and we have $\mathcal{O} = \{d_1, \dots, d_n\}$ where d_j means that user j was detected.

Now we need to compute the probabilities $p_c(d_j|u_i)$ for all $1 \leq i, j \leq n$. We first observe some symmetries of the protocol. First, the probability of observing the initiator is the same, independently of who is the initiator. We denote this probability by α. Moreover, the probability of detecting a user other than the initiator is the same for all other users. We denote this probability by β. It can be shown ([RR98]) that

$$\alpha = c\,\frac{1 - \frac{n-1}{m}p_f}{m - np_f} \qquad\qquad \beta = \alpha - \frac{c}{m}$$

Note that there is also the possibility of not observing any user, if the message arrives to a server without passing through any corrupted user. To compute the matrix, we condition on the event that some user was observed, which is reasonable since otherwise anonymity is not an issue. Thus the conditional probabilities of the matrix are:

$$p_c(d_j|u_i) = \begin{cases} \frac{\alpha}{s} & \text{if } i = j \\ \frac{\beta}{s} & \text{otherwise} \end{cases}$$

where $s = \alpha + (n-1)\beta$. The matrix for $n = 20, c = 5, p_f = 0.7$ is shown in Figure 9.3.

An advantage of the symmetry is that the corner points of the probability of error for such a matrix have a simple form.

Proposition 9.4.1. *Let $(\mathcal{A}, \mathcal{O}, p_c)$ be a channel. Assume that all values of the matrix $p_c(\cdot|\cdot)$ are either α or β, with $\alpha, \beta > 0$, and that there is at most one α per column. Then all solutions to the systems of Definition 9.3.5 have at most two distinct non-zero elements, equal to x and $\frac{\alpha}{\beta}x$ for some $x \in (0, 1]$.*

	d_1	d_2	$\dots$	d_{20}
u_1	0.468	0.028	$\dots$	0.028
u_2	0.028	0.468	$\dots$	0.028
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
u_{20}	0.028	0.028	$\dots$	0.468

Figure 9.3: The channel matrix of Crowds for $n = 20, c = 5, p_f = 0.7$. The events u_i, d_j mean that user i is the initiator and user j was detected respectively.

Proof. Since all values of the matrix are either α or β, the equations of all the systems in Definition 9.3.5 are of the form $x_i = x_j$ or $\alpha \cdot x_i = \beta \cdot x_j$.[3] Assume that a solution of such a system has three distinct non-zero elements $x_1 > x_2 > x_3 > 0$. We consider the following two cases:

1. x_2, x_3 are related to each other by an equation. Since $x_2 > x_3$ this equation can only be $\alpha \cdot x_2 = \beta \cdot x_3$, where $p_c(o|a_2) = \alpha$ for some observable o. Since there is at most one α per column we have $p_c(o|a_1) = \beta$ and thus $p_c(o|a_1)x_1 = \beta\, x_1 > \beta\, x_3 = \alpha\, x_2 = p_c(o|a_2)x_2$ which violates the inequalities of Definition 9.3.5.

2. x_2, x_3 are not related to each other. Thus they must be related to x_1 by two equations (assuming $\alpha > \beta$) $\beta \cdot x_1 = \alpha \cdot x_2$ and $\beta \cdot x_1 = \alpha \cdot x_3$. This implies that $x_2 = x_3$ which is a contradiction.

Similarly for more than three non-zero elements. □

The above proposition allows us to efficiently compute the scaling factor of Proposition 9.2.5 to improve the Santhi-Vardy bound.

Proposition 9.4.2. *Let $(\mathcal{A}, \mathcal{O}, p_c)$ be a channel with $n = |\mathcal{A}|$. Assume that all columns and all rows of the matrix $p_c(\cdot|\cdot)$ have exactly one element equal to $\alpha > 0$ and all others equal to $\beta > 0$. Then the scaling factor of Proposition 9.2.5 can be computed in $O(n^2)$ time.*

Proof. By Proposition 9.4.1, all corner points of P_e have two distinct non-zero elements x and $\frac{\alpha}{\beta}x$. If we fix the number k_1 of elements equal to x and the number k_2 of elements equal to $\frac{\alpha}{\beta}x$ then x can be uniquely computed in constant time. Due to the symmetry of the matrix, P_e as well as the Santhi-Vardy bound will have the same value for all corner points with the same k_1, k_2. So it is sufficient to compute the ratio in only one of them. Then by varying k_1, k_2, we can compute the best ratio without even computing all the corner points. Note that there are $O(n^2)$ possible values of k_1, k_2 and since we need to compute one point for each of them, the total computation can be performed in $O(n^2)$ time. □

[3] Note that by construction of $G(\mathcal{C})$ the coefficients of all equations are non-zero, so in our case either α or β.

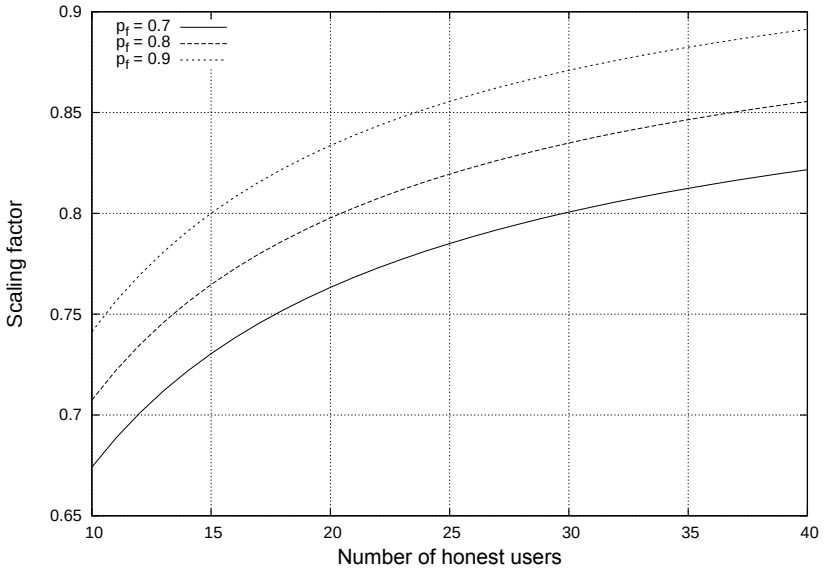


Figure 9.4: The improvement (represented by the scaling factor) with respect to the Santhi-Vardy bound for various instances of Crowds.

We can now apply the algorithm described above to compute the scaling factor $c_o \leq 1$. Multiplying the Santhi-Vardy bound by c_o will give us an improved bound for the probability of error. The results are shown in Figure 9.4. We plot the obtained scaling factor while varying the number of honest users, for $c = 5$ and for various values of the parameter p_f. A lower scaling factor means a bigger improvement with respect to the Santhi-Vardy bound. We remind that we probability of error, in this case, gives the probability that the attacker "guesses" the wrong sender. The higher it is, the more secure is the protocol. It is worth noting that the scaling factor increases when the number of honest users increases or when the probability of forwarding increases. In other words, the improvement is better when the probability of error is smaller (and the system is less anonymous). When increasing the number of users (without increasing the number c of corrupted ones), the protocol offers more anonymity and the capacity increases. In this case the Santhi-Vardy bound becomes closer to the corner points of P_e and there is little room for improvement.

9.4.2 Crowds in a grid network

We now consider a grid-shaped network as shown in Figure 9.5. In this network there is a total of nine users, each of whom can only communicate with the four that are adjacent to him. We assume that the network "wraps" at the edges, so user 1 can communicate with both user 3 and user 7. Also, we assume that the only corrupted user is user 5.

In this example we have relaxed the assumption of a clique network, showing that a model-checking approach can be used to analyze more complicated network topologies (but of course is limited to specific instances). Moreover, the lack of homogeneity in this network creates a situation where the maximum

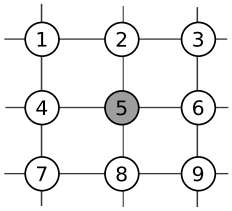


Figure 9.5: An instance of Crowds with nine users in a grid network. User 5 is the only corrupted one.

	d_2	d_4	d_6	d_8
u_1	0.33	0.33	0.17	0.17
u_3	0.33	0.17	0.33	0.17
u_7	0.17	0.33	0.17	0.33
u_9	0.17	0.17	0.33	0.33
u_2	0.68	0.07	0.07	0.17
u_4	0.07	0.68	0.17	0.07
u_6	0.07	0.17	0.68	0.07
u_8	0.17	0.07	0.07	0.68

Figure 9.6: The channel matrix of the examined instance of Crowds. The symbols u_i, d_j mean that user i is the initiator and user j was detected respectively.

probability of error is given by a non-uniform input distribution. This emphasizes the importance of abstracting from the input distribution: assuming a uniform one would be not justified in this example.

Similarly to the previous example, the set of anonymous events will be $\mathcal{A} = \{u_1, u_2, u_3, u_4, u_6, u_7, u_8, u_9\}$ where u_i means that user i is the initiator. For the observable events we notice that only the users 2, 4, 6 and 8 can communicate with the corrupted user. Thus we have $\mathcal{O} = \{d_2, d_4, d_6, d_8\}$ where d_j means that user j was detected.

To compute the channel's matrix, we have modeled Crowds in the language of the PRISM model-checker ([KNP04]), which is essentially a formalism to describe Markov Decision Processes. PRISM can compute the probability of reaching a specific state starting from a given one. Thus, each conditional probability $p_c(d_j|u_i)$ is computed as the probability of reaching a state where the attacker has detected user j, starting from the state where i is the initiator. Similarly to the previous example, we compute all probabilities conditioned on the fact that some observation was made, which corresponds to normalizing the rows of the matrix.

In Figure 9.6 the channel matrix is displayed for the examined Crowds instance, computed using probability of forwarding $p_f = 0.8$. We have split the users in two groups, the ones who cannot communicate directly with the corrupted user, and the ones who can. When a user of the first group, say user 1, is the initiator, there is a higher probability of detecting the users that are adjacent to him (users 2 and 4) than the other two (users 6 and 8) since the

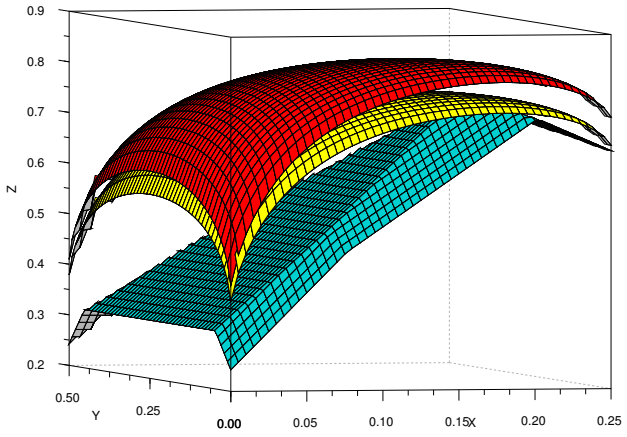


Figure 9.7: The lower curve is the probability of error in the examined instance of Crowds. The upper two are the Santhi and Vardy's bound and its improved version.

message needs two steps to arrive to the latters. So $p_c(d_2|u_1) = p_c(d_4|u_1) = 0.33$ are greater than $p_c(d_6|u_1) = p_c(d_8|u_1) = 0.17$. In the second group users have direct communication to the attacker, so when user 2 is the initiator, the probability $p_c(d_2|u_2)$ of detecting him is high. From the remaining three observables d_8 has higher probability since user 8 can be reached from user 2 in one step, while users 4 and 6 need two steps. Inside each group the rows are symmetric since the users behave similarly. However between the groups the rows are different which is caused by the different connectivity to the corrupted user 5.

We can now compute the probability of error for this instance of Crowds, which is displayed in the lower curve of Figure 9.7. Since we have eight users, to plot this function we have to map it to the three dimensions. We do this by considering the users 1, 3, 7, 9 to have the same probability x_1, the users 2, 8 to have the same probability x_2 and the users 4, 6 to have the same probability $1 - x_1 - x_2$. Then we plot P_e as a function of x_1, x_2 in the ranges $0 \leq x_1 \leq 1/4$, $0 \leq x_2 \leq 1/2$. Note that when $x_1 = x_2 = 0$ there are still two users (4, 6) among whom the probability is distributed, so P_e is not 0. The upper curve of Figure 9.7 shows the Santhi and Vardy's bound on the probability of error. Since all the rows of the matrix are different the bound is not tight, as illustrated.

We can obtain a better bound by applying Proposition 9.2.5. The set of corner points, characterized by Theorem 9.3.2, is finite and can be automatically constructed by solving the corresponding systems of inequalities. A prototype tool that computes the set of corner points for an arbitrary matrix is available at [Cha07]. After finding the corner points, we compute the scaling factor $c_o = \max_u P_e(\vec{u})/h(\vec{u})$, where h is the original bound, and take $c_o \cdot h$ as the improved bound. In our example we found $c_o = 0.925$ which was given for the corner point $\vec{u} = (0.17, 0.17, 0.17, 0.17, 0.08, 0.08, 0.08, 0.08)$.

9.5 Protocol composition

In this section we consider the case where a protocol is executed multiple times by the same user, either forced by the attacker himself or by some external factor. For instance, in Crowds users send messages along randomly selected routes. For various reasons this path might become unavailable, so the user will need to create an new one, thus re-executing the protocol. If the attacker is part of the path, he could also cause it to fail by stop forwarding messages, thus obliging the sender to recreate it (unless measures are taken to prevent this, as it is done in Crowds).

From the point of view of hypothesis testing, the above scenario corresponds to performing the experiment multiple times while the same hypothesis holds through the repetition. We assume that the the outcomes of the repeated experiments are independent. This corresponds to assuming that the protocol is memoryless, i.e. each time it is reactivated, it works according to the same probability distribution, independently from what happened in previous sessions.

As in the previous sections, we consider the Bayesian approach, which requires the knowledge of the matrix of the protocol and of the *a priori* distribution of the hypotheses, and tries to infer the *a posteriori* probability of the actual hypothesis w.r.t. a given sequence of observations. As argued in previous chapters, the first assumption (knowledge of the matrix of the protocol) is usually granted in an anonymity setting, since the way the protocol works is public. The second assumption may look too strong, since the attacker does not usually know the distribution of the anonymous events. However, in this section we will show that, under certain conditions, the *a priori* distribution becomes less and less relevant with the repetition of the experiment, and, at the limit, it does not matter at all.

Let $S = (\mathcal{A}, \mathcal{O}, p_c)$ be an anonymity system. The situation in which the protocol is re-executed n times with the same event a as input corresponds to the n-repetition S^n of S, defined in Section 4.2. The observables in S^n are sequences $\vec{o} = (o_1, \ldots, o_n)$ of observables of S and, since we consider the repetitions to be independent, the conditional probabilities for S^n will be given by[4]

$$p_c(\vec{o}|a) = \prod_{i=1}^{n} p_c(o_i|a) \tag{9.16}$$

As discussed in Section 9.1 the decision function adopted by the adversary to infer the anonymous action from the sequence of observables will be a function of the form $f_n : \mathcal{O}^n \to \mathcal{A}$. Also let Let $E_{f_n} : \mathcal{A} \to 2^{\mathcal{O}^n}$ be the error region of f_n and let $\eta_n : \mathcal{A} \to [0, 1]$ be the function that associates to each $a \in \mathcal{A}$ the probability of inferring the wrong input event on the basis of f_n, namely $\eta_n(a) = \sum_{\vec{o} \in E_{f_n}(a)} p(\vec{o}|a)$. Then the probability of error of f_n will be the expected value of $\eta_n(a)$:

$$P_{f_n} = \sum_{a \in A} p(a)\eta_n(a)$$

[4]With a slight abuse of notations we denote by p_c the probability matrix of both S and S^n. It will be clear from the context to which we refer to.

The MAP rule and the notion of MAP decision function can be extended to the case of protocol repetition in the obvious way. Namely a MAP decision function in the context of protocol repetition is a function f_n such that for each $\vec{o} \in \mathcal{O}^n$ and $a, a' \in \mathcal{A}$

$$f_n(\vec{o}) = a \Rightarrow p(\vec{o}|a)p(a) \geq p(\vec{o}|a')p(a')$$

Also in the case of protocol repetition the MAP rule gives the best possible result, namely if f_n is a MAP decision function then $P_{f_n} \leq P_{h_n}$ for any other decision function h_n.

9.5.1 Independence from the input distribution

In this section we will see that under a certain condition on the matrix of the protocol, and for n large enough, the knowledge of the input distribution becomes unnecessary for hypothesis testing, in the sense that the MAP decision functions can be approximated by other decision functions that do not depend on the distribution on $\mathcal{A}$.

The following definition establishes the condition on the matrix.

Definition 9.5.1. *Given an anonymity system $(\mathcal{A}, \mathcal{O}, p_c)$, we say that the system is determinate iff all rows of the matrix p_c are pairwise different, i.e. the probability distributions $p_c(\cdot|a)$, $p_c(\cdot|a')$ are different for each pair a, a' with $a \neq a'$.*

Next proposition shows that if a protocol is determinate, then it can be approximated by a decision function which compares only the elements along the column corresponding to the observed event, without considering the input probabilities. By "approximated" we mean that as n increases, the probability of the subset of $\mathcal{O}^n$ in which the two functions give the same result converges to 1.

This property is based on a remark in [CT91], page 316, stating that, for n large enough, in the fraction $p(\vec{o}|a)p(a)/p(\vec{o}|a')p(a')$ the factor $p(a)/p(a')$ is dominated by the factor $p(\vec{o}|a)/p(\vec{o}|a')$ (provided, one needs to add, that the latter is different from 1). In [CT91] they give also a sketch of the proof of this remark; the proof of our proposition is is a development of that sketch.

Proposition 9.5.2. *Given a determinate anonymity system $(\mathcal{A}, \mathcal{O}, p_c)$, for any distribution $p_{\mathcal{A}}$ on $\mathcal{A}$, any MAP decision functions f_n and any decision function $g_n : \mathcal{O}^n \to \mathcal{A}$ such that*

$$g_n(\vec{o}) = a \;\Rightarrow\; p_c(\vec{o}|a) \geq p_c(\vec{o}|a') \quad \forall \vec{o} \in \mathcal{O}^n \forall a, a' \in \mathcal{A}$$

we have that g_n approximates f_n. Namely, for any $\epsilon > 0$, there exists n such that the probability of the set $\{\vec{o} \in \mathcal{O}^n \mid f_n(\vec{o}) \neq g_n(\vec{o})\}$ is smaller than ϵ.

Proof. For any value $o \in \mathcal{O}$, and for any sequence of observable outcomes $\vec{o} \in \mathcal{O}^n$, let $n(o, \vec{o})$ denote the number of o's that occur in $\vec{o}$. Let a be the actual input. Observe that, by the strong law of large numbers ([CT91]), for any $\delta > 0$ the probability of the set $\{\vec{o} \in \mathcal{O}^n \mid \forall o \in \mathcal{O}\ |n(o, \vec{o})/n - p(o|a)| < \delta\}$ goes to 1 as n goes to ∞. We show that, as a consequence of the above observation, the

probability of the set $S = \{\vec{o} \in \mathcal{O}^n \,|\, \forall a' \neq a \; p(\vec{o}|a)p(a) > p(\vec{o}|a')p(a')\}$ goes to 1 as n goes to ∞. In fact, $p(\vec{o}|a)p(a) > p(\vec{o}|a')p(a')$ iff

$$\frac{1}{n}\log\frac{p(\vec{o}|a)p(a)}{p(\vec{o}|a')p(a')} > 0$$

Then we have

$$\begin{aligned}
\frac{1}{n}\log\frac{p(\vec{o}|a)}{p(\vec{o}|a')} &= \frac{1}{n}\log\prod_{i=1}^{n}\frac{p(o_i|a)}{p(o_i|a')} && \text{(by (9.16))}\\
&= \frac{1}{n}\sum_{i=1}^{n}\log\frac{p(o_i|a)}{p(o_i|a')} \\
&= \frac{1}{n}\sum_{o\in\mathcal{O}} n(o,\vec{o})\log\frac{p(o|a)}{p(o|a')} && \text{(by definition of } n(o,\vec{o})\text{)}\\
&\underset{n\to\infty}{\longrightarrow} \sum_{o\in\mathcal{O}} p(o|a)\log\frac{p(o|a)}{p(o|a')} && \text{(strong law of large numb.)}\\
&= D(p(\cdot|a) \parallel p(\cdot|a')) && \text{(Kullback–Leibler distance)}
\end{aligned}$$

so

$$\begin{aligned}
\frac{1}{n}\log\frac{p(\vec{o}|a)p(a)}{p(\vec{o}|a')p(a')} &= \frac{1}{n}\log\frac{p(\vec{o}|a)}{p(\vec{o}|a')} + \frac{1}{n}\log\frac{p(a)}{p(a')} \\
&\underset{n\to\infty}{\longrightarrow} D(p(\cdot|a) \parallel p(\cdot|a')) && \text{(since } \frac{1}{n}\log\frac{p(a)}{p(a')} \underset{n\to\infty}{\longrightarrow} 0\text{)}\\
&> 0 && \text{(by determinacy)}
\end{aligned}$$

Given a MAP decision function f_n, consider now the set $S' = \{\vec{o} \in \mathcal{O}^n \,|\, f_n(\vec{o}) = a\}$. Because of the definition of f_n, we have that $S \subseteq S'$. Hence also the probability of the set S' goes to 1 as n goes to ∞. Following a similar reasoning, we can prove that for any g_n satisfying the premises of proposition, the probability of the set $\{\vec{o} \in \mathcal{O}^n \,|\, g_n(\vec{o}) = a\}$ goes to 1 as n goes to ∞. We can therefore conclude that the same holds for the probability of the set $\{\vec{o} \in \mathcal{O}^n \,|\, g_n(\vec{o}) = f_n(\vec{o})\}$. □

The conditional probability $p(o|a)$ (resp. $p(\vec{o}|a)$) is called *likelihood* of a given o (resp. $\vec{o}$). The criterion for the definition of g_n used in Proposition 9.5.2 is to choose the a which maximizes the likelihood of o, and it is known in literature as the *Maximum Likelihood rule*. In the following we will call *Maximum Likelihood* (*ML*) decision functions those functions that, like g_n, satisfy the ML criterion. The Maximum Likelihood principle is very popular in statistic, its advantage over the Bayesian approach being that it does not require any knowledge of the a priori probability on A.

9.5.2 Bounds on the probability of error

In this section we discuss some particular cases of matrices and the corresponding bounds on the probability of error associated to the MAP and ML decision functions. We also discuss the probability of error in relation to various bounds on the capacity of the corresponding channel.

Determinate matrix We start with the bad case (from the anonymity point of view), which is when the matrix is determinate:

Proposition 9.5.3. *Given a determinate anonymity system $(\mathcal{A}, \mathcal{O}, p_c)$, for any distribution $p_{\mathcal{A}}$ on $\mathcal{A}$ and for any $\epsilon > 0$, there exists n such that the property*

$$g_n(\vec{o}) = a \;\Rightarrow\; p_c(\vec{o}|a) \geq p_c(\vec{o}|a') \quad \forall a' \in \mathcal{A}$$

determines a unique decision function g_n on a set of probability greater than $1-\epsilon$, and the probability of error P_{g_n} is smaller than ϵ.

Proof. Given $\vec{o} \in \mathcal{O}^n$, define $g_n(\vec{o}) = a$ iff a is the value of $\mathcal{A}$ for which $p(\vec{o}|a)$ is greatest. By following the same lines as in the proof of Proposition 9.5.2, we have that the set $\{\vec{o} \in \mathcal{O}^n \,|\, \forall a' \in \mathcal{A}\ p(\vec{o}|a) > p(\vec{o}|a')\}$ has probability greater than $1-\epsilon$ for n sufficiently large. Consequently, the choice of a is unique.

As for P_{g_n}, we observe that for n sufficiently large the set $E_{g_n} = \{\vec{o} \in \mathcal{O}^n \,|\, \exists a' \in \mathcal{A}\ p(\vec{o}|a) \leq p(\vec{o}|a')\}$ has probability smaller than ϵ. Hence $\eta_n(a) = \sum_{\vec{o} \in E_{g_n}(a)} p(\vec{o}|a) < \epsilon$ and $P_{g_n} = \sum_{a \in \mathcal{A}} p(a)\eta_n(a) < \epsilon$. □

Proposition 9.5.3 and its proof tell us that, in case of determinate matrices, there is essentially only one decision function, and its value is determined, for n sufficiently large, by the a for which $p(\vec{o}|a)$ is greatest.

One extreme case of determinate matrix is when the capacity is 0.

Maximum capacity If the channel has no noise, which means that for each observable $\vec{o}$ there exists at most one a such that $p_c(\vec{n}|a) \neq 0$, then the probability of error for an ML function is 0 for every input distribution. In fact

$$\begin{aligned} P_{g_n} &= 1 - \textstyle\sum_{\vec{o}} \max_j p(\vec{o}|a_j)x_j \\ &= 1 - \textstyle\sum_j \sum_{\vec{o}} p(\vec{o}|a_j)x_j \\ &= 1 - \textstyle\sum_j x_j = 0 \end{aligned}$$

Hence in the case of capacity 0 the error is 0 for every n. In particular, it is already 0 after the first observation (i.e. we are already certain about which hypothesis holds) and we don't need to repeat the protocol.

The same holds for a MAP function, the assumption that $p_c(\vec{n}|a) \neq 0$ for at most one a implies that $\max_j(p(\vec{o}|a_j)x_j) = \max_j p(\vec{o}|a_j)x_j$.

Identical rows Consider now the case in which determinacy does not hold, i.e. when there are at least two identical rows in the matrix, in correspondence, say, of a_1 and a_2. In such case, for the sequences $\vec{o} \in \mathcal{O}^n$ such that $p_c(\vec{o}|a_1)$ (or equivalently $p_c(\vec{o}|a_2)$) is maximal, the value of a ML function g_n is not uniquely determined, because we could choose either a_1 or a_2. Hence we have more than one ML decision function.

More in general, if there are k identical rows $a_1, a_2, \ldots, a_k$, the ML criterion gives k different possibilities each time we get an observable $\vec{o} \in \mathcal{O}^n$ for which $p_c(\vec{o}|a_1)$ is maximal. Intuitively this is a situation which may induce an error which is difficult to get rid of, even by repeating the protocol many times.

The situation is different and if we know the a priori distribution and we use a MAP function f_n. In this case we have to maximize $p(a)p(\vec{o}|a)$ and even

in case of identical rows, the a priori knowledge can help to make a sensible guess about the most likely a.

Both in the case of the ML and of the MAP functions, however, we shown that the probability of error is bound from below by an expression that depends on the probabilities of $a_1, a_2, \ldots, a_k$ only. In fact, we can show that this is the case for *any* decision function, whatever criterion they use to select the hypothesis.

Proposition 9.5.4. *If the matrix has some identical rows corresponding to $a_1, a_2, \ldots, a_k$ then for any decision function h_n we have that $P_{h_n} \geq \min_{1\leq i\leq k}\{p(a_i)\}$*

Proof. Assume that $p(a_\ell) = \min_{1\leq i\leq k}\{p(a_i)\}$. We have:

$$
\begin{aligned}
P_{h_n} &= \sum_{a\in\mathcal{A}} p(a)\eta_n(a) \\
&\geq \sum_{1\leq i\leq k} p(a_i)\eta_n(a_i) \\
&\geq \sum_{1\leq i\leq k} p(a_\ell)\eta_n(a_i) && (p(a_\ell) = \min_{1\leq i\leq k}\{p(a_i)\}) \\
&= \sum_{1\leq i\leq k} p(a_\ell) \sum_{h_n(\vec{o})\neq a_i} p(\vec{o}|a_i) \\
&= \sum_{1\leq i\leq k} p(a_\ell) \sum_{h_n(\vec{o})\neq a_i} p(\vec{o}|a_\ell) && (p(\vec{o}|a_i) = p(\vec{o}|a_\ell)) \\
&= p(a_\ell) \sum_{1\leq i\leq k} \sum_{h_n(\vec{o})\neq a_i} p(\vec{o}|a_\ell) \\
&= p(a_\ell) \sum_{1\leq i\leq k} (1 - \sum_{h_n(\vec{o})=a_i} p(\vec{o}|a_\ell)\ ) \\
&\geq (k-1)p(a_\ell) && (\textstyle\sum_{1\leq i\leq k}\sum_{h_n(\vec{o})=a_i} p(\vec{o}|a_\ell) \leq 1)
\end{aligned}
$$

□

Note that the expression $(k-1)p(a_\ell)$ does not depend on n. Assuming that the a_i's have positive probability, from the above proposition we derive that the probability of error is bound from below by a positive constant. Hence the probability of error does not converge to 0.

Corollary 9.5.5. *If there exist $a_1, a_2, \ldots, a_k$ with positive probability, $k \geq 2$, and whose corresponding rows in the matrix are identical, then for any decision function h_n the probability of error is bound from below by a positive constant.*

Remark 9.5.6. *In Proposition 9.5.4 we are allowed to consider any subset of identical rows. In general it is not necessarily the case that a larger subset gives a better bound. In fact, as the subset increases, k increases too, but the minimal $p(a_i)$ may decrease. To find the best bound in general one has to consider all the possible subsets of identical rows.*

Capacity 0 Capacity 0 is the extreme case of identical rows: it corresponds, in fact, to the situation in which all the rows of the matrix are identical. This is, of course, the optimal case with respect to anonymity. All the rows are the same, consequently the observations are of no use for the attacker to infer the anonymous event, i.e. to define the "right" $g_n(\vec{o})$, since all $p_c(\vec{o}|a)$ are maximal.

The probability of error of any decision function is bound from below by $(|\mathcal{A}| - 1) \min ip(a_i)$. Note that by Remark 9.5.6 we may get better bounds by considering subsets of the rows instead than all of them.

Conditional capacity 0 From the point of view of testing the anonymous events we note the following: given a $\vec{o} \in \mathcal{O}^n$, there exists exactly one group r_i of a's such that $p(\vec{o}|a) > 0$, and $p(\vec{o}|a_1) = p(\vec{o}|a_2)$ for all a_1, a_2 in r_i. Hence the attacker knows that the right anonymous event is an a in r_i, but he does not know exactly which one. In other words, the observation gives to the attacker complete knowledge about the group, but tells him nothing about the exact event a in the group, as expected.

For each $r \in \mathcal{R}$ we have that the probability of error is bounded by $(|\mathcal{A}_r| - 1) \min_{i \in r} p(a_i)$.

Probable innocence Concerning the testing of the anonymous events, it is interesting to note that, if the attacker has the possibility of repeating the test with the same input an arbitrary number of times, then probable innocence does not give any guarantee. In fact, Definition 6.2.2 does not prevent the function $p(\vec{o}|\cdot)$ from having a maximum with probability close to 1, for a sufficiently long sequence of observables $\vec{o}$. So the probability of error corresponding to g_n would converge to 0. A similar reasoning can be done for f_n. The only exception is when two (or more) rows a_1, a_2 are equal and correspond to maximals. Imposing this condition for all anonymous actions and all the rows is equivalent to requiring strong anonymity. In conclusion, probable innocence maintains an upper bound on anonymity through protocol repetition only if the system is strongly anonymous. This result is in accordance with the one in Chapter 6.

9.6 Related work

In the field of anonymity and privacy, the idea of using the techniques and concepts of hypothesis testing to reason about the capabilities of an adversary seems to be relatively new. The only other works we are aware of are [Mau00, PHW04, PHW05]. However, those works do not use the setting of hypothesis testing in the information theoretic framework, like we do, so the connection with our work is quite loose.

Part III

Adding Nondeterminism

Ten

The problem of the scheduler

Up to now we have modeled anonymity protocols in a *purely probabilistic* framework, and we described their behavior by assigning probability measures to the observable events. This framework allowed us to fruitfully analyze protocols like the Dining Cryptographers or Crowds and it can be used for a variety of other protocols. However, security protocols often give rise to concurrent and interactive activities that can be best modeled by *nondeterminism*. Examples of such behavior are the order in which messages arrive in a network or resources that are available to a limited number of users but without being able to predict which ones will manage to access them. Such behavior depends on factors that are either too complicated to describe explicitly, or even totally unknown, in both cases they are best modeled by nondeterminism.

Thus it is convenient to specify such protocols using a formalism which is able to represent both *probabilistic* and *nondeterministic* behavior. Formalisms of this kind have been explored in both Automata Theory [Var85, HJ89, YL92, Seg95, SL95] and in Process Algebra [HJ90, BS01, And02, MOW04, PH05, DPP05]. See also [SV04, JLY01] for comparative and more inclusive overviews.

Due to the presence of nondeterminism, in such formalisms it is not possible to define the probability of events in *absolute* terms. We need first to decide how each nondeterministic choice during the execution will be resolved. This decision function is called *scheduler*. Once the scheduler is fixed, the behavior of the system (*relatively* to the given scheduler) becomes fully probabilistic and a probability measure can be defined following standard techniques.

It has been observed by several researchers that in security the notion of scheduler needs to be restricted, or otherwise any secret choice of the protocol could be revealed by making the choice of the scheduler depend on it. This issue was for instance one of the main topics of discussion at the panel of CSFW 2006. We illustrate it here with an example on anonymity. We use the CCS_p calculus, introduced in Section 2.5, where the construct $P +_p Q$ represents a process that evolves into P with probability p and into Q with probability $1-p$.

The system *Sys* consists of a receiver R and two senders S, T communicating via private channels a, b respectively. Which of the two senders is successful is decided probabilistically by R. After reception, R sends a signal *ok*.

$$R \triangleq a.\overline{ok}.0 \ +_{0.5} \ b.\overline{ok}.0$$

$$S \triangleq \bar{a}.0$$
$$T \triangleq \bar{b}.0$$
$$Sys \triangleq (\nu a)(\nu b)(R \mid S \mid T)$$

The signal ok is not private, but since it is the same in both cases, in principle an external observer should not be able to infer from it the identity of the sender (S or T). So the system should be anonymous. However, consider a team of two attackers A and B defined as

$$A \triangleq ok.\bar{s}.0 \qquad\qquad B \triangleq ok.\bar{t}.0$$

and consider the parallel composition $Sys \mid A \mid B$. We have that, under certain schedulers, the system is no longer anonymous. More precisely, a scheduler could leak the identity of the sender via the channels s, t by forcing R to synchronize with A on ok if R has chosen the first alternative, and with B otherwise. This is because in general a scheduler can see the whole history of the computation, in particular the random choices, even those which are supposed to be private. Note that the visibility of the synchronization channels to the scheduler is not crucial for this example: we would have the same problem, for instance, if S, T were both defined as $\bar{a}.0$, R as $a.\overline{ok}.0$, and Sys as $(\nu a)((S +_{0.5} T) \mid R)$.

The above example demonstrates that, with the standard definition of scheduler, it is not possible to represent a truly private random choice (or a truly private nondeterministic choice, for the matter) with the current probabilistic process calculi. This is a clear shortcoming when we want to use these formalisms for the specification and verification of security protocols.

There is another issue related to verification: a private choice has certain algebraic properties that would be useful in proving equivalences between processes. In fact, if the outcome of a choice remains private, then it should not matter at which point of the execution the process makes such choice, until it actually uses it. Consider for instance A and B defined as follows

$$\begin{array}{ll} A \triangleq a(x).([x=0]\overline{ok} & B \triangleq a(x).[x=0]\overline{ok} \\ \quad +_{0.5} & \quad +_{0.5} \\ \quad [x=1]\overline{ok}) & \quad a(x).[x=1]\overline{ok} \end{array}$$

Process A receives a value and then decides randomly whether it will accept the value 0 or 1. Process B does exactly the same thing except that the choice is performed before the reception of the value. If the random choices in A and B are private, intuitively we should have that A and B are equivalent ($A \approx B$). This is because it should not matter whether the choice is done before or after receiving a message, as long as the outcome of the choice is completely invisible to any other process or observer. However, consider the parallel context $C = \overline{a}0 \mid \overline{a}1$. Under any scheduler A has probability at most $1/2$ to perform $\overline{ok}$. With B, on the other hand, the scheduler can choose between $\overline{a}0$ and $\overline{a}1$ based on the outcome of the probabilistic choice, thus making the maximum probability of $\overline{ok}$ equal to 1. The execution trees of $A \mid C$ and $B \mid C$ are shown in Figure 10.1.

In general when $+_p$ represents a private choice we would like to have

$$C[P +_p Q] \approx C[\tau.P] +_p C[\tau.Q] \tag{10.1}$$

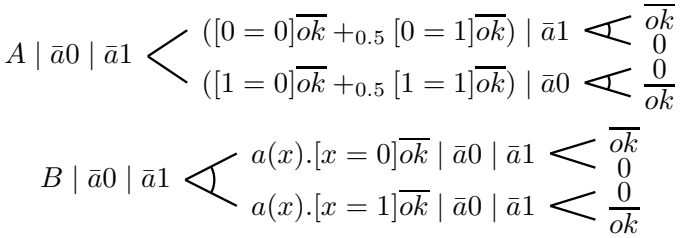


Figure 10.1: Execution trees for $A \mid C$ and $B \mid C$

for all processes P, Q and all contexts C *not containing replication (or recursion)*. In the case of replication the above cannot hold since $!(P +_p Q)$ makes available each time the choice between P and Q, while $(!\tau.P) +_p (!\tau.Q)$ chooses once and for all which of the two (P or Q) should be replicated. Similarly for recursion. The reason why we need a τ is explained in Section 10.3.

The algebraic property (10.1) expresses in an abstract way the privacy of the probabilistic choice. Moreover, this property is also useful for the verification of security properties. In fact, in the next chapter we use this property to prove the correctness of a fair exchange protocol. In principle (10.1) should be useful for any kind of verification in the process algebra style.

We propose a process-algebraic approach to the problem of hiding the outcome of random choices. Our framework is based on the CCS_p calculus, which is an extension of CCS with an internal probabilistic choice construct[1]. This calculus is a variant of the one studied in [DPP05], the main differences being that we use replication instead than recursion, and we lift some restrictions that were imposed in [DPP05] to obtain a complete axiomatization. The semantics of CCS_p is given in terms of Segala's *simple probabilistic automata*, which were introduced in section 2.4.

In order to limit the power of the scheduler, we extend CCS_p with terms representing explicitly the notion of scheduler. The latter interact with the original processes via a labeling system. This will allow to specify at the syntactic level (by a suitable labeling) which choices should be visible to schedulers, and which ones should not.

Contribution The main contributions of this chapter are the following:

- A process calculus CCS_σ in which the scheduler is represented as a process, and whose power can therefore be controlled at the syntactic level.

- The adaptation of the standard notions of probabilistic testing preorders to CCS_σ, and the "sanity check" that they are still precongruences with respect to all the operators except the nondeterministic sum. For the latter we have the problem that P and $\tau.P$ are must equivalent, but $Q + P$ and $Q + \tau.P$ are not. This is typical for the CCS $+$: usually it does not preserve weak equivalences.

[1] We actually consider a variant of CCS where recursion is replaced by replication. The two languages are not equivalent, but we believe that the issues regarding the differences between replication and recursion are orthogonal to the topics investigated in this chapter.

- The proof that, under suitable conditions on the labelings of C, $\tau.P$ and $\tau.Q$, CCS_σ satisfies the property expressed by (10.1), where $\approx$ is probabilistic testing equivalence.

- An application of CCS_σ to an extended anonymity example (the Dining Cryptographers Protocol, DCP). We also briefly outline how to extend CCS_σ so to allow the definition of private nondeterministic choice, and we apply it to the DCP with nondeterministic master. To our knowledge this is the first formal treatment of the scheduling problem in DCP and the first formalization of a nondeterministic master for the (probabilistic) DCP.

Plan of the chapter In the next section we define a preliminary version of the language CCS_σ and of the corresponding notion of scheduler. In Section 10.2 we compare our notion of scheduler with the more standard "semantic" notion, and we improve the definition of CCS_σ so to retrieve the full expressive power of the semantic schedulers. In Section 10.3 we study the probabilistic testing preorders, their compositionality properties, and the conditions under which (10.1) holds. Section 10.4 presents an application to security. Section 10.5 discusses some related work.

10.1 A variant of CCS with explicit scheduler

In this section we present a variant of CCS in which the scheduler is explicit, in the sense that it has a specific syntax and its behavior is defined by the operational semantics of the calculus. We will refer to this calculus as CCS_σ. Processes in CCS_σ contain labels that allow us to refer to a particular subprocess. A scheduler also behaves like a process, using however a different and much simpler syntax, and its purpose is to guide the execution of the main process using the labels that the latter provides. A *complete process* is a process running in parallel with a scheduler, and we will formally describe their interaction by defining an operational semantics for complete processes.

10.1.1 Syntax

Let a range over a countable set of *channel names* and l over a countable set of *atomic labels*. The syntax of CCS_σ, shown in Figure 10.2, is the same as the one of CCS_p except for the presence of labels. These are used to select the subprocess which "performs" a transition. Since only the operators with an initial rule can originate a transition, we only need to assign labels to the prefix and to the probabilistic sum. For reasons explained later, we also put labels on 0, even though this is not required for scheduling transitions. We use labels of the form l^s where l is an atomic label and the index s is a finite string of 0 and 1, possibly empty[2]. Indexes are used to avoid multiple copies of the same label in case of replication, which occurs dynamically due to the bang operator. As explained in the semantics, each time a process is replicated we relabel it using appropriate indexes.

[2]For simplicity we will write l for l^ϵ.

$I ::= 0\,I \mid 1\,I \mid \epsilon$	**label indexes**	$S, T ::=$	**scheduler**
$L ::= l^I$	**labels**	$L.S$	schedule single action
		$\mid (L, L).S$	synchronization
$P, Q ::=$	**processes**	$\mid$ **if** L **then** S **else** S	label test
$L{:}\alpha.P$	prefix		
$\mid P \mid Q$	parallel	$\mid 0$	nil
$\mid P + Q$	nondeterm. choice		
$\mid L{:}\sum_i p_i P_i$	internal prob. choice	$CP ::= P \parallel S$	**complete process**
$\mid (\nu a)P$	restriction		
$\mid\ !P$	replication		
$\mid L{:}0$	nil		

Figure 10.2: The syntax of the core CCS_σ

A scheduler selects a sub-process for execution on the basis of its label, so we use $l.S$ to represent a scheduler that selects the process with label l and continues as S. In the case of synchronization we need to select two processes simultaneously, hence we need a scheduler of the form $(l_1, l_2).S$. Using **if-then-else** the scheduler can test whether a label is available in the process (in the top-level) and act accordingly. A complete process is a process put in parallel with a scheduler, for example $l_1{:}a.l_2{:}b \parallel l_1.l_2$. Note that for processes with an infinite execution path we need schedulers of infinite length. So, to be more formal, we should define schedulers as infinite trees with 3 types of internal nodes, instead of using a BNF grammar.

10.1.2 Semantics

The operational semantics of the CCS_σ-calculus is given in terms of probabilistic automata defined inductively on the basis of the syntax, according to the rules shown in Figure 10.3.

ACT is the basic communication rule. In order for $l : \alpha.P$ to perform α, the scheduler should select this process for execution, so the scheduler needs to be of the form $l.S$. After the execution the complete process will continue as $P \parallel S$. The RES rule models restriction on channel a: communication on this channel is not allowed by the restricted process. Similarly to the Section 2.5, we denote by $(\nu a)\mu$ the measure μ' such that $\mu'((\nu a)P \parallel S) = \mu(P \parallel S)$ for all processes P and $\mu'(R \parallel S) = 0$ if R is not of the form $(\nu a)P$. SUM1 models nondeterministic choice. If $P \parallel S$ can perform a transition to μ, which means that S selects one of the labels of P, then $P + Q \parallel S$ will perform the same transition, i.e. the branch P of the choice will be selected and Q will be discarded. For example

$$l_1{:}a.P + l_2{:}b.Q \parallel l_1.S \xrightarrow{a} \delta(P \parallel S)$$

Note that the operands of the sum do not have labels, the labels belong to the subprocesses of P and Q. In the case of nested choices, the scheduler must go

$$\text{ACT}\quad \frac{}{l{:}\alpha.P \parallel l.S \xrightarrow{\alpha} \delta(P \parallel S)} \qquad \text{RES}\quad \frac{P \parallel S \xrightarrow{\alpha} \mu \quad \alpha \neq a, \overline{a}}{(\nu a)P \parallel S \xrightarrow{\alpha} (\nu a)\mu}$$

$$\text{SUM1}\quad \frac{P \parallel S \xrightarrow{\alpha} \mu}{P + Q \parallel S \xrightarrow{\alpha} \mu} \qquad \text{PAR1}\quad \frac{P \parallel S \xrightarrow{\alpha} \mu}{P \mid Q \parallel S \xrightarrow{\alpha} \mu \mid Q}$$

$$\text{COM}\quad \frac{P \parallel l_1 \xrightarrow{a} \delta(P' \parallel 0) \qquad Q \parallel l_2 \xrightarrow{\overline{a}} \delta(Q' \parallel 0)}{P \mid Q \parallel (l_1, l_2).S \xrightarrow{\tau} \delta(P' \mid Q' \parallel S)}$$

$$\text{PROB}\quad \frac{}{l{:}\sum_i p_i P_i \parallel l.S \xrightarrow{\tau} \sum_i p_i \delta(P_i \parallel S)}$$

$$\text{REP1}\quad \frac{P \parallel S \xrightarrow{\alpha} \mu \quad n = \mathbf{n}(P, \mu) + 1}{!P \parallel S \xrightarrow{\alpha} \rho_{0,n}(\mu) \mid \rho_{1,n}(!P)}$$

$$\text{REP2}\quad \frac{P \parallel l_1 \xrightarrow{a} \delta(P_1 \parallel 0) \qquad P \parallel l_2 \xrightarrow{\overline{a}} \delta(P_2 \parallel 0) \qquad n = \mathbf{n}(P, P_1, P_2) + 1}{!P \parallel (l_1, l_2).S \xrightarrow{\tau} \delta(\rho_{0,n}(P_1) \mid \rho_{10,n}(P_2) \mid \rho_{11,n}(!P) \parallel S)}$$

$$\text{IF1}\quad \frac{l \in tl(P) \qquad P \parallel S_1 \xrightarrow{\alpha} \mu}{P \parallel \textbf{if } l \textbf{ then } S_1 \textbf{ else } S_2 \xrightarrow{\alpha} \mu} \qquad \text{IF2}\quad \frac{l \notin tl(P) \qquad P \parallel S_2 \xrightarrow{\alpha} \mu}{P \parallel \textbf{if } l \textbf{ then } S_1 \textbf{ else } S_2 \xrightarrow{\alpha} \mu}$$

Figure 10.3: The semantics of CCS_σ. SUM1 and PAR1 have corresponding right rules SUM2 and PAR2, omitted for simplicity.

deep and select the label of a prefix, thus resolving all the choices at once.

PAR1 has a similar behavior for the parallel composition. The scheduler selects P to perform a transition on the basis of the label. The difference is that in this case Q is not discarded; it remains in the continuation. $\mu \mid Q$ denotes the measure μ' such that $\mu'(P \mid Q \parallel S) = \mu(P \parallel S)$. COM models synchronization. If $P \parallel l_1$ can perform the action a and $Q \parallel l_2$ can perform $\bar{a}$, then $(l_1, l_2).S$, scheduling both l_1 and l_2 at the same time, can synchronize the two. PROB models internal probabilistic choice. Note that the scheduler cannot affect the outcome of the choice, it can only schedule the choice as a whole (this is why a probabilistic sum has a label) and the process will move to a measure containing all the operands with corresponding probabilities.

REP1 and REP2 model replication. The rules are the same as in CCS_p, with the addition of a re-labeling operator $\rho_{t,n}$. The reason for this is that we want to avoid ending up with multiple copies of the same label as the result of replication, since this would create ambiguities in scheduling as explained in Section 10.1.3. $\rho_{t,n}(P)$ appends t to the index of all labels of P at position n, padding the index with zeros if needed:

$$\begin{aligned}
\rho_{t,n}(l^s{:}\alpha.P) &= l^{s0^m t}{:}\alpha.\rho_{t,n}(P) \\
\rho_{t,n}(l^s{:}\textstyle\sum_i p_i P_i) &= l^{s0^m t}{:}\textstyle\sum_i p_i \rho_{t,n}(P_i) \\
\rho_{t,n}(l^s{:}0) &= l^{s0^m t}{:}0
\end{aligned}$$

$$\text{where } m = n - |s| - 1$$

and homomorphically on the other operators (for instance $\rho_{t,n}(P \mid Q) = \rho_{t,n}(P) \mid \rho_{t,n}(Q)$). We denote by 0^m the string consisting of m zeroes. We also denote by $\rho_{t,n}(\mu)$ the measure μ' such that $\mu'(\rho_{t,n}(P) \parallel S) = \mu(P \parallel S)$.

Note that n must be bigger than the length of all the indexes of P. To ensure this, we define $\mathbf{n}(P_1, \ldots, P_m)$ as the function returning the maximum index length of any label in $P_1, \ldots, P_m$, and similarly for $\mathbf{n}(\mu)$. We use $\mathbf{n}(\cdot)$ in the semantics to select a proper n. Note also that we relabel only the resulting process, not the continuation of the scheduler: there is no need for relabeling the scheduler since we are free to choose the continuation as we please.

Finally **if-then-else** allows the scheduler to adjust its behavior based on the labels that are available in P. $tl(P)$ gives the set of top-level labels of P and is defined as

$$tl(l{:}\alpha.P) = tl(l{:}\textstyle\sum_i p_i P_i) = tl(l{:}0) = \{l\}$$

and as the union of the top-level labels of all sub-processes for the other operators. Then **if** l **then** S_1 **else** S_2 behaves like S_1 if l is available in P and as S_2 otherwise. This is needed when P is the outcome of a probabilistic choice, as discussed in Section 10.2.

10.1.3 Deterministic labelings

The idea in CCS_σ is that a *syntactic* scheduler will be able to completely resolve the nondeterminism of the process, without needing to rely on a *semantic* scheduler at the level of the automaton. This means that the execution of a process in parallel with a scheduler should be fully probabilistic. To achieve this we will impose a condition on the labels that we can use in CCS_σ processes. A *labeling* is an assignment of labels to the prefixes, the probabilistic sums and the 0s of a process. We will require all labelings to be *deterministic* in the following sense.

Definition 10.1.1. *A labeling of a process P is* deterministic *iff for all schedulers S there is only one transition rule $P \parallel S \xrightarrow{\alpha} \mu$ that can be applied and the labelings of all processes P' such that $\mu(P' \parallel S') > 0$ are also deterministic.*

In the general case, it is impossible to decide weather a particular labeling is deterministic. However, there are simple ways to construct labeling that are guaranteed to be deterministic. The most simple family are the linear labelings.

Definition 10.1.2. *A labeling is called* linear *iff for all labels $l_1^{s_1}, l_2^{s_2}$ appearing in the process, $l_1 \neq l_2$ or $s_1 \not\preceq s_2 \wedge s_2 \not\preceq s_1$, where $\preceq$ is the prefix relation on indexes.*

The idea is that in a linear labeling all labels should be pairwise distinct. The extra condition on the indexes forbids having two (distinct) labels l, l^0 since they could become equal as the result of relabeling the first. This is important for the following proposition.

Proposition 10.1.3. *Linear labelings are preserved by transitions.*

Proof. First, notice that the rules only *append* strings to the indexes of the process. That is if $P \xrightarrow{\alpha} \mu$, $\mu(Q) > 0$ and $l^t \in lab(Q)$ then there exists a label $l^s \in P$ such that $s \preceq t$. This is clear since the only relabeling operator $\rho_{t,n}$ only appends strings to indexes.

We will write $s \nsim t$ for $s \not\preceq t \wedge t \not\preceq s$. First, we notice that $s \nsim t$ iff $s_i \neq t_i$ for some $i \leq \max\{|s|, |t|\}$ where s_i, t_i denote the i-th character of s, t respectively. As a consequence we have that

$$s \nsim t \;\Rightarrow\; ss' \nsim tt' \qquad \text{for all } s', t' \tag{10.2}$$

since $ss' \nsim tt'$ still differ at the i-th character.

The proof is by induction of the "proof tree" of the transition. The base cases (rules ACT, PROB) are easy since the labels of the resulting process are a subset of the original ones. For the inductive case, the rules RES, SUM1/2, IF1, IF2 are easy since the resulting measure μ is the same as in the premise, so a direct application of the induction hypothesis suffices. Now consider the PAR1 rule

$$\frac{P \parallel S \xrightarrow{\alpha} \mu}{P \mid Q \parallel S \xrightarrow{\alpha} \mu \mid Q}$$

Assume that $P \mid Q$ has a linear labeling and consider a process P' such that $\mu(P') > 0$. We want to show that $P'|Q$ has a linear labeling, that is if two labels of $P'|Q$ have the same base then their indexes must be prefix-incomparable. Since Q has a linear labeling and so does P' (from the induction hypothesis), we only need to compare indexes between P' and Q. Let $l^s \in lab(P'), l^t \in lab(Q)$. Since P' comes from a transition of P then there exists $l^{s'} \in lab(P)$ such that $s' \preceq s$, and since $P \mid Q$ has a linear labeling then $s' \nsim t$. So from (10.2) we have $s \nsim t$.

Then consider the REP1 rule

$$\frac{P \parallel S \xrightarrow{\alpha} \mu \quad n = \mathbf{n}(P, \mu) + 1}{!P \parallel S \xrightarrow{\alpha} \rho_{0,n}(\mu) \mid \rho_{1,n}(!P)}$$

Let P' be a process such that $\mu(P') > 0$. Again we only need to compare indexes between $\rho_{0,n}(P')$ and $\rho_{1,n}(!P)$. Let $l^s \in lab(\rho_{0,n}(P'))$ and $l^t \in lab(\rho_{1,n}(!P))$. By construction s has 0 in the n-th position, while t has 1, so $s \nsim t$.

Finally, consider the REP2 rule

$$\frac{P \parallel l_1 \xrightarrow{a} \delta(P_1 \parallel 0) \qquad P \parallel l_2 \xrightarrow{\overline{a}} \delta(P_2 \parallel 0) \qquad n = \mathbf{n}(P, P_1, P_2) + 1}{!P \parallel (l_1, l_2).S \xrightarrow{\tau} \delta(\rho_{0,n}(P_1) \mid \rho_{10,n}(P_2) \mid \rho_{11,n}(!P) \parallel S)}$$

Let $l^{s_1} \in lab(\rho_{0,n}(P_1))$, $l^{s_2} \in lab(\rho_{10,n}(P_2))$ and $l^t \in lab(\rho_{11,n}(!P))$. Again, by construction, s_1 has 0 in the n-th position while s_2, t have 1, and s_2 has 0 in the $(n+1)$-th position while t has 1. So $s_1 \nsim s_2, s_1 \nsim t$ and $s_2 \nsim t$. □

Proposition 10.1.4. *A linear labeling is deterministic.*

Proof. Let P be a process with a linear labeling and let S be a scheduler. We want to show that there is only one transition $P \parallel S \xrightarrow{\alpha} \mu$ enabled. In a linear labeling, all labels are pairwise distinct, so the label(s) in the root of S appear

at most once in P. So from the rules PAR1/PAR2, at most one is applicable, since at most one branch of $P \mid Q$ contains the required label. The same holds for SUM1/SUM2.

We want to show that we can construct at most one proof tree for the transition of $P \parallel S$. Since we eliminated one rule of the pairs PAR1/2, SUM1/2, for the remaining rules and for a fixed "type" of process and scheduler, there is at most one rule applicable. For example for $P \mid Q$ and $l.S$ only PAR is applicable, for $P \mid Q$ and $(l_1, l_2).S$ only COM is applicable, for $!P$ and $l.S$ only REP1 and so on. And since the premises of all rules involve a simpler process or a simpler scheduler, the result comes easily by induction on the structure of $P \parallel S$.

The proof that all processes enabled by μ have also deterministic labelings comes from the fact that linear labelings are preserved by transitions. □

There are labelings that are deterministic without being linear. In fact, such labelings will be the means by which we hide information from the scheduler. However, the property of being deterministic is crucial since it implies that the scheduler will resolve all the nondeterminism of the process.

Proposition 10.1.5. *Let P be a CCS$_\sigma$ process with a deterministic labeling. Then for all schedulers S, the automaton produced by $P \parallel S$ is fully probabilistic.*

Proof. Direct application of the definition of deterministic labeling. □

10.2 Expressiveness of the syntactic scheduler

CCS$_\sigma$ with deterministic labelings allows us to separate probabilities from nondeterminism in a straightforward way: a process in parallel with a scheduler behaves in a fully probabilistic way and the nondeterminism arises from the fact that we can have many different schedulers. We may now ask the question: how powerful are the syntactic schedulers wrt the semantic ones, i.e. those defined directly over the automaton?

Let P be a CCS$_p$ process and P_σ be the CCS$_\sigma$ process obtained from P by applying a linear labeling. We denote this relation by $P \equiv_l P_\sigma$. We say that the semantic scheduler ζ of P is equivalent to the syntactic scheduler S of P_σ, written $\zeta \sim_P S$, iff the automata $etree(P, \zeta)$ and $P_\sigma \parallel S$ are probabilistically bisimilar.

A scheduler S is *non-blocking* for a process P if it always schedules some transitions, except when P itself is blocked. Let $Sem(P)$ be the set of the semantic schedulers for the process P and $Syn(P_\sigma)$ be the set of the non-blocking syntactic schedulers for process P_σ. Then we can show that for all semantic schedulers of P we can create a equivalent syntactic one for P_σ.

Proposition 10.2.1. *Let P be a CCS process and let P_σ be a CCS$_\sigma$ process obtained by adding a linear labeling to P. Then $\forall \zeta \in Sem(P)\ \exists S \in Syn(P_\sigma)$: $\zeta \sim_P S$.*

Proof. Let P be a CCS$_p$ process and let $M = (S, P, A, \mathcal{D})$ be the corresponding automaton. An execution of M is a sequence $\varphi = P\alpha_1 P_1 \ldots \alpha_n P_n$ such that $P_{i-1} \xrightarrow{\alpha_i} \mu$ and $\mu(P_i) > 0$. Let $\zeta : exec^*(M) \to \mathcal{D}$ be a scheduler for M.

$etree(\zeta, M)$ is a fully probabilistic automaton having as states the executions of M and where $\varphi \xrightarrow{\alpha} \mu'$ iff $\zeta(\varphi) = (P_n, \alpha, \mu)$ and $\mu'(\varphi\alpha P_{n+1}) = \mu(P_{n+1})$.

Let P_σ be a CCS_σ process such that $P \equiv_l P_\sigma$. To simplify the notation we will use Q for CCS_σ processes, so let $Q = P_\sigma$.

First note that for each rule in the semantics of CCS_p there is a corresponding rule for CCS_σ with the only addition being the syntactic scheduler and the labels of the resulting process. Thus, we can show that

$$P \xrightarrow{\alpha} \mu \wedge P \equiv_l Q \Rightarrow \quad \begin{array}{l} \exists S : Q \parallel S \xrightarrow{\alpha} \mu' \text{ and } \forall 1 \leq i \leq n : \\ \mu(P_i) = \mu'(Q_i \parallel S_c) \text{ with } P_i \equiv_l Q_i \end{array}$$

where $\{P_1, \ldots, P_n\}$ is the support of μ. If $t = (P, \alpha, \mu) \in \mathcal{D}$ (the tuple describing the transition of P) then let $sched(t, Q)$ denote the head of the scheduler S above. For example if $t = (a.P', a, P')$ and $Q = l{:}a.Q'$ then $sched(t, Q) = l$.

We construct the syntactic scheduler for a process Q corresponding to the semantic scheduler ζ, at state φ with $lstate(\varphi) \equiv_l Q$, as follows

$$\begin{array}{l} S(\zeta, \varphi, Q) \triangleq sched(\zeta(\varphi), Q). \\ \qquad \textbf{if } lm(Q_1) \textbf{ then } S(\zeta, \varphi\alpha P_1, Q_1) \textbf{ else} \\ \qquad \ldots \\ \qquad \textbf{if } lm(Q_{n-1}) \textbf{ then } S(\zeta, \varphi\alpha P_{n-1}, Q_{n-1}) \textbf{ else} \\ \qquad S(\zeta, \varphi\alpha P_n, Q_n) \end{array} \tag{10.3}$$

where $\zeta(\varphi) = (P, \alpha, \mu)$, $\{P_1, \ldots, P_n\}$ is the support of μ and $Q_1, \ldots, Q_n$ are the corresponding processes in the support of μ' in the transition $Q \parallel S \xrightarrow{a} \mu'$ with $S = sched(\zeta(\varphi), Q)$. Such a transition always exists, as explained in the previous paragraph. $lm(Q)$ returns the left-most label appearing in Q, note that all processes contain at least one label since they contain at least one 0.

Now let $\zeta \in Syn(P)$, $\varphi_0 = P$ (empty execution) and $S = S(\zeta, \varphi_0, Q)$. We compare the automata $etree(P, \zeta)$ and $Q \parallel S$ and we show that they are bisimilar by creating a bisimulation relation that relates their starting states φ_0 and $Q \parallel S$. First we define an equivalence $\equiv_Q$ on schedulers as

$$S \equiv_Q S' \quad \text{iff} \quad Q \parallel S \xrightarrow{\alpha} \mu \;\Leftrightarrow\; Q \parallel S' \xrightarrow{\alpha} \mu$$

Intuitively $S \equiv_Q S'$ iff they have the same effect on the process Q, for example if S' is an **if-then-else** construct that enables S. We now define a relation $\mathcal{R} \subseteq states(etree(P, \zeta)) \cup states(Q \parallel S)$ as follows

$$\varphi \;\mathcal{R}\; (Q \parallel S) \quad \text{iff} \quad lstate(\varphi) \equiv_l Q \;\text{ and }\; S \equiv_Q S(\zeta, \varphi, Q)$$

and we show that $\mathcal{R}$ is a strong bisimulation. Suppose that $\varphi \;\mathcal{R}\; (Q \parallel S)$ and $\varphi \xrightarrow{\alpha} \mu$. Let $\{P_1, \ldots, P_n\}$ be the support of μ. Since $S \equiv_Q S(\zeta, \varphi, Q)$ then (by construction of $S(\zeta, \varphi, Q)$) there exists a transition $Q \parallel S \xrightarrow{\alpha} \mu'$ where $\mu'(Q_i \parallel S_c) = \mu(P_i)$ and $P_i \equiv_l Q_i$ for $1 \leq i \leq n$. The scheduler S_c above, common for all Q_i's, is the **if-then-else** construct of (10.3), containing all $S(\zeta, \varphi\alpha P_i, Q_i)$'s, each guarded by **if** $lm(Q_i)$. Since the label of Q is linear then all labels are pairwise distinct, so the Q_i's have disjoint labels that is $lm(Q_i)$ cannot appear in $lb(Q_j)$ for $i \neq j$. This means that $S_c \equiv_{Q_i} S(\zeta, \varphi\alpha P_i, Q_i)$ since

only the i-th branch of S_c can be enabled by Q_i. Thus we have $P_i \mathcal{R} (Q_i \parallel S_c)$, for all $1 \leq i \leq n$, which implies that $\mu \mathcal{R} \mu'$.

Similarly for the case where $Q \parallel S \xrightarrow{\alpha} \mu$. By definition of $S(\zeta, \varphi, Q)$ there exists a transition $P \xrightarrow{\alpha} \mu'$ where $\mu'(P_i) = \mu(Q_i \parallel S_c)$ and $P_i \equiv_l Q_i$ for $1 \leq i \leq n$. So again $P_i \mathcal{R} (Q_i \parallel S_c)$, for all $1 \leq i \leq n$, thus $\mu \mathcal{R} \mu'$. □

To obtain this result the label test (**if-then-else**) is crucial, in the case P performs a probabilistic choice. The scheduler uses the test to find out the result of the probabilistic choice and adapt its behavior accordingly (as the semantic scheduler is allowed to do). For example let $P = l : (l_1 : a +_p l_2 : b) \mid (l_3 : c + l_4 : d)$. For this process, the scheduler $l.(\textbf{if } l_1 \textbf{ then } l_3.l_1 \textbf{ else } l_4.l_2)$ first performs the probabilistic choice. If the result is $l_1 : a$ it performs c, a, otherwise it performs d, b. This is also the reason we need labels for 0, in case it is one of the operands of the probabilistic choice.

One would expect to obtain also the inverse of Proposition 10.2.1, showing the same expressive power for the two kinds of schedulers. We believe that this is indeed true, but it is technically more difficult to state. The reason is that the simple translation we did from CCS_p processes to CCS_σ, namely adding a linear labeling, might introduce choices that are not present in the original process. For example let $P = (a +_p a) \mid (c + d)$ and $P_\sigma = l : (l_1 : a +_p l_2 : a) \mid (l_3 : c + l_4 : d)$. In P the choice $a +_p a$ is not a real choice, it can only do an τ transition and go to a with probability 1. But in P_σ we make the two outcomes distinct due to the labeling. So the syntactic scheduler $l.(\textbf{if } l_1 \textbf{ then } l_3.l_1 \textbf{ else } l_4.l_2)$ has no semantic counterpart simply because P_σ has more choices that P, but this is an artifact of the translation. A more precise translation that would establish the exact equivalence of schedulers is left as future work.

10.2.1 Using non-linear labelings

Up to now we are using only linear labelings which, as we saw, give us the whole power of semantic schedulers. However, we can construct non-linear labelings that are still deterministic, that is there is still only one transition possible at any time even though we have multiple occurrences of the same label. There are various cases of useful non-linear labelings.

Proposition 10.2.2. *Let P,Q be CCS_σ processes with deterministic labelings (not necessarily disjoint). The following labelings are all deterministic:*

$$l : (P +_p Q) \tag{10.4}$$

$$l_1 : a.P + l_2 : b.Q \tag{10.5}$$

$$(\nu a)(\nu b)(l_1 : a.P + l_1 : b.Q \mid l_2 : \bar{a}) \tag{10.6}$$

Proof. Processes (10.4),(10.6) have only one transition enabled, while (10.5) has two, all enabled by exactly one scheduler. After any of these transitions, only one of P, Q remains. □

Consider the case where P and Q in the above proposition share the same labels. In (10.4) the scheduler cannot select an action inside P, Q, it must select the choice itself. After the choice, only one of P, Q will be available so there will be no ambiguity in selecting transitions. The case (10.5) is similar but

with nondeterministic choice. Now the guarding prefixes must have different labels, since the scheduler should be able to resolve the choice, however after the choice only one of P, Q will be available. Hence, again, the multiple copies of the labels do not constitute a problem. In (10.6) we allow the same label on the guarding prefixes of a nondeterministic choice. This is because the guarding channels a, b are restricted and only one of the corresponding output actions is available ($\bar{a}$). As a consequence, there is no ambiguity in selecting transitions. A scheduler (l_1, l_2) can only perform a synchronization on a, even though l_1 appears twice.

However, using multiple copies of a label limits the power of the scheduler, since the labels provide information about the outcome of a probabilistic choice (and allow the scheduler to choose different strategies through the use of the scheduler choice). In fact, this is exactly the technique we will use to achieve the goals described in the beginning of this chapter. Consider for example the process:

$$l{:}(l_1{:}\bar{a}.R_1 +_p l_1{:}\bar{a}.R_2) \mid l_2{:}a.P \mid l_3{:}a.Q \tag{10.7}$$

From Proposition 10.2.2(10.4) this labeling is deterministic. However, since both branches of the probabilistic sum have the same label l_1, the scheduler cannot resolve the choice between P and Q based on the outcome of the choice. There is still nondeterminism: the scheduler $l.(l_1, l_2)$ will select P and the scheduler $l.(l_1, l_3)$ will select Q. However this selection will be independent from the outcome of the probabilistic choice.

Note that we did not impose any direct restrictions on the schedulers, we still consider all possible syntactic schedulers for the process (10.7) above. However, having the same label twice limits the power of the syntactic schedulers with respect to the semantic ones. This approach has the advantage that the restrictions are limited to the choices with the same label. We already know that having pairwise distinct labels gives the full power of the semantic scheduler. So the restriction is local to the place where we, intentionally, put the same labels.

10.3 Testing relations for CCS$_\sigma$ processes

Testing relations ([NH84]) are a method of comparing processes by considering their interaction with the environment. A *test* is a process running in parallel with the one being tested and which can perform a distinguished action ω that represents success. Two processes are testing equivalent if they can pass the same tests. This idea is very useful for the analysis of security protocols, as suggested in [AG99], since a test can be seen as an adversary who interferes with a communication agent and declares ω if an attack is successful. Then two processes are testing equivalent if they are vulnerable to the same attacks.

In the probabilistic setting we take the approach of [JLY01] which considers the exact probability of passing a test (in contrast to [PH05] which considers only the ability to pass a test with probability non-zero (may-testing) or one (must-testing)). This approach leads to the definition of two preorders $\sqsubseteq_{\mathbf{may}}$ and $\sqsubseteq_{\mathbf{must}}$. $P \sqsubseteq_{\mathbf{may}} Q$ means that if P can pass O then Q can also pass O with the same probability. $P \sqsubseteq_{\mathbf{must}} Q$ means that if P always passes O with at least some probability then Q always passes O with at least the same probability.

A labeling of a process is *fresh* (with respect to a set $\mathcal{P}$ of processes) if its labels do not appear in any other process in $\mathcal{P}$ (note that it is not required to be linear). A test O is a CCS_σ process with a fresh labeling (wrt all tested processes), containing the distinguished action ω. Let $Test_{\mathcal{P}}$ denote the set of all tests with respect to $\mathcal{P}$ and let $(\nu)P$ denote the restriction on all channels of P, thus allowing only τ actions. We define $p_\omega(P, S, O)$ to be the probability of the set of executions of the fully probabilistic automaton $(\nu)(P \mid O) \parallel S$ that contain ω. Note that this set can be produced as a countable union of disjoint cones so its probability is well-defined.

Definition 10.3.1. *Let P, Q be CCS_σ processes. We define must and may testing preorders as follows:*

$$P \sqsubseteq_{\boldsymbol{may}} Q \quad \textit{iff} \quad \forall O\ \forall S_P\ \exists S_Q\ :\ p_\omega(P, S_P, O) \leq p_\omega(Q, S_Q, O)$$
$$P \sqsubseteq_{\boldsymbol{must}} Q \quad \textit{iff} \quad \forall O\ \forall S_Q\ \exists S_P\ :\ p_\omega(P, S_P, O) \leq p_\omega(Q, S_Q, O)$$

where O ranges over $Test_{P,Q}$ and S_X ranges over $Syn((\nu)(X \mid O))$.

We also define $\approx_{\mathbf{may}}, \approx_{\mathbf{must}}$ to be the equivalences induced by $\sqsubseteq_{\mathbf{may}}, \sqsubseteq_{\mathbf{must}}$ respectively.

A context C is a process with a hole. A preorder $\sqsubseteq$ is a precongruence if $P \sqsubseteq Q$ implies $C[P] \sqsubseteq C[Q]$ for all contexts C. May and must testing are precongruences if we restrict to contexts with linear and fresh labelings and without occurrences of $+$. This result is essentially an adaptation to our framework of the analogous precongruence property in [YL92].

Proposition 10.3.2. *Let P, Q be CCS_σ processes such that $P \sqsubseteq_{\boldsymbol{may}} Q$ and let C be a context with a linear and fresh labeling (wrt P, Q) and in which $+$ does not occur. Then $C[P] \sqsubseteq_{\boldsymbol{may}} C[Q]$. Similarly for $\sqsubseteq_{\boldsymbol{must}}$.*

Proof. Without loss of generality we assume that tests do not perform internal actions, but only synchronizations with the tested process. The proof will be by induction on the structure of C. Let O range over tests with fresh labelings, let S_P range over $Syn((\nu)(C[P] \mid O))$ and S_Q range over $Syn((\nu)(C[Q] \mid O))$. The induction hypothesis is:

$$\textbf{may})\ \forall O\ \forall S_P\ \exists S_Q\ :\ p_\omega(C[P], S_P, O) \leq p_\omega(C[Q], S_Q, O) \quad \text{and}$$
$$\textbf{must})\ \forall O\ \forall S_Q\ \exists S_P\ :\ p_\omega(C[P], S_P, O) \leq p_\omega(C[Q], S_Q, O)$$

We have the following cases for C:

- Case $C = []$. Trivial.
- Case $C = l_1{:}a.C'$
  The scheduler S_P has to be of the form $S_P = (l_1, l_2).S'_P$ where l_2 is the label of a $\overline{a}$ prefix in O (if no such prefix exists then the case is trivial).

  A scheduler of the form $(l_1, l_2).S$ can schedule any process of the form $l_1{:}a.X$ (with label l_1) giving the transition:

$$(\nu)(l_1{:}a.X \mid O) \parallel (l_1, l_2).S \xrightarrow{\tau} \delta((\nu)(X \mid O') \parallel S)$$

and producing always the same O'. The probability p_ω will be

$$p_\omega(l_1{:}a.X, (l_1, l_2).S, O) = p_\omega(X, S, O') \tag{10.8}$$

Thus for (**may**) we have

$$\begin{array}{rcll} p_\omega(C[P], (l_1, l_2).S'_P, O) & = & p_\omega(C'[P], S'_P, O') & (10.8) \\ & \leq & p_\omega(C'[Q], S'_Q, O') & \text{Ind. Hyp.} \\ & = & p_\omega(C[Q], (l_1, l_2).S'_Q, O) & (10.8) \\ & = & p_\omega(C[Q], S_Q, O) & \end{array}$$

For (**must**) we can perform the above derivation in the opposite direction, given that a scheduler for $C[Q]$ must be of the form $S_Q = (l_1, l_2).S'_Q$.

- Case $C = C' \mid R$
  Since we only consider contexts with linear and fresh labeling, the labeling of $R \mid O$ is fresh wrt $C'[]$, so $R \mid O$ is itself a test, and

$$p_\omega(X \mid R, S, O) = p_\omega(X, S, R \mid O) \tag{10.9}$$

  Thus for (**may**) we have

$$\begin{array}{rcll} p_\omega(C[P], S_P, O) & = & p_\omega(C'[P], S_P, R \mid O) & (10.9) \\ & \leq & p_\omega(C'[Q], S_Q, R \mid O) & \text{Ind. Hyp.} \\ & = & p_\omega(C[Q], S_Q, O) & (10.9) \end{array}$$

  For (**must**) we can perform the above derivation in the opposite direction.

- Case $C = l_1{:}(C' +_p R)$
  Since we consider only contexts with linear and fresh labelings, the labels of C' are disjoint from those of R. Thus, in order to be non-blocking, the scheduler of a process of the form $l_1{:}(C'[P] +_p R)$ must detect the outcome of the probabilistic choice and continue as S_C if the outcome is $C'[P]$ or as S_R otherwise. For example S_P could be $l_1.\textbf{if } l \textbf{ then } S_C \textbf{ else } S_R$ or a more complicated **if-then-else**. So we have

$$\begin{array}{l} p_\omega(l_1{:}(C'[P] +_p R), S, O) = p\ p_\omega(C'[P], S_C, O) + \\ \qquad\qquad\qquad\qquad \bar{p}\ p_\omega(R, S_R, O) \end{array} \tag{10.10}$$

  where $\bar{p} = 1 - p$. For (**may**) we have

$$\begin{array}{rcll} & & p_\omega(l_1{:}(C'[P] +_p R), S_P, O) & \\ & = & p\ p_\omega(C'[P], S_C, O) + \bar{p}\ p_\omega(R, S_R, O) & (10.10) \\ & \leq & p\ p_\omega(C'[Q]), S'_C, O) + \bar{p}\ p_\omega(R, S_R, O) & \text{Ind. Hyp.} \\ & = & p_\omega(l_1{:}(C'[Q] +_p R), l_1.(\textbf{if } l \textbf{ then } S'_C \textbf{ else } S_R), O) & \\ & = & p_\omega(C[Q], S_Q, O) & \end{array}$$

Where $l \in tl(C'[Q])$ (and thus $l \notin tl(R)$). We used the **if-then-else** in S_Q to imitate the test of S_P.

For (**must**) we can perform the above derivation in the opposite direction.

- Case $C = (\nu a)C'$
  The process $(\nu)((\nu a)C'[X] \mid O)$ has the same transitions as $(\nu)(C'[X] \mid (\nu a)O)$. The result follows by the induction hypothesis.

- Case $C = !C'$.
  may) We will first prove that for all $m \geq 1$:

$$\forall O\ \forall S_P\ \exists S_Q\ :\ p_\omega(C'[P]^m, S_P, O) \leq p_\omega(C'[Q]^m, S_Q, O) \qquad (10.11)$$

  $C'[P]^m$ is defined as $C'[P]^1 = \rho_{0,n}(C'[P])$ and

$$C'[P]^m = C'[P]^{m-1} \mid \rho_{1^{m-1}0,n}(C'[P]) \quad m > 1$$

  where $n = \mathbf{n}(C'[P]) + 1$. Intuitively, $C'[P]^m$ is the m-times unfolding of $!C'[P]$, taking into account the relabeling that takes place each time a new process is spawned. The proof is by induction on m. The base case $m = 1$ is trivial. Assuming that it holds for $m-1$, the argument is similar to the case of parallel context. Let $R = \rho_{1^{m-1}0,n}(C'[P])$, so $C'[P]^m = C'[P]^{m-1} \mid R$ and since all labels in R are relabeled to make them disjoint from those of $C'[P]^{m-1}$, we have that $R \mid O$ has a fresh labeling so it is itself a test. Thus

$$\begin{aligned} p_\omega(C'[P]^m, S_P, O) &= p_\omega(C'[P]^{m-1}, S_P, R \mid O) && (10.9) \\ &\leq p_\omega(C'[Q]^{m-1}, S_Q, R \mid O) && \text{Ind. Hyp.} \\ &= p_\omega(C[Q]^m, S_Q, O) && (10.9) \end{aligned}$$

  So (10.11) holds. Now assume that the negation of the induction hypothesis holds, that is

$$\exists O \exists S_P\ \forall S_Q\ :\ p_\omega(!C'[P], S_P, O) > p_\omega(!C'[Q], S_Q, O)$$

  There can be executions containing ω of arbitrary length, however their probability will go to zero as the length increases. Thus there will be an m such that if we consider only executions of length at most m then the above inequality will still hold. But these executions can be simulated by $C'[P]^m, C'[Q]^m$ which is impossible by (10.11).

  Similarly for (**must**).

□

This also implies that $\approx_{\mathbf{may}}, \approx_{\mathbf{must}}$ are congruences. Note that P, Q in the above proposition are not required to have linear labelings, P might include multiple occurrences of the same label thus limiting the power of the schedulers S_P. This shows the locality of the scheduler's restriction: some choices inside P are hidden from the scheduler but the rest of the context is fully visible.

If we remove the freshness condition of the context then Proposition 10.3.2 is no longer true. Let $P = l_1{:}a.l_2{:}b$, $Q = l_3{:}a.l_4{:}b$ and $C = l{:}(l_1{:}a.l_2{:}c +_p [\,])$. We have $P \approx_{\mathbf{may}} Q$ but $C[P], C[Q]$ can be separated by the test $O = \bar{a}.\bar{b}.\omega \mid \bar{a}.\bar{c}.\omega$ (when the labeling is omitted assume a linear one). It is easy to see that $C[Q]$ can pass the test with probability 1 by selecting the correct branch of O based on the outcome of the probabilistic choice. In $C[P]$ this is not possible because of the labels l_1, l_2 that are common in P, C. On the other hand, it is not clear if the linearity of the context's labeling is indispensable for the above Proposition, the condition is needed for the proof but yet we haven't found any counter-examples.

We can now state the result that we announced in the beginning of the chapter.

Theorem 10.3.3. *Let P, Q be CCS_σ processes and C a context with a linear and fresh labeling and without occurrences of bang. Then*

$$\begin{array}{lcll} l{:}(C[l_1{:}\tau.P] +_p C[l_1{:}\tau.Q]) & \approx_{\boldsymbol{may}} & C[l{:}(P +_p Q)] & \textit{and} \\ l{:}(C[l_1{:}\tau.P] +_p C[l_1{:}\tau.Q]) & \approx_{\boldsymbol{must}} & C[l{:}(P +_p Q)] & \end{array}$$

Proof. Since we will always use the label l for all probabilistic sum $+_p$, and l_0 for $\tau.P$ and $\tau.Q$, we will omit these labels to make the proof more readable. We will also denote $(1-p)$ by $\bar{p}$.

Let $R_1 = C[\tau.P] +_p C[\tau.Q]$ and $R_2 = C[P +_p Q]$. We will prove that for all tests O and for all schedulers $S_1 \in Syn((\nu)(R_1 \mid O))$ there exists $S_2 \in Syn((\nu)(R_2 \mid O))$ such that $p_\omega(R_1, S_1, O) = p_\omega(R_2, S_2, O)$ and vice versa. This implies both $R_1 \approx_{\mathbf{may}} R_2$ and $R_1 \approx_{\mathbf{must}} R_2$.

Without loss of generality we assume that tests do not perform internal actions, but only synchronizations with the tested process. First, it is easy to see that

$$p_\omega(P +_p Q, l.S, O) = p\, p_\omega(P, S, O) + \bar{p}\, p_\omega(Q, S, O) \qquad (10.12)$$

$$p_\omega(l_1{:}a.P, (l_1, l_2).S, O) = p_\omega(P, S, O') \qquad (10.13)$$

where $(\nu)(l_1{:}a.P \mid O) \parallel (l_1, l_2).S \xrightarrow{\tau} \delta((\nu)(P \mid O' \parallel S))$.

In order for the scheduler of R_1 to be non-blocking, it has to be of the form $l.S_1$, since the only possible transition of R_1 is the probabilistic choice labeled by l. By (10.12) we have

$$p_\omega(C[\tau.P] + C[\tau.Q], l.S_1, O) = p\, p_\omega(C[\tau.P], S_1, O) + \bar{p}\, p_\omega(C[\tau.Q], S_1, O)$$

The proof will be by induction on the structure of C. Let O range over tests with fresh labelings, let S_1 range over non-blocking schedulers for both $C[\tau.P]$ and $C[\tau.Q]$ (such that $l.S_1$ is a non-blocking scheduler for R_1) and let S_2 range over non-blocking schedulers for R_2. The induction hypothesis is:

$\Rightarrow$) $\forall O\ \forall S_1\ \exists S_2$:

$$p\, p_\omega(C[\tau.P], S_1, O) + \bar{p}\, p_\omega(C[\tau.Q], S_1, O) = p_\omega(C[P +_p Q], S_2, O) \quad \text{and}$$

$\Leftarrow$) $\forall O\ \forall S_2\ \exists S_1$:

$$p\, p_\omega(C[\tau.P], S_1, O) + \bar{p}\, p_\omega(C[\tau.Q], S_1, O) = p_\omega(C[P +_p Q], S_2, O)$$

We have the following cases for C:

- Case $C = []$. Trivial.

- Case $C = l_1{:}a.C'$
  The scheduler S_1 of $C[\tau.P]$ and $C[\tau.Q]$ has to be of the form $S_1 = (l_1, l_2).S_1'$ where l_2 is the label of a $\overline{a}$ prefix in O (if no such prefix exists then the case is trivial).

  A scheduler of the form $(l_1, l_2).S$ can schedule any process of the form $l_1{:}a.X$ (with label l_1) giving the transition:

  $$(\nu)(l_1{:}a.X \mid O) \parallel (l_1, l_2).S \xrightarrow{\tau} \delta((\nu)(X \mid O') \parallel S)$$

  and producing always the same O'. The probability p_ω for these processes will be given by equation (10.13).

  Thus for ($\Rightarrow$) we have

  $$\begin{aligned}
  & p\ p_\omega(l_1{:}a.C[\tau.P], (l_1, l_2).S_1', O) + \bar{p}\ p_\omega(l_1{:}a.C[\tau.Q], (l_1, l_2).S_1', O) \\
  &= p\ p_\omega(C'[\tau.P], S_1', O') + \bar{p}\ p_\omega(C'[\tau.Q], S_1', O') && (10.13) \\
  &= p_\omega(C'[P +_p Q], S_2', O') && \text{Ind. Hyp.} \\
  &= p_\omega(l_1{:}a.C'[P +_p Q], (l_1, l_2).S_2', O) && (10.13) \\
  &= p_\omega(R_2, S_2, O)
  \end{aligned}$$

  For ($\Leftarrow$) we can perform the above derivation in the opposite direction, given that a scheduler for $R_2 = l_1 : a.C'[P +_p Q]$ must be of the form $S_2 = (l_1, l_2).S_2'$.

- Case $C = C' \mid R$
  Since we only consider contexts with linear and fresh labelings, the labeling of $R \mid O$ is fresh so it is itself a test, and

  $$p_\omega(X \mid R, S, O) = p_\omega(X, S, R \mid O) \qquad (10.14)$$

  Thus for ($\Rightarrow$) we have

  $$\begin{aligned}
  & p\ p_\omega(C'[\tau.P] \mid R, S_1, O) + \bar{p}\ p_\omega(C'[\tau.Q] \mid R, S_1, O) \\
  &= p\ p_\omega(C'[\tau.P], S_1, R \mid O) + \bar{p}\ p_\omega(C'[\tau.Q], S_1, R \mid O) && (10.14) \\
  &= p_\omega(C'[P +_p Q], S_2, R \mid O) && \text{Ind. Hyp.} \\
  &= p_\omega(C'[P +_p Q] \mid R, S_2, O) && (10.14) \\
  &= p_\omega(R_2, S_2, O)
  \end{aligned}$$

  For ($\Leftarrow$) we can perform the above derivation in the opposite direction.

- Case $C = l_1{:}(C' +_q R)$
  Since we consider only contexts with linear and fresh labelings, the labels of C' are disjoint from those of R, thus the scheduler of a process of the form $l_1{:}(C'[X] +_q R)$ must be of the form $S = l_1.(\textbf{if } l_C \textbf{ then } S_C \textbf{ else } S_R)$

where $l_C \in tl(C'[X])$, S_C is a scheduler containing labels of $C'[X]$ and S_R is a scheduler containing labels of R. Moreover

$$\begin{aligned} & p_\omega(l_1{:}(C'[X] +_q R), S, O) \\ = \;& q\; p_\omega(C'[X], \mathbf{if}\ l_C\ \mathbf{then}\ S_C\ \mathbf{else}\ S_R, O) + \\ & \bar{q}\; p_\omega(R, \mathbf{if}\ l_C\ \mathbf{then}\ S_C\ \mathbf{else}\ S_R, O) \\ = \;& q\; p_\omega(C'[X], S_C, O) + \bar{q}\; p_\omega(R, S_R, O) \end{aligned} \tag{10.15}$$

As a consequence, the scheduler S_1 of $C[\tau.P]$ and $C[\tau.Q]$ has to be of the form $S_1 = l_1.(\mathbf{if}\ l_C\ \mathbf{then}\ S_C\ \mathbf{else}\ S_R)$. Note that $tl(C'[\tau.P]) = tl(C'[\tau.Q])$ so the two processes cannot be separated by a test. S_C will schedule both (possibly separating them later).

For ($\Rightarrow$) we have

$$\begin{aligned} & p\; p_\omega(l_1{:}(C'[\tau.P] +_q R), S_1, O) + \bar{p}\; p_\omega(l_1{:}(C'[\tau.Q] +_q R), S_1, O) \\ = \;& q(p\; p_\omega(C'[\tau.P], S_C, O) + \bar{p}\; p_\omega(C'[\tau.Q], S_C, O)) + \\ & \bar{q}\; p_\omega(R, S_R, O) && (10.15) \\ = \;& q\; p_\omega(C'[P +_p Q]), S'_C, O) + \\ & \bar{q}\; p_\omega(R, S_R, O) && \text{Ind. Hyp.} \\ = \;& p_\omega(l_1{:}(C'[P +_p Q] +_q R), l_1.(\mathbf{if}\ l'_C\ \mathbf{then}\ S'_C\ \mathbf{else}\ S_R), O) && (10.15) \\ = \;& p_\omega(R_2, S_2, O) \end{aligned}$$

Where $l'_C \in tl(C'[P +_p Q])$ (and thus $l'_C \notin tl(R)$).

For ($\Leftarrow$) we can perform the above derivation in the opposite direction, given that a scheduler for $R_2 = l_1{:}(C'[P +_p Q] +_q R)$ must be of the form $S_2 = l_1.(\mathbf{if}\ l'_C\ \mathbf{then}\ S'_C\ \mathbf{else}\ S_R)$.

- Case $C = C' + R$
  Consider the process $C'[l_0 : \tau.P] + R$. The scheduler S_1 of this process has to choose between $C'[l_0{:}\tau.P]$ and R.

  There are two cases to have a transition using the SUM1, SUM2 rules.

  i) Either $S_1 = S_R$ and

  $$\text{SUM2}\ \frac{(\nu)(R \mid O) \parallel S_R \xrightarrow{\alpha} \mu}{(\nu)(C'[l_0{:}\tau.P] + R \mid O) \parallel S_R \xrightarrow{\alpha} \mu}$$

  In this case

  $$p_\omega(C'[l_0{:}\tau.P] + R, S_R, O) = p_\omega(R, S_R, O) \tag{10.16}$$

  ii) Or $S_1 = S_C$ and

  $$\text{SUM1}\ \frac{(\nu)(C'[l_0{:}\tau.P] \mid O) \parallel S_C \xrightarrow{\alpha} \mu}{(\nu)(C'[l_0{:}\tau.P] + R \mid O) \parallel S_C \xrightarrow{\alpha} \mu}$$

  In this case

  $$p_\omega(C'[l_0{:}\tau.P] + R, S_C, O) = p_\omega(C'[l_0{:}\tau.P], S_C, O) \tag{10.17}$$

Now consider the process $C'[l_0:\tau.Q] + R$. Since P and Q are behind the $l_0:\tau$ action, we have $tl(C'[l_0:\tau.Q] = tl(C'[l_0:\tau.P])$. Thus S_R and S_C will select R and $C'[l_0:\tau.Q]$ respectively and the equations (10.16) and (10.17) will hold.

In the case (i) ($S = S_R$) we have:

$$\begin{aligned} & p\ p_\omega(C'[\tau.P] + R, S_R, O) + \bar{p}\ p_\omega(C'[\tau.Q] + R), S_R, O) \\ = \quad & p\ p_\omega(R, S_R, O) + \bar{p}\ p_\omega(R, S_R, O) \quad (10.16) \\ = \quad & p_\omega(R, S_R, O) \\ = \quad & p_\omega(C'[P +_p Q] + R, S_R, O) \\ = \quad & p_\omega(R_2, S_2, O) \end{aligned}$$

In the case (ii) ($S = S_C$) we have:

$$\begin{aligned} & p\ p_\omega(C'[\tau.P] + R, S_C, O) + \bar{p}\ p_\omega(C'[\tau.Q] + R), S_C, O) \\ = \quad & p\ p_\omega(C'[\tau.P], S_C, O) + \bar{p}\ p_\omega(C'[\tau.Q], S_C, O) \quad (10.17) \\ = \quad & p_\omega(C'[P +_p Q], S'_C, O) \qquad\qquad \text{Ind. Hyp.} \\ = \quad & p_\omega(C'[P +_p Q] + R, S'_C, O) \\ = \quad & p_\omega(R_2, S_2, O) \end{aligned}$$

For ($\Leftarrow$) we can perform the above derivation in the opposite direction.

- Case $C = (\nu a)C'$
  The process $(\nu)((\nu a)C'[X] \mid O)$ has the same transitions as $(\nu)(C'[X] \mid (\nu a)O)$. The result follows by the induction hypothesis.

□

There are two crucial points in the above Theorem. The first is that the labels of the context are copied, thus the scheduler cannot distinguish between $C[l_1:\tau.P]$ and $C[l_1:\tau.Q]$ based on the labels of the context. The second is that P, Q are protected by a τ action labeled by the same label l_1. This is to ensure that in the case of a nondeterministic sum ($C = R + []$) the scheduler cannot find out whether the second operand of the choice is P or Q unless it commits to selecting the second operand. For example let $R = a +_{0.5} 0$, $P = a$, $Q = 0$ (all omitted labels are linear). Then $R_1 = (R + P) +_{0.1} (R + Q)$ is not testing equivalent to $R_2 = R + (P +_{0.1} Q)$ since they can be separated by $O = \overline{a}.\omega$ and a scheduler that resolves $R + P$ to P and $R + Q$ to R (it will be of the form **if** l_P **then** S_P **else** S_R). However, if we take $R'_1 = (R+l_1:\tau.P)+_{0.1}(R+l_1:\tau.Q)$ then R'_1 is testing equivalent to R_2 since now the scheduler cannot see the labels of P, Q so if it selects P then it is bound to also select Q.

The problem with replication is simply the persistence of the processes. Clearly $!P +_p !Q$ cannot be equivalent to $!(P +_p Q)$, since the first replicates only one of P, Q while the second replicates both. However Theorem 10.3.3 together with Proposition 10.3.2 imply that

$$C'[l:(C[l_1:\tau.P] +_p C[l_1:\tau.Q])] \approx_{\mathbf{may}} C'[C[l:(P +_p Q)]] \qquad (10.18)$$

where C is a context without bang and C' is a context without $+$. The same is also true for $\approx_{\mathbf{must}}$. This means that we can lift the sum towards the root of the context until we reach a bang. Intuitively we cannot move the sum outside the bang since each replicated copy must perform a different probabilistic choice with a possibly different outcome.

Theorem 10.3.3 shows that the probabilistic choice is indeed private to the process and invisible to the scheduler. The process can perform it at any time, even in the very beginning of the execution, without making any difference to an outside observer.

10.4 An application to security

In this section we discuss an application of our framework to anonymity. In particular, we show how to specify the Dining Cryptographers protocol so that it is robust to scheduler-based attacks. We first propose a method to encode *secret value passing*, which will turn out to be useful for the specification.

10.4.1 Encoding secret value passing

We propose to encode the passing of a secret message as follows:

$$\begin{array}{lcl} l{:}c(x).P & \triangleq & \sum_{v\in V} l{:}c_v.P[v/x] \\ l{:}\bar{c}\langle v\rangle.P & \triangleq & l{:}\overline{c_v}.P \end{array}$$

where V is the set of values that can be transmitted through channel c. This is the usual encoding of value passing in CCS: we use a non-deterministic sum with a distinct channel c_v for each v. The novelty is that we use the same label in all the branches of the nondeterministic sum. To ensure that the resulting labeling will be deterministic we should restrict the channels c_v and make sure that there will be at most one output on c. We will write $(\nu c)P$ for $(\nu_{v\in V} c_v)P$. For example, the labeling of the following process is deterministic:

$$(\nu c)(l_1{:}c(x).P \mid l{:}(l_2{:}\bar{c}\langle v\rangle +_p l_2{:}\bar{c}\langle w\rangle))$$

This case is a combination of the cases (10.4) and (10.6) of Proposition 10.2.2. The two outputs on c are on different branches of the probabilistic sum, so during an execution at most one of them will be available. Thus there is no ambiguity in scheduling the sum produced by $c(x)$. The scheduler $l.(l_1, l_2)$ will perform a synchronization on c_v or c_w, whatever is available after the probabilistic choice. In other words, using the labels we manage to hide the information about which value was transmitted to P.

10.4.2 Dining cryptographers with probabilistic master

We consider once again the problem of the dining cryptographers, this time adding a factor that we omitted from the previous analysis. We already presented in Section 5.2 a proof that the protocol satisfies anonymity under the assumption of fair coins, that is $p_c(\vec{o}|a_i) = p_c(\vec{o}|a_j)$ for all announcements $\vec{o}$ and users a_i, a_j. In this analysis, however, we only considered the value that each cryptographer announces, without considering the order in which they

$$
\begin{aligned}
Master &\triangleq l_1 : \textstyle\sum_{i=0}^{2} p_i(\underbrace{\overline{m}_0\langle i == 0\rangle}_{l_2} \mid \underbrace{\overline{m}_1\langle i == 1\rangle}_{l_3} \mid \underbrace{\overline{m}_2\langle i == 2\rangle}_{l_4}) \\
Crypt_i &\triangleq \underbrace{m_i(pay)}_{l_{5,i}} . \underbrace{c_{i,i}(coin_1)}_{l_{6,i}} . \underbrace{c_{i,i\oplus 1}(coin_2)}_{l_{7,i}} . \underbrace{\overline{out}_i\langle pay \otimes coin_1 \otimes coin_2\rangle}_{l_{8,i}} \\
Coin_i &\triangleq l_{9,i} : ((\underbrace{\bar{c}_{i,i}\langle 0\rangle}_{l_{10,i}} \mid \underbrace{\bar{c}_{i\ominus 1,i}\langle 0\rangle}_{l_{11,i}}) +_{0.5} (\underbrace{\bar{c}_{i,i}\langle 1\rangle}_{l_{10,i}} \mid \underbrace{\bar{c}_{i\ominus 1,i}\langle 1\rangle}_{l_{11,i}})) \\
Prot &\triangleq (\nu \vec{m})(Master \mid (\nu \vec{c})(\textstyle\prod_{i=0}^{2} Crypt_i \mid \prod_{i=0}^{2} Coin_i))
\end{aligned}
$$

Figure 10.4: Encoding of the dining cryptographers with probabilistic master

make their announcements. In other words, we considered the announcement aad to be the same, whether it corresponds to $c_1 = a, c_2 = a, c_3 = d$ or to $c_2 = a, c_3 = d, c_1 = a$ (in the indicated order).

If we want to allow the cryptographers to make announcements in any order, then the only reasonable way to model the choice of order is non-deterministically. But this leads immediately to a simple attack: if the scheduler is unrestricted then it can base its strategy on the decision of the master, by selecting the paying cryptographer last (or first). Clearly, an external observer would trivially identify the payer just from the fact that he spoke last. A similar situation would arise if the scheduler based its decision on the value of the coins.

A natural question to ask at this point is whether this attack is realistic, or just an artifact of the non-deterministic model. For instance, is it possible for the scheduler to know the decision of the master? The answer is that this attack could appear in practice without even a malicious intention from the part of the scheduler. For example, the payer needs to make one more calculation to add 1 to its announcement, so it could be the case that he needs more time to make his announcement than the other cryptographers so he is scheduled last. Moreover, [Cho07] shows a simple implementation of the Dining Cryptographers in Java where the scheduling problem appears because of the way Java optimizes threads.

In any case, the scheduler restrictions, if any, should be part of the requirements when stating the anonymity properties of a protocol. For example the analysis should state "assuming that the coins are fair and that the scheduler's decisions are independent from the master's choice and from the coins, DC satisfies strong anonymity". This way an implementor of the protocol will have to verify that the scheduler condition is satisfied, or somehow assume that it is.

In our framework we can solve the problem by giving a specification of the DCP in which the choices of the master and of the coins are made invisible to the scheduler. The specification is shown in Figure 10.4. We use some meta-syntax for brevity: The symbols $\oplus$ and $\ominus$ represent the addition and subtraction modulo 3, while $\otimes$ represents the addition modulo 2 (xor). The notation $i == n$ stands for 1 if $i = n$ and 0 otherwise.

There are many sources of nondeterminism: the order of communication between the master and the cryptographers, the order of reception of the coins, and the order of the announcements. The crucial points of our specification, which make the nondeterministic choices independent from the probabilistic ones, are: (a) all communications internal to the protocol (master-cryptographers and cryptographers-coins) are done by secret value passing, and (b) in each probabilistic choice the different branches have the same labels. For example, all branches of the master contain an output on m_0, always labeled by l_2, but with different values each time.

We can extend the definition of strong anonymity to the non-deterministic setting in a straightforward way, as suggested in [BP05]. Now each scheduler S induces a different family of conditional distributions $p_S(\cdot|a)$. So for each scheduler we will define an anonymity system $Syst_S = (\mathcal{A}, \mathcal{O}, p_S)$ and we require that all of them satisfy strong anonymity. That is for all schedulers S, observables o and users a, a': $p_S(o|a) = p_S(o|a')$.

In our example, let $\vec{o}$ represent an observable (the sequence of announcements), and $p_S(\vec{o} \mid \overline{m_i}\langle 1\rangle)$ represent the conditional probability, under the scheduler S, that the protocol produces $\vec{o}$ given that the master has selected cryptographer i as the payer. Thanks to the above independence, the specification satisfies strong anonymity.

Proposition 10.4.1 (Strong anonymity)**.** *The protocol in Figure 10.4 satisfies strong anonymity, that is: for all schedulers S and for all observables $\vec{o}$: $p_S(\vec{o} \mid \overline{m}_0\langle 1\rangle) = p_S(\vec{o} \mid \overline{m}_1\langle 1\rangle) = p_S(\vec{o} \mid \overline{m}_2\langle 1\rangle)$.*

Proof. Since the process is finite and so is the number of schedulers the Proposition can be verified by calculating the probability of all traces under all schedulers (this could be even done automatically). Here we make a higher level argument to show that the Proposition holds.

Let v_1, v_2, v_3 be the values announced by the cryptographers, that is v_i is the output of the subprocess $\overline{out}_i\langle pay \otimes coin_1 \otimes coin_2\rangle$. These values depend only on the selection of the master (pay) and the outcome of the coins ($coin_1, coin_2$) and not on the scheduler, the latter can only affect their order. From the proof of strong anonymity for a fixed announcement order (Theorem 5.2.1) we know that $p(v_1, v_2, v_3|a_i) = p(v_1, v_2, v_3|a_j)$ for all cryptographers i, j and all values of v_1, v_2, v_3.

Now the observables of the protocol are of the form $\vec{o} = \overline{out_{k_1}}\langle v_{k_i}\rangle, \overline{out_{k_2}}\langle v_{k_2}\rangle, \overline{out_{k_3}}\langle v_{k_3}\rangle$ where k_1, k_2, k_3 is the index of the cryptographer who speaks first, second and third respectively. The order (that is the k_i's) depends on the scheduler. However, in all random choices the same labels appear in both branches of the choice, so a scheduler cannot use an **if-then-else** test to "detect" the outcome of the choice (it would be useless since the same branch of the **if** would be always activated). As a consequence, the order is fixed for a particular scheduler, that is a scheduler uniquely defines the k_i's above. With a fixed order, the probability of each $\vec{o}$ is equal to the probability of the corresponding v_i's, thus

$$p_S(\vec{o} \mid \overline{m}_i\langle 1\rangle) = p(v_1, v_2, v_3|a_i) = p(v_1, v_2, v_3|a_j) = p_S(\vec{o} \mid \overline{m}_j\langle 1\rangle)$$

□

$$P \quad ::= \ldots \mid l{:}\{P\} \qquad\qquad \text{INDEP} \quad \frac{P \parallel T \xrightarrow{\alpha} \mu}{l{:}\{P\} \parallel l.S, T \xrightarrow{\alpha} \mu'} \text{ where } \mu'(P' \parallel S, T') = \mu(P' \parallel T')$$
$$CP \quad ::= P \parallel S, T$$

Figure 10.5: Adding an "independent" scheduler to the calculus

Note that different schedulers will produce different traces (we still have nondeterminism) but they will not depend on the choice of the master.

Some previous treatment of the DCP, including [BP05], had solved the problem of the leak of information due to too-powerful schedulers by simply considering as observable sets of announcements instead than sequences. Thus one could think that using a true concurrent semantics, for instance event structures, would solve the problem. We would like to remark that this is false: true concurrency would weaken the scheduler enough in the case of the DCP, but not in general. For instance, it would not help in the anonymity example in the beginning of this chapter.

10.4.3 Dining cryptographers with nondeterministic master

Up to now we considered the master in the dining cryptographers to be probabilistic, that is we assume that the master makes his decision using some probability distribution. An interesting question is whether we can remove this assumption, that is make the same analysis with a non-deterministic master. However, this case poses a conceptual problem: as we discussed in the previous paragraph, the decision of the master should be invisible to the scheduler. But if the master is non-deterministic then the scheduler itself will make the decision, so how is it possible for a scheduler to be oblivious to his own choices?

We sketch here a method to hide also certain nondeterministic choices from the scheduler. First we need to extend the calculus with the concept of a second *independent* scheduler T that we assume to resolve the nondeterministic choices that we want to make transparent to the main scheduler S. The new syntax and semantics are shown in Figure 10.5. $l : \{P\}$ represents a process where the scheduling of P is protected from the main scheduler S. The scheduler S can "ask" T to schedule P by selecting the label l. Then T resolves the nondeterminism of P as expressed by the INDEP rule. Note that we need to adjust also the other rules of the semantics to take T into account, but this change is straightforward. We assume that T does not collaborate with S so we do not need to worry about the labels in P.

To model the dining cryptographers with nondeterministic master we replace the $Master$ process in Figure 10.4 by the following one.

$$Master \triangleq l_1{:}\Big\{\textstyle\sum_{i=0}^{2} l_{12,i}{:}\tau.(\underbrace{\overline{m}_0\langle i == 0\rangle}_{l_2} \mid \underbrace{\overline{m}_1\langle i == 1\rangle}_{l_3} \mid \underbrace{\overline{m}_2\langle i == 2\rangle}_{l_4})\Big\}$$

Essentially we have replaced the probabilistic choice by a *protected* nondeterministic one. Note that the labels of the operands are different but this is not a problem since this choice will be scheduled by T. Note also that after

the choice we still have the same labels l_2, l_3, l_4, however the labeling is still deterministic, similarly to the case 10.5 of Proposition 10.2.2.

In case of a nondeterministic selection of anonymous events, and a probabilistic anonymity protocol, the notion of strong anonymity has not been established yet, although some possible definitions have been discussed in [BP05]. Our framework makes it possible to give a natural and precise definition. As we did in the previous paragraph, we will define an anonymity system $Syst_S = (\mathcal{A}, \mathcal{O}, p_S)$ for each scheduler S, where $p_S(\cdot|a_i)$ is a probability distribution on $\mathcal{O}$ corresponding to cryptographer i. The selection of cryptographer i is made by the corresponding scheduler $T_i = l_{12,i}$, so we define p_S as

$$p_S(\vec{o}|a_i) = p_{S,T_i}(\vec{o})$$

where p_{S,T_i} is the probability measure on traces induced by the semantics of the process $Prot \parallel S, T_i$. Finally we require all anonymity systems $Syst_S$ to be strongly anonymous in the usual sense (Def. 5.1.1).

Definition 10.4.2 (Strong anonymity for nondeterministic anonymous events)**.** *A protocol with nondeterministic selection of the anonymous event satisfies strong anonymity iff all the anonymity systems $Syst_S = (\mathcal{A}, \mathcal{O}, p_S)$ defined above are strongly anonymous. This means that for all observables $\vec{o} \in \mathcal{O}$, schedulers S, and independent schedulers T_i, T_j (selecting anonymous events a_i, a_j), we have: $p_{S,T_i}(\vec{o}) = p_{S,T_j}(\vec{o})$.*

We can prove the above property for our protocol:

Proposition 10.4.3. *The DCP with nondeterministic master, specified in this section, satisfies strong anonymity.*

Proof. Similar to Proposition 10.4.1, since $p_{S,T_i}(\vec{o})$ is equal to $p_S(\vec{o} \mid \overline{m}_i\langle 1\rangle)$ in the protocol with probabilistic master. □

10.5 Related work

The works that are most closely related to ours are [CCK+06a, CCK+06b, GvRS07]. The authors of [CCK+06a, CCK+06b] consider probabilistic automata and introduce a restriction on the scheduler to the purpose of making them suitable to applications in security protocols. Their approach is based on dividing the actions of each component of the system in equivalence classes (*tasks*). The order of execution of different tasks is decided in advance by a so-called *task scheduler*. The remaining nondeterminism within a task is resolved by a second scheduler, which models the standard *adversarial scheduler* of the cryptographic community. This second entity has limited knowledge about the other components: it sees only the information that they communicate during execution.

Reference [GvRS07] defines a notion of admissible scheduler by introducing an equivalence relation on the nodes of the execution tree, and requiring that an admissible scheduler maps two equivalent nodes into bisimilar steps. Both we and [GvRS07] have developed, independently, the solution to the problem of the scheduler in the Dining Cryptographers as an example of application to security.

Another work along these lines is [dAHJ01], which uses partitions on the state-space to obtain partial-information schedulers. However [dAHJ01] considers a synchronous parallel composition, so the setting is rather different from [CCK+06a, CCK+06b, GvRS07] and ours.

Our approach is in a sense *dual* to the above ones. Instead of defining a restriction on the class of schedulers, we provide a way to specify that a choice is transparent to the schedulers. We achieve this by introducing labels in process terms, used to represent both the nodes of the execution tree and the next action or step to be scheduled. We make two nodes indistinguishable to schedulers, and hence the choice between them private, by associating to them the same label. Furthermore, in contrast with [CCK+06a, CCK+06b], our "equivalence classes" (schedulable actions with the same label) can change dynamically, because the same action can be associated to different labels during the execution. However we don't know at the moment whether this difference determines a separation in the expressive power.

Eleven

Analysis of a contract-signing protocol

Up to now we have focused exclusively on the notion of anonymity. In this chapter we look at the more general category of *probabilistic security protocols*, that is protocols involving probabilistic choices and often relying on specific randomized primitives such as the *Oblivious Transfer* ([Rab81]). Such protocols are used for various purposes including signing contracts, sending certified email and protecting the anonymity of communication agents. There are various examples in this category, notably the contract signing protocol in [EGL85] and the privacy-preserving auction protocol in [NPS99].

A large effort has been dedicated to the formal verification of security protocols, and several approaches based on process-calculi techniques have been proposed. However, in the particular case of probabilistic protocols, they have been analyzed mainly by using model checking methods, while only few attempts of applying process calculi techniques have been made. One proposal of this kind is [AG02], which defines a probabilistic version of the noninterference property, and uses a probabilistic variant of CCS and of bisimulation to analyze protocols wrt this property.

In this chapter we show how to apply the tools developed in the previous chapter to analyze probabilistic security protocols. We express the intended security properties of the protocol using the may-testing preorder discussed in the previous chapter: a process P is considered smaller than a process Q if, for each test, the probability of passing the test is smaller for P than for Q. Following the lines of [AG99], a test can be seen as an adversary who interacts with an agent in order to break some security property. Then the analysis proceeds as follows: we first model the protocol in the CCS_σ calculus, then we create a specification that models the ideal behavior of the protocol and can be shown to satisfy the desired property. The final step is to show that the protocol is smaller than the specification with respect to the testing preorder. If this holds, then an attack of any possible adversary (viewed as an arbitrary test) has even smaller probability of breaking the protocol than of breaking the specification, so the protocol itself satisfies the desired property.

We illustrate this technique on a fair exchange protocol (used for contract signing), where the property to verify is fairness. In this kind of protocol two agents, A and B, want to exchange information simultaneously, namely each of them is willing to send its secrets only if he receives the ones of the other party.

We consider the Partial Secrets Exchange protocol (PSE, [EGL85]) which uses the Oblivious Transfer as its main primitive. An important characteristic of the fair exchange protocols is that the adversary is in fact one of the agents and not an external party. After encoding the protocol in CCS_σ, we give a specification which models the ideal behavior of A. We then express fairness by means of a testing relation between the protocol and the specification and we prove that it holds.

It should be noted that in this analysis, the ability of CCS_σ to hide information from the scheduler is not used in a direct way. In fact we use linear labelings in both the protocol and the specification. However, the distributivity property of the probabilistic plus (which is based on the ability to hide information from the scheduler) plays a crucial role in the proof of the testing relation between the protocol and the specification.

Plan of the chapter The rest of the chapter is organized as follows: in the next section we introduce some syntactic constructs for CCS_σ processes that are needed to model the PSE protocol. In Section 11.2 we illustrate the Oblivious Transfer primitive, the Partial Secrets Exchange protocol (PSE), and their encoding in CCS_σ. In Section 11.3 we specify the fairness property and we prove the correctness of PSE. In Section 11.4 we discuss related work, notably the analysis of the PSE protocol using probabilistic model checking.

11.1 Syntactic extensions of CCS_σ

In this section we add some constructs to CCS_σ that are needed to model the fair exchange protocol. Namely we add tuples, polyadic value passing and a matching operator. These constructs are encoded in the pure calculus, so they are merely "syntactic sugar", they do not change the semantics of the calculus.

11.1.1 Creating and splitting tuples

Protocols often concatenate messages and split composed messages in parts. We encode tuples of channels by replacing them with a single channel that represents the tuple:

$$\langle v_1, \ldots, v_n \rangle \triangleq v$$

where v is the corresponding composed channel. We also allow the decomposition of a tuple using the construct let $\langle x_1, \ldots, x_n \rangle = v$ in P encoded as

$$\text{let } \langle x_1, \ldots, x_n \rangle = v \text{ in } P \triangleq P[v_1/x_1, \ldots, v_n/x_n]$$

where v is the channel representing the tuple $\langle v_1, \ldots, v_n \rangle$.

11.1.2 Polyadic value passing

We already discussed a way to use value passing in Section 10.4.1, using the following encoding:

$$\begin{aligned} l{:}c(x).P &\triangleq \textstyle\sum_{v \in V} l_c{:}c_v.P[v/x] \\ l{:}\bar{c}\langle v \rangle.P &\triangleq l{:}\overline{c_v}.P \end{aligned}$$

where V is the set of possible values that can be sent through channel c and for each $v \in V$, c_v is a distinct channel and l_v is a distinct label. The goal in Section 10.4.1 was to encode *secret* value passing, where the scheduler knows that some value was transmitted in the channel but does not know which one. For that reason we were using the same label l in all branches of the nondeterministic plus. In this chapter we are not interested in hiding this information so we use different labels for each branch.

We can pass polyadic values by using tuples. Let $V_1, \ldots, V_n$ be the set of values for the variables $x_1, \ldots, x_n$ respectively and $V = V_1 \times \ldots \times V_n$. We encode polyadic value passing as follows:

$$\begin{aligned} l{:}c(x_1, \ldots, x_n).P &\triangleq l{:}c(x).\text{let } \langle x_1, \ldots, x_n \rangle = x \text{ in } P \\ &= \textstyle\sum_{v \in V} l_v{:}c_v.P[v_1/x_1, \ldots, v_n/x_n] \end{aligned}$$

Here $v \in V$ is a composed channel representing the tuple $\langle v_1, \ldots, v_n \rangle$. The polyadic output is simply the output of the corresponding tuple.

Note that the substitution operator $P[v/x]$ does not replace occurrences of x that are bound in P by some other input or let..in construct Note also that the encoding of value passing does not allow the use of free variables, that is variables that are bounded by no input. Such variables will not be substituted during the translation and the resulting process will not be a valid CCS_σ process.

11.1.3 Matching

Now that we can perform an input on a variable, we might need to test the value that we received. This is the purpose of matching, denoted by the construct $[x = y]P$. We encode it as follows:

$$\begin{aligned} [c = c]P &\triangleq P \\ [c = d]P &\triangleq 0 \quad c \neq d \end{aligned}$$

where c, d are channel names. If variables are used for matching then we need first to substitute variables by channels following the encoding of value passing, and then apply the encoding of matching. For example

$$\begin{gathered} \bar{c}\langle v \rangle \mid \bar{c}\langle w \rangle \mid c(x).([x = v]P \mid [x = w]Q) = \\ \overline{c_v} \mid \overline{c_w} \mid c_v.([v = v]P_v \mid [v = w]Q_v) + c_w.([w = v]P_w \mid [w = w]Q_w) = \\ \overline{c_v} \mid \overline{c_w} \mid c_v.(P_v \mid 0) + c_w.(0 \mid Q_w) \end{gathered}$$

where $P_v = P[v/x], P_w = P[w/x]$ and similarly for Q.

11.1.4 Using extended syntax in contexts

We also allow the new syntactic constructs to be used in contexts in the following way. A context $C[]$ with extended syntax denotes the pure context $C'[]$ that is produced using the encodings. Note that the translation is only done once for the context itself, so $C[P]$ denotes $C'[P]$, P is not translated. However

note that P cannot contain free variables, so if P' is the translation of P, then the translation of $C[P]$ is equal to $C'[P']$.

Since extended contexts denote pure contexts then $\sqsubseteq_{\mathbf{may}}, \sqsubseteq_{\mathbf{must}}$ are pre-congruences also wrt extended contexts.

11.2 Probabilistic Security Protocols

In this section we discuss probabilistic security protocols based on the Oblivious Transfer and we show how to model them using the CCS_σ calculus.

11.2.1 1-out-of-2 Oblivious Transfer

The Oblivious Transfer is a primitive operation used in various probabilistic security protocols. In this particular version a sender A sends exactly one of the messages M_1, M_2 to a receiver B. The latter receives i and M_i where i is 1 or 2, each with probability 1/2. Moreover A should get no information about which message was received by B. More precisely the protocol $\text{OT}^1_2(A, B, M_1, M_2)$ should satisfy the following conditions:

1. If A executes $\text{OT}^1_2(A, B, M_1, M_2)$ properly then B receives exactly one message, $(1, M_1)$ or $(2, M_2)$, each with probability 1/2.

2. After the execution of $\text{OT}^1_2(A, B, M_1, M_2)$, if it is properly executed, for A the probability that B got M_i remains 1/2.

3. If A deviates from the protocol, in order to increase his probability of learning what B received, then B can detect his attempt with probability at least 1/2.

It is worth noting that in the literature the reception of the index i by B is often not mentioned, at least not explicitly ([EGL85]). However, omitting the index can lead to possible attacks. Consider the case where A executes (properly) $\text{OT}^1_2(M_1, M_1)$. Then B will receive M_1 with probability one, but he cannot distinguish it from the case where he receives M_1 as a result of $\text{OT}^1_2(M_1, M_2)$. So A is forcing B to receive M_1. We will see that, in the case of the PSE protocol, A could exploit this situation in order to get an unfair advantage. Note that the condition 3 does not apply to this situation since this cannot be considered as a deviation from the Oblivious Transfer. A generic implementation of the Oblivious Transfer could not detect such behavior since A executes OT properly, the problem lies only in the data being transferred.

Using the indexes, however, solves the problem since B will receive $(2, M_1)$ with probability one half. This is distinguishable from any outcome of $\text{OT}^1_2(M_1, M_1)$ so, in the case of PSE, B could detect that he's being cheated. Implementations of the Oblivious Transfer do provide the index information, even though sometimes it is not mentioned ([EGL85]). In other formulations of the OT the receiver can actually select which message he wants to receive, so this problem is irrelevant.

Encoding in CCS_σ The Oblivious Transfer can be modeled in CCS_σ using a server process to coordinate the transfer, making it impossible to cheat. The

```
PSE (A, B, {a_i}_i, {b_i}_i) {
    for i = 1 to n do
        OT^1_2(A,B, a_i, a_{i+n})
        OT^1_2(B, A, b_i, b_{i+n})
    next
    for j = 1 to m do
        for i = 1 to 2n do
            A sends jth bit of a_i to B
        for i = 1 to 2n do
            B sends jth bit of b_i to B
    next
}
```

Figure 11.1: Partial Secrets Exchange protocol

processes of the sender and the server are the following:

$$\begin{aligned} \mathrm{OT}^1_2(m_1, m_2, c_{as}) &\triangleq \overline{c_{as}}\langle m_1\rangle.\overline{c_{as}}\langle m_2\rangle.0 \\ S(c_{as}, c_{sb}) &\triangleq c_{as}(x_1).c_{as}(x_2).(\overline{c_{sb}}\langle 1, x_1\rangle +_{0.5} \overline{c_{sb}}\langle 2, x_2\rangle) \end{aligned}$$

where m_1, m_2 are the names to be sent. As in most cases in this chapter, we omit the labeling assuming that a linear one is used. c_{as} is a channel private to A and S and c_{sb} a channel private to B and S. Each agent communicates only with the server and not directly with the other agent. B receives the message from the server (which should be in parallel with A and B) by making an input action on c_{sb}.

It is easy to see that these processes correctly implement the Oblivious Transfer. The only requirement is that A should not contain c_{sb}, so that he can only communicate with B through the server.

11.2.2 Partial Secrets Exchange Protocol

This protocol is the core of three probabilistic protocols for contract signing, certified email and coin tossing, all presented in [EGL85]. It involves two agents, each having $2n$ secrets split in pairs, $(a_1, a_{n+1}), ..., (a_n, a_{2n})$ for A and $(b_1, b_{n+1}), ..., (b_n, b_{2n})$ for B. Each secret consists of m bits. The purpose is to exchange a single pair of secrets under the constraint that, if at a specific time B has one of A's pairs, then with high probability A should also have one of B's pairs and vice versa.

The protocol, displayed in Figure 11.1, consists of two parts. During the first A and B exchange their pairs of secrets using OT^1_2. After this step A knows exactly one half of each of B's pairs and vice versa. During the second part, all secrets are exchanged bit per bit. Half of the bits received are already known from the first step, so both agents can check whether they are valid. Obviously, if both A and B execute the protocol properly then all secrets are revealed.

The problem arises when B tries to cheat and sends incorrectly some of his secrets. In this case it can be proved that with high probability some of the tests of A will fail causing A to stop the execution of the protocol and avoid

revealing his secrets. The idea is that, in order for B to cheat, he must send at least one half of each of his pairs incorrectly. However he cannot know which of the two halves is already received by A during the first part of the protocol. So a pair sent incorrectly will have only one half probability of being accepted by A, leading to a total 2^{-n} probability of success.

Now imagine, as discussed in Section 11.2.1, that B executes $\mathrm{OT}^1_2(B, A, b_i, b_i)$, thus forcing A to receive b_i. Now, in the second part, he can send all $\{b_{i+n} \mid 1 \leq i \leq n\}$ incorrectly without failing any test. Moreover A cannot detect this situation. If indexes are available A will receive $(2, b_{i+n})$ with probability one half and since he knows that b_{i+n} is not the second half of the corresponding pair he will stop the protocol.

Encoding in CCS_σ In this paragraph we present an encoding of the PSE protocol in the CCS_σ calculus. First, it should be noted that the secrets exchanged by PSE should be *recognizable*, which means that agent A cannot compute B's secrets, but he can recognize them upon reception. Of course a secret can be recognized only as a whole, no single bit can be recognized by itself. To model this feature we allow B's secrets to appear in A's process, as if A knew them. However we allow a secret to appear only as a whole (not decomposed) and only inside a match construct, which means that it can only be used to recognize another message.

The encoding is displayed in Figure 11.2, as usual the labeling is omitted assuming a linear one. We denote by a_i (resp. b_i) the i-th secret of A (resp. B) and by a_{ij} (resp. b_{ij}) the j-th bit of a_i (resp. b_i). r_i is the i-th message received by Oblivious Transfer and k_i is the corresponding index.

The first part consists of the first 7 lines of the process definition. In this part A sends his pairs using OT^1_2, receives the ones of B and decomposes them. To check the received messages A starts a loop of n steps, each of which is guarded by an input action on q_i for synchronization. During the i-th step, the $TestOT$ sub-process tests r_i against b_i or b_{i+n} depending on the outcome of the OT, that is on the value of k_i. The $testpair_i$ channels are used to send the needed values to the $TestOT$ sub-process.

The second part consists of a loop of m steps, each of which is guarded by an input action on s_j. During each step the j-th bit of each secret is sent and the corresponding bits of B are received in d_{ij}. Then there is a nested loop of n tests controlled by the input actions on t_{ij}. Each test, performed by the $Test$ sub-process, ensures that B's bits are valid. $Test(i, j)$ checks the j-th bit of the i-th pair. The bit received during the first part, namely r_{ij}, is compared to d_{ij} or $d_{(i+n)j}$ depending on k_i. If the bit is valid, an output action on t_{i+1j} is performed to continue to the next test. Again, the $testbit_{ij}$ channels are used to send the necessary values to the $Test$ sub-process.

Finally, an instance of the protocol is an agent A put in parallel with servers for all oblivious transfers:

$$I \triangleq \nu c_{as_1} \dots c_{as_n} \nu c_{sa_1} \dots c_{sa_n} \big(A \mid \prod_{i=1}^{n} S(c_{as_i}, c_{sb_i}) \mid S(c_{bs_i}, c_{sa_i})\big)$$

the channels c_{as_i} and c_{sa_i} are restricted to prevent B from communicating directly with A without using the Oblivious Transfer.

$A \triangleq$

$\nu testpair_1 \ldots \nu testpair_n \, \nu testbit_{11} \ldots \nu testbit_{nm}$

$\nu q_1 \ldots \nu q_{n+1} \, \nu s_1 \ldots \nu s_{m+1} \, \nu t_{11} \ldots \nu t_{n+1m}$

$\prod_{i=1}^{n} \mathrm{OT}^1_2(a_i, a_{(i+n)}, c_{as_i}) \mid$	Send half a pair by OT
$c_{sa_1}(k_1, r_1).\text{let } \langle r_{11}, \ldots, r_{1m} \rangle = r_1 \text{ in } \ldots$	Receive half of each of B's pairs
$c_{sa_n}(k_n, r_n).\text{let } \langle r_{n1}, \ldots, r_{nm} \rangle = r_n \text{ in}$	and decompose them in bits
$(\overline{q_1} \mid$	Loop over pairs $(1 \ldots n)$
$\quad \prod_{i=1}^{n} q_i.\, \overline{testpair_i}\langle k_i, r_i \rangle \mid$	Check i-th received pair
$q_{n+1}.(\overline{s_1} \mid$	Loop over bits $(1 \ldots m)$
$\quad \prod_{j=1}^{m} s_j.\, \overline{c_p}\langle a_{1j} \rangle. \ldots \overline{c_p}\langle a_{(2n)j} \rangle.$	Send j-th bit of all secrets
$\qquad c_p(d_{1j}). \ldots c_p(d_{(2n)j}).$	Receive j-th bit of all B's secrets
$\qquad (\overline{t_{1j}} \mid$	Check received bits
$\qquad\quad \prod_{i=1}^{n} t_{ij}.\, \overline{testbit_{ij}}\langle k_i, r_{ij}, d_{ij}, d_{(i+n)j} \rangle \mid$	
$\qquad t_{n+1j}.\overline{s_{j+1}}) \mid$	
$s_{m+1}.\overline{ok}\,)) \mid$	Success. End of protocol
$\prod_{i=1}^{n} TestOT_{spec}(i) \mid$	Pair tests
$\prod_{j=1}^{m} \prod_{i=1}^{n} Test_{spec}(i, j)$	Bit tests

$TestOT(i) \triangleq testpair_i(k, w).([k = 1][w = b_i]\, \overline{q_{i+1}} \mid [k = 2][w = b_{i+n}]\, \overline{q_{i+1}})$

$Test(i, j) \triangleq testbit_{ij}(k, w, x, y).([k = 1][w = x]\, \overline{t_{i+1j}} \mid [k = 2][w = y]\, \overline{t_{i+1j}})$

Figure 11.2: Encoding of PSE protocol

11.3 Verification of Security Properties

A well known method for expressing and proving security properties using process calculi is by means of *specifications*. A specification P_{spec} of a protocol P is a process which is simple enough in order to prove (or accept) that it models the correct behavior of the protocol. Then the correctness of P is implied by $P \simeq P_{spec}$ where $\simeq$ is a testing equivalence. The idea is that, if there exists an attack for P, this attack can be modeled by a test O which performs the attack and outputs ω if it succeeds. Then P should pass the test and since $P \simeq P_{spec}$, P_{spec} should also pass it, which is a contradiction (no attack exists for P_{spec}).

However, in case of probabilistic protocols, attacks do exist but only succeed with a very small probability. So examining only the ability of passing a test is not sufficient since the fact that P_{spec} has an attack is no longer contradictory. Instead we will use a specification which can be shown to have very small probability of being attacked and we will express the correctness of P as $P \sqsubseteq_{\mathbf{may}} P_{spec}$ where $\sqsubseteq_{\mathbf{may}}$ is the may-testing preorder defined in Section 10.3.

$A_{spec} \triangleq$

... same as the original protocol ...

$\prod_{i=1}^{n} TestOT_{spec}(i) \mid$ Pair tests

$\nu guess_1 \ldots \nu guess_n$

$\left(\prod_{j=1}^{m}\prod_{i=1}^{n} Test_{spec}(i,j) \mid \prod_{i=1}^{n}(!\,\overline{guess_i} +_{0.5} 0)\right)$ Bit tests

$TestOT_{spec}(i) \triangleq testpair_i(k,w).\overline{q_{i+1}}$

$Test_{spec}(i,j) \triangleq testbit_{ij}(k,w,x,y).([x = b_{ij}][y = b_{(i+n)j}]\,\overline{t_{i+1j}} \mid guess_i.\overline{t_{i+1j}})$

Figure 11.3: A specification for the PSE protocol

Then an attack of high probability for P should be applicable with at least the same probability for P_{spec} which is contradictory. In this chapter we only use may-testing so we will simply write $\sqsubseteq$ for $\sqsubseteq_{\mathbf{may}}$.

11.3.1 A specification for PSE

Let us recall the fairness property for the PSE protocol.

> If B receives one of A's pairs then with high probability A should also be able to receive one of B's pairs.

First, we must point out an important difference between this type of protocols and the traditional cryptographic ones. In traditional protocols both A and B are considered honest. The purpose of the protocol is to ensure that no outside adversary can access the messages being transferred. On the other hand, in PSE the adversary is B himself, who might try to deviate from the protocol in order to get A's secrets without revealing his own.

As a consequence we will give a specification only for A, modeling his ideal behavior under any possible behavior of B. The goal for A is to timely detect a cheating attempt of B. A safe way to do this is to allow A to know in advance the message that he is about to receive. Of course this is not realistic in practice but this is typical when using specifications to prove security properties: we model the ideal behavior, possibly in a non-implementable way, and we show that the actual implementation is equivalent.

Knowing the real message, A can test whether each bit that he receives is correct or not. However the tests should not be strict. Even if B is sending incorrect data, the specification should accept it with a certain probability because in the real protocol there is a non-zero (but small) probability of accepting incorrect data. Essentially, the specification models A's behavior under the most successful attack, that is an attack in which B can cheat with the highest possible probability (which is still low enough).

The specification is displayed in Figure 11.3. It is the same as the original protocol, except from the pair and bit tests. The pair test, performed by $TestOT_{spec}$, accepts all messages without really testing anything. On the other hand, $Test_{spec}$ tests the incoming bits against the real ones (and not against

the ones received by the OT as $Test$ does). If both bits are correct they are accepted. However, even if the bits are not correct they can be accepted if an input on channel $guess_i$ is possible. This channel denotes the fact that B was able to guess which part of pair i was received by A, thus he can send the other part incorrectly without being detected. This should happen with probability one half for each pair, which is modeled by the sub-process $\prod_{i=1}^{n}(!\overline{guess_i}+_{0.5}0)$ that runs in parallel with the tests. Note that the guess is made once for each pair, if succeeded then B can send all bits of the corresponding pair incorrectly without being detected.

Note that in the specification the input from the Oblivious Transfer is not used at all and since both bits are tested against the real ones we can be sure that the specification can only be cheated to the extend allowed by $guess_i$. In the rest of this section we prove the correctness of PSE. To achieve that we first show that the specification satisfies the fairness property. Then we prove that the original protocol is smaller than the specification wrt the may-testing preorder.

11.3.2 Proving the correctness of PSE

Correctness of the specification. First we show that the specification is indeed a proper specification for PSE with respect to fairness. This means that, if B does not reveal his secrets then A should reveal his own ones with very small probability. So suppose that B wants to cheat and let l be the maximum number of bits that B is willing to reveal for his secrets. Since one pair is enough for A, B should send at least one of the first $l+1$ bits of each of his pairs incorrectly.

As we already discussed A_{spec} knows all the correct bits of B's secrets and he can test them when they are received. The sub-process $Test_{spec}(i,j)$ will succeed with probability 1 if b_{ij} and $b_{(i+n)j}$ are sent correctly, but only with probability 1/2 if not (since channel $guess_i$ is activated only with probability 1/2). If the test fails then the whole process stalls. Since incorrect bits will be sent for all pairs in the first $l+1$ steps, the total probability of advancing to step $l+2$ and reveal its $l+2$ bits is 2^{-n}.

This means that A_{spec} satisfies fairness. If B at some point of the protocol has l bits of one of A's pairs, then with probability at least $1-2^{-n}$ A will have $l-1$ bits of at least one of B's pairs. If $l=m$ (B has a whole pair) then A should have at least $m-1$ bits and the last bit can be easily computed by trying both 0 and 1. In other words B cannot gain an advantage of more than one bit with probability greater than 2^{-n}.

Relation between the protocol and the specification Having proved the correctness of the specification with respect to fairness, it remains to show its relation with the original protocol. An instance of the specification is a process A_{spec} put in parallel with servers for all oblivious transfers:

$$I_{spec} \triangleq \nu c_{as_1} \ldots \nu c_{as_n} \nu c_{sa_1} \ldots \nu c_{sa_n} \\ \left(A_{spec} \mid \prod_{i=1}^{n} (S(c_{as_i}, c_{sb_i}) \mid S(c_{bs_i}, c_{sa_i})\right)$$

PSE will be considered correct wrt fairness if:

$$I \sqsubseteq I_{spec}$$

If $I \sqsubseteq I_{spec}$ holds then if I is vulnerable with high probability to an attack O, then I_{spec} will be also vulnerable with at least the same probability. Since we know that the probability of a successful attack for I_{spec} is very small, we can conclude that an attack on I is very unlikely.

Theorem 11.3.1. *PSE is correct with respect to fairness.*

Proof. We want to prove that $I \sqsubseteq I_{spec}$. The two processes differ only in the definition of pair and bit tests. We define I_w to be the same as I_{spec} after replacing $TestOT_{spec}$ with $TestOT_w$ and $Test_{spec}$ with $Test_w$ defined as:

$$TestOT_w(i) \triangleq \begin{cases} TestOT_{spec}(i) & \text{if } i < w \\ TestOT(i) & \text{if } i \geq w \end{cases}$$

$$Test_w(i,j) \triangleq \begin{cases} Test_{spec}(i,j) & \text{if } i < w \\ Test(i,j) & \text{if } i \geq w \end{cases}$$

The idea is that I_w behaves as the specification for the first $w-1$ pairs and as the original protocol for the other ones. First note that $I_{spec} = I_{n+1}$. Moreover I_1 is the same as I with the addition of the $\prod_{i=1}^{n}(!\,\overline{guess_i} +_{0.5} 0)$ sub-process. However, the $guess_i$ channels are restricted and never used (since no occurrence of $Test_{spec}$ exists in I_1) so it is easy to show that $I \approx I_1$. Then we can prove the correctness of PSE by induction on w and it suffices to show that

$$I_w \sqsubseteq I_{w+1} \quad \forall w \in \{1..n\} \tag{11.1}$$

We will show that the relation (11.1) holds following a sequence of transformations that respect the $\sqsubseteq$ preorder. I_w and I_{w+1} differ only in the $Test$ sub-processes. Moreover $TestOT_w(i)$ and $TestOT_{w+1}(i)$ differ only for $i = w$, for which we have:

$$\begin{aligned} TestOT_w(w) &= testpair_i(k,w). \\ &\quad ([k=1][w=b_i]\overline{q_{w+1}} \mid [k=2][w=b_{i+n}]\overline{q_{w+1}}) \\ TestOT_{w+1}(w) &= testpair_i(k,w).\overline{q_{w+1}} \end{aligned}$$

Since k can only have one value, the one branch of $TestOT_w(w)$ will stall. So $TestOT_{w+1}(w)$ is the same as $TestOT_w(w)$ except that it doesn't test anything, so it is easy to see that $TestOT_w \sqsubseteq TestOT_{w+1}$.

Since $\sqsubseteq$ is a precongruence (Theorem 10.3.2) we can replace the $TestOT_w$ sub-processes in I_w by $TestOT_{w+1}$. Let K be the resulting process, we have that $I_w \sqsubseteq K$. Now K and I_{w+1} differ only in the $Test_w$ processes and again $Test_w(i,j)$ and $Test_{w+1}(i,j)$ differ only for $i = w$. However $Test_w(w,j)$ is not smaller than $Test_{w+1}(w,j)$ so we cannot replace the first by the second.

In order to overcome this problem we notice that k_w and r_w were received through the c_{sa_w} channel. Since this channel is restricted, r_w must have been transferred using the Oblivious Transfer server $S(bs_w, sa_w)$. This process receives two values x_1, x_2 and sends one of them, each with probability one half.

Now let K', I'_{w+1} be the processes obtained by K, I_{w+1} respectively by replacing x_1, x_2 in $S(bs_w, sa_w)$ by b_w, b_{w+n}. (that is by hard-coding the correct w-th pair in the Oblivious Transfer). It is easy to see that $K \sqsubseteq K'$ since their only difference are in the matches containing k_w, r_w and since K' contains the correct values the matches are at least as probable to succeed as in K. Moreover $I'_{w+1} \approx I_{w+1}$ since both don't use k_w, r_w at all. It is now sufficient to show that $K' \sqsubseteq I'_{w+1}$.

Now K' contains the modified OT server for the w-th pair

$$S(c_{bs_w}, c_{sa_w}) = c_{bs_w}(x_1).c_{bs_w}(x_2).(\overline{c_{sa_w}}\langle 1, b_w\rangle +_{0.5} \overline{c_{sa_w}}\langle 2, b_{w+n}\rangle)$$

and it can be written in the form:

$$K' = \nu(M|S(c_{bs_w}, c_{sa_w}))$$

where ν denotes (for simplicity) the restriction on all OT channels. From the distributivity of the probabilistic plus $+_p$ (Theorem 10.3.3) we have

$$\begin{aligned} K' &\approx \nu(M|S(1, b_w)) +_{0.5} \nu(M|S(2, b_{w+n}) \quad \text{where} \qquad (11.2)\\ S(k, w) &= c_{bs_w}(x_1).c_{bs_w}(x_2).\overline{c_{sa_w}}\langle k, w\rangle \end{aligned}$$

Now let $M[]$ be the context obtained from M by replacing the $\prod_{j=1}^{m} Test_w(w, j)$ sub-process by a hole. We define

$$\begin{aligned} P_1 &= \prod_{j=1}^{m} testbit_{wj}(k, w, x, y).[b_{wj} = x]\overline{t_{w+1j}}\\ P_2 &= \prod_{j=1}^{m} testbit_{wj}(k, w, x, y).[b_{(w+n)j} = y]\overline{t_{w+1j}} \end{aligned}$$

P_1 contains all tests $Test_w(w, j)$ with $1, b_{wj}$ hard-coded (that is the left-choice of the OT server). Then $\nu(M|S(1, b_w))$ is may-equivalent to $\nu(M[P_1]|S(1, b_w))$ (the latter has the values transmitted by the OT hard-coded) which in turn is may-equivalent to $\nu(M[P_1]|S(d, d))$ (we replaced OT's output by a dummy message since it is hard-coded in $M[P_1]$). We have the similar derivation for $M[P_2]$, so from (11.2) we get

$$\begin{aligned} K' &\approx \nu(M[P_1]|S(d, d)) +_{0.5} \nu(M[P_2]|S(d, d))\\ &= C[P_1] +_{0.5} C[P_2] \end{aligned}$$

where $C[] = \nu(M[]|S(d, d))$.

K' and I'_{w+1} differ only in the $Test_w$ sub-processes. Since I_{w+1} doesn't use the output of the OT for the w-th pair at all, we can show that

$$\begin{aligned} I'_{w+1} &\approx C[Q] \quad \text{where}\\ Q &= (\prod_{j=1}^{m} Test_{w+1}(w, j)) \mid (!\overline{guess_w} +_{0.5} 0)\\ &= \prod_{j=1}^{m} testbit_{w,j}(k, w, x, y).([x = b_{wj}][y = b_{(w+n)j}]\overline{t_{w+1j}} \mid guess_w(x).\overline{t_{w+1j}}) \mid\\ &\quad (!\overline{guess_w} +_{0.5} 0) \end{aligned}$$

So we finally have to show that

$$C[P_1] +_{0.5} C[P_2] \sqsubseteq C[Q]$$

We start by showing that $P_1 +_{0.5} P_2 \sqsubseteq Q$. P_1, P_2 can only perform $\overline{t_{w+1j}}$ actions. For a fixed j the probability of performing $\overline{t_{w+1j}}$ depends on the value that is passed (by a test O) through channel $testbit_{wj}$. If it passes $x = b_{wj}$ and $y = b_{(w+n)j}$ then the probability is 1, if only one of the two is passed then the probability is $1/2$ and if none is passed it is 0.

On the other hand Q has at least one half probability of performing $\overline{t_{w+1j}}$ since $guess_w$ is activated with probability one half. Moreover if $x = b_{wj}$ and $y = b_{(w+n)j}$ then both tests of Q succeed and the probability of producing the action is 1. Thus in all cases Q performs the actions with higher probability than $P_1 +_{0.5} P_2$ so we have $P_1 +_{0.5} P_2 \sqsubseteq Q$. Then since $\sqsubseteq$ is a precongruence we have $C[P_1 +_{0.5} P_2] \sqsubseteq C[Q]$ and from the distributivity of $+_p$ we get

$$C[P_1] +_{0.5} C[P_2] \approx C[P_1 +_{0.5} P_2] \sqsubseteq C[Q]$$

which implies $I_w \sqsubseteq I_{w+1}$.

We can finish the proof by induction on w. □

The crucial part in the above proof is the use of the distributivity of the probabilistic sum. This property allowed us to focus on small sub-processes to prove that $P_1 +_{0.5} P_2 \sqsubseteq Q$, and then obtain the same relation after applying the context. This shows the usefulness of the distributivity property as a technical means to prove security properties.

11.4 Related Work

Security protocols have been extensively studied during the last decade and many formal methods have been proposed for their analysis. However, the vast majority of these methods refer to nondeterministic protocols and are not suitable for the probabilistic setting, since they do not allow to model random choices. One exception is the work of Aldini and Gorrieri ([AG02]), where they use a probabilistic process algebra to analyze fairness in a non-repudiation protocol. Their work is close to ours in spirit, although technically it is quite different. In particular, we base our analysis on a notion of testing while theirs is based on a notion of bisimulation.

With respect to the application, the results the most related to ours come from Norman and Shmatikov ([NS03], [NS05]), who use probabilistic model checking to study fairness in two probabilistic protocols, including the Partial Exchange Protocol. In particular, in [NS05] they model the PSE using PRISM, a probabilistic model checker. Their treatment however is very different from ours: their model describes only the "correct" behavior for both A and B, as specified by the protocol. B's ability to cheat is limited to prematurely stopping the execution, so attacks in which B deviates completely from the protocol are not taken into account. Having a simplified model is important in model checking since it helps overcoming the search state explosion problem, thus making the verification feasible.

The results in [NS05] show that with probability one B can gain a one bit advantage, that is he can get all m bits of a pair of A by revealing only $m - 1$

bits of his. This is achieved simply by stopping the execution after receiving the last bit from A. Moreover a method of overcoming the problem is proposed, which gives this advantage to A or B, each with probability one half. Is is worth noting that this is a very weak form of attack and could be considered as negligible, since A can compute the last bit very easily by trying both 0 and 1. Besides a one bit advantage will always exist in contract signing protocols, simply because synchronous communication is not feasible.

In our approach, by modeling an adversary as an arbitrary CCS_σ process we allow him to perform a vast range of attacks including sending messages, performing calculations, monitoring public channels etc. Our analysis shows not only that a one bit attack is possible, but more important that no attack to obtain an advantage of two or more bits exists with non-negligible probability. Moreover our method has the advantage of being easily extensible. For example, treating more sessions, even an infinite number of ones, can be done by putting many copies of the processes in parallel.

Of course, the major advantage of the model checking approach, with respect to ours, is that it can be totally automated.

Bibliography

[AG99] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*, 148(1):1–70, 10 January 1999.

[AG02] Alessandro Aldini and Roberto Gorrieri. Security analysis of a probabilistic non-repudiation protocol. In Holger Hermanns and Roberto Segala, editors, *Process Algebra and Probabilist Methods*, volume 2399 of *Lecture Notes in Computer Science*, page 17, Heidelberg, 2002. Springer.

[And02] Suzana Andova. *Probabilistic process algebra*. PhD thesis, Technische Universiteit Eindhoven, 2002.

[Bil95] Patrick Billingsley. *Probability and Measure*. Wiley, New York, third edition, 1995.

[BP05] Mohit Bhargava and Catuscia Palamidessi. Probabilistic anonymity. In Martín Abadi and Luca de Alfaro, editors, *Proceedings of CONCUR*, volume 3653 of *Lecture Notes in Computer Science*, pages 171–185. Springer, 2005.

[BS01] Emanuele Bandini and Roberto Segala. Axiomatizations for probabilistic bisimulation. In *Proceedings of the 28th International Colloquium on Automata, Languages and Programming*, volume 2076 of *Lecture Notes in Computer Science*, pages 370–381. Springer, 2001.

[Cay89] Arthur Cayley. A theorem on trees. *Quart. J. Math.*, 23:376–378, 1889.

[CC05] Tom Chothia and Konstantinos Chatzikokolakis. A survey of anonymous peer-to-peer file-sharing. In *Proceedings of the IFIP International Symposium on Network-Centric Ubiquitous Systems (NCUS 2005)*, volume 3823 of *Lecture Notes in Computer Science*, pages 744–755. Springer, 2005.

[CCK+06a] Ran Canetti, Ling Cheung, Dilsun Kaynar, Moses Liskov, Nancy Lynch, Olivier Pereira, and Roberto Segala. Task-structured probabilistic i/o automata. In *Proceedings the 8th International Workshop on Discrete Event Systems (WODES'06)*, Ann Arbor, Michigan, 2006.

[CCK+06b] Ran Canetti, Ling Cheung, Dilsun Kirli Kaynar, Moses Liskov, Nancy A. Lynch, Olivier Pereira, and Roberto Segala. Time-bounded task-PIOAs: A framework for analyzing security protocols. In Shlomi Dolev, editor, *Proceedings of the 20th International Symposium in Distributed Computing (DISC '06)*, volume 4167 of *Lecture Notes in Computer Science*, pages 238–253. Springer, 2006.

[Cha81] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 4(2), February 1981.

[Cha88] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1:65–75, 1988.

[Cha07] Konstantinos Chatzikokolakis, 2007. Protype software developed for the thesis. http://www.lix.polytechnique.fr/~kostas/software.html.

[CHM01] David Clark, Sebastian Hunt, and Pasquale Malacaria. Quantitative analysis of the leakage of confidential data. In *Proc. of QAPL 2001*, volume 59 (3) of *Electr. Notes Theor. Comput. Sci*, pages 238–251. Elsevier Science B.V., 2001.

[CHM05] David Clark, Sebastian Hunt, and Pasquale Malacaria. Quantified interference for a while language. In *Proc. of QAPL 2004*, volume 112 of *Electr. Notes Theor. Comput. Sci*, pages 149–166. Elsevier Science B.V., 2005.

[Cho07] Tom Chothia. How anonymity can fail because of the scheduler. Demonstration of the scheduler problem of the Dining Cryptographers in Java. http://homepages.cwi.nl/~chothia/DCschedular/, 2007.

[CM07] Konstantinos Chatzikokolakis and Keye Martin. A monotonicity principle for information theory. Submitted for publication, 2007.

[CP05a] Konstantinos Chatzikokolakis and Catuscia Palamidessi. A framework for analyzing probabilistic protocols and its application to the partial secrets exchange. In *Proceedings of the Symp. on Trustworthy Global Computing*, volume 3705 of *Lecture Notes in Computer Science*, pages 146–162. Springer, 2005.

[CP05b] Konstantinos Chatzikokolakis and Catuscia Palamidessi. A framework for analyzing probabilistic protocols and its application to the partial secrets exchange. *Theoretical Computer Science*, 2005. To appear.

[CP06a] Konstantinos Chatzikokolakis and Catuscia Palamidessi. Probable innocence revisited. In Theodosis Dimitrakos, Fabio Martinelli, Peter Y. A. Ryan, and Steve A. Schneider, editors, *Third International Workshop on Formal Aspects in Security and Trust (FAST*

2005), Revised Selected Papers, volume 3866 of *Lecture Notes in Computer Science*, pages 142–157. Springer, 2006.

[CP06b] Konstantinos Chatzikokolakis and Catuscia Palamidessi. Probable innocence revisited. *Theoretical Computer Science*, 367(1-2):123–138, 2006.

[CP07] Konstantinos Chatzikokolakis and Catuscia Palamidessi. Making random choices invisible to the scheduler. In Luís Caires and Vasco Thudichum Vasconcelos, editors, *CONCUR*, volume 4703 of *Lecture Notes in Computer Science*, pages 42–58. Springer, 2007.

[CPP06] Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Prakash Panangaden. Anonymity protocols as noisy channels. In *Postproceedings of the Symp. on Trustworthy Global Computing*, Lecture Notes in Computer Science. Springer, 2006. To appear.

[CPP07a] Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Prakash Panangaden. Anonymity protocols as noisy channels. *Information and Computation*, 2007. To appear.

[CPP07b] Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Prakash Panangaden. Probability of error in information-hiding protocols. In *Postproceedings of CSF'07*, Lecture Notes in Computer Science. Springer, 2007. To appear.

[CSWH00] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability*, volume 2009 of *Lecture Notes in Computer Science*, pages 44–66. Springer, 2000.

[CT91] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., 1991.

[dAHJ01] Luca de Alfaro, Thomas A. Henzinger, and Ranjit Jhala. Compositional methods for probabilistic systems. In Kim Guldstrand Larsen and Mogens Nielsen, editors, *Proceedings of the 12th International Conference on Concurrency Theory (CONCUR 2001)*, volume 2154 of *Lecture Notes in Computer Science*. Springer, 2001.

[DC07] Roberto Di Cosmo. On privacy and anonymity in electronic and non electronic voting: the ballot-as-signature attack. Available online at http://hal.archives-ouvertes.fr/hal-00142440, 2007.

[Dee89] Steve Deering. Host extensions for IP multicasting. RFC 1112, August 1989.

[DKR06] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Verifying properties of electronic voting protocols. In *Proceedings of the IAVoSS Workshop On Trustworthy Elections (WOTE'06)*, pages 45–52. Cambridge, UK, 2006.

[DPP05] Yuxin Deng, Catuscia Palamidessi, and Jun Pang. Compositional reasoning for probabilistic finite-state behaviors. In Aart Middeldorp, Vincent van Oostrom, Femke van Raamsdonk, and Roel C. de Vrijer, editors, *Processes, Terms and Cycles: Steps on the Road to Infinity*, volume 3838 of *Lecture Notes in Computer Science*, pages 309–337. Springer, 2005.

[DPP06] Yuxin Deng, Catuscia Palamidessi, and Jun Pang. Weak probabilistic anonymity. In *Proceedings of the 3rd International Workshop on Security Issues in Concurrency (SecCo)*, Electronic Notes in Theoretical Computer Science. Elsevier Science B.V., 2006. To appear.

[DPW06] Yuxin Deng, Jun Pang, and Peng Wu. Measuring anonymity with relative entropy. In *Proceedings of the 4th International Workshop on Formal Aspects in Security and Trust (FAST)*, Lecture Notes in Computer Science. Springer, 2006. To appear.

[DSCP02] Claudia Díaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards measuring anonymity. In Roger Dingledine and Paul F. Syverson, editors, *Proceedings of the workshop on Privacy Enhancing Technologies (PET) 2002*, volume 2482 of *Lecture Notes in Computer Science*, pages 54–68. Springer, 2002.

[EGL85] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, 1985.

[Gra90] R. M. Gray. *Entropy and Information Theory*. Springer-Verlag, New York, 1990.

[Gra91] J. W. Gray, III. Toward a mathematical foundation for information flow security. In *Proceedings of the 1991 IEEE Computer Society Symposium on Research in Security and Privacy (SSP '91)*, pages 21–35, Washington - Brussels - Tokyo, May 1991. IEEE.

[GSB02] Mesut Gunes, Udo Sorges, and Imed Bouazzi. Ara – the ant-colony based routing algorithm for manets. In *Proceedings of the International Workshop on Ad Hoc Networking (IWAHN 2002)*, Vancouver, August 2002.

[GvRS07] Flavio D. Garcia, Peter van Rossum, and Ana Sokolova. Probabilistic anonymity and admissible schedulers, 2007. arXiv:0706.1019v1.

[HJ89] H. Hansson and B. Jonsson. A framework for reasoning about time and reliability. In *Proceedings of the 10th IEEE Symposium on Real-Time Systems*, pages 102–111, Santa Monica, California, USA, 1989. IEEE Computer Society Press.

[HJ90] H. Hansson and B. Jonsson. A calculus for communicating systems with time and probabitilies. In *Proceedings of the Real-Time Systems Symposium - 1990*, pages 278–287, Lake Buena Vista, Florida, USA, 1990. IEEE Computer Society Press.

[HO03] Joseph Y. Halpern and Kevin R. O'Neill. Anonymity and information hiding in multiagent systems. In *Proc. of the 16th IEEE Computer Security Foundations Workshop*, pages 75–88, 2003.

[HO05] Joseph Y. Halpern and Kevin R. O'Neill. Anonymity and information hiding in multiagent systems. *Journal of Computer Security*, 13(3):483–512, 2005.

[HP00] Oltea Mihaela Herescu and Catuscia Palamidessi. Probabilistic asynchronous π-calculus. In Jerzy Tiuryn, editor, *Proceedings of FOSSACS 2000 (Part of ETAPS 2000)*, volume 1784 of *Lecture Notes in Computer Science*, pages 146–160. Springer, 2000.

[HR07] M.E. Hellman and J. Raviv. Probability of error, equivocation, and the chernoff bound. *IEEE Trans. on Information Theory*, IT–16:368–372, 2007.

[HS04] Dominic Hughes and Vitaly Shmatikov. Information hiding, anonymity and privacy: a modular approach. *Journal of Computer Security*, 12(1):3–36, 2004.

[JLY01] Bengt Jonsson, Kim G. Larsen, and Wang Yi. Probabilistic extensions of process algebras. In Jan A. Bergstra, Alban Ponse, and Scott A. Smolka, editors, *Handbook of Process Algebra*, chapter 11, pages 685–710. Elsevier, 2001.

[KNP04] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. PRISM 2.0: A tool for probabilistic model checking. In *Proceedings of the First International Conference on Quantitative Evaluation of Systems (QEST) 2004*, pages 322–323. IEEE Computer Society, 2004.

[Low02] Gavin Lowe. Quantifying information flow. In *Proc. of CSFW 2002*, pages 18–31. IEEE Computer Society Press, 2002.

[LS91] Kim G. Larsen and Arne Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, September 1991.

[Mar07] Keye Martin. Topology in information theory in topology. *Theoretical Computer Science*, 2007. To appear.

[Mau00] Ueli M. Maurer. Authentication theory and hypothesis testing. *IEEE Transactions on Information Theory*, 46(4):1350–1356, 2000.

[McL90] John McLean. Security models and information flow. In *IEEE Symposium on Security and Privacy*, pages 180–189, 1990.

[Mil89] R. Milner. *Communication and Concurrency*. International Series in Computer Science. Prentice Hall, 1989.

[MMA06] Keye Martin, Ira S. Moskowitz, and Gerard Allwein. Algebraic information theory for binary channels. *Electr. Notes Theor. Comput. Sci.*, 158:289–306, 2006.

[MNCM03] Ira S. Moskowitz, Richard E. Newman, Daniel P. Crepeau, and Allen R. Miller. Covert channels and anonymizing networks. In Sushil Jajodia, Pierangela Samarati, and Paul F. Syverson, editors, *WPES*, pages 79–88. ACM, 2003.

[MNS03] Ira S. Moskowitz, Richard E. Newman, and Paul F. Syverson. Quasi-anonymous channels. In *IASTED CNIS*, pages 126–131, 2003.

[MOW04] Michael Mislove, Joël Ouaknine, and James Worrell. Axioms for probability and nondeterminism. In F. Corradini and U. Nestmann, editors, *Proc. of the 10th Int. Wksh. on Expressiveness in Concurrency (EXPRESS '03)*, volume 96 of *Electronic Notes in Theoretical Computer Science*, pages 7–28. Elsevier, 2004.

[NH84] Rocco De Nicola and Matthew C. B. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34(1-2):83–133, 1984.

[NPS99] Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, pages 129–139. ACM Press, 1999.

[NS03] Gethin Norman and Vitaly Shmatikov. Analysis of probabilistic contract signing. In A. Abdallah, P. Ryan, and S. Schneider, editors, *Proc. BCS-FACS Formal Aspects of Security (FASec'02)*, volume 2629 of *LNCS*, pages 81–96. Springer, 2003.

[NS05] Gethin Norman and Vitaly Shmatikov. Analysis of probabilistic contract signing. *Formal Aspects of Computing (to appear)*, 2005.

[Par81] D. Park. Concurrency and automata on infinite sequences. In *Proceedings of the Fifth GI-Conference on Theoretical Computer Science*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183, New York, 1981. Springer-Verlag.

[PH05] Catuscia Palamidessi and Oltea M. Herescu. A randomized encoding of the π-calculus with mixed choice. *Theoretical Computer Science*, 335(2-3):373–404, 2005.

[PHW04] Alessandra Di Pierro, Chris Hankin, and Herbert Wiklicky. Approximate non-interference. *Journal of Computer Security*, 12(1):37–82, 2004.

[PHW05] Alessandra Di Pierro, Chris Hankin, and Herbert Wiklicky. Measuring the confinement of probabilistic systems. *Theoretical Computer Science*, 340(1):3–56, 2005.

[PK04] Andreas Pfitzmann and Marit Köhntopp. Anonymity, unobservability, and pseudonymity: A proposal for terminology, draft v0.21, September 2004.

[Rab81] Michael O. Rabin. How to exchange secrets by oblivious transfer. *Technical Memo TR-81, Aiken Computation Laboratory, Harvard University*, 1981.

[Rén66] Alfred Rényi. On the amount of missing information and the Neyman-Pearson lemma. In *Festschriftf for J. Neyman*, pages 281–288. Wiley, New York, 1966.

[Riv06] Ronald L. Rivest. The threeballot voting system. Technical report, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 2006.

[Roy88] H. L. Royden. *Real Analysis*. Macmillan Publishing Company, New York, third edition, 1988.

[RR98] Michael K. Reiter and Aviel D. Rubin. Crowds: anonymity for Web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[RS01] Peter Y. Ryan and Steve Schneider. *Modelling and Analysis of Security Protocols*. Addison-Wesley, 2001.

[SD02] Andrei Serjantov and George Danezis. Towards an information theoretic metric for anonymity. In Roger Dingledine and Paul F. Syverson, editors, *Proceedings of the workshop on Privacy Enhancing Technologies (PET) 2002*, volume 2482 of *Lecture Notes in Computer Science*, pages 41–53. Springer, 2002.

[Seg95] Roberto Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, June 1995. Available as Technical Report MIT/LCS/TR-676.

[SGR97] P.F. Syverson, D.M. Goldschlag, and M.G. Reed. Anonymous connections and onion routing. In *IEEE Symposium on Security and Privacy*, pages 44–54, Oakland, California, 1997.

[Sha93] C. E. Shannon. Some geometrical results in channel capacity. In *Collected Papers of C.E. Shannon*, pages 259–265. IEEE Press, 1993.

[Shm02] Vitaly Shmatikov. Probabilistic analysis of anonymity. In *15th IEEE Computer Security Foundations Workshop (CSFW)*, pages 119–128, 2002.

[Shm04] V. Shmatikov. Probabilistic model checking of an anonymity system. *Journal of Computer Security*, 12(3/4):355–377, 2004.

[SL95] Roberto Segala and Nancy Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–273, 1995. An extended abstract appeared in *Proceedings of CONCUR '94*, LNCS 836: 481-496.

[SS96] Steve Schneider and Abraham Sidiropoulos. CSP and anonymity. In *Proc. of the European Symposium on Research in Computer Security (ESORICS)*, volume 1146 of *Lecture Notes in Computer Science*, pages 198–218. Springer, 1996.

[SS99] Paul F. Syverson and Stuart G. Stubblebine. Group principals and the formalization of anonymity. In *World Congress on Formal Methods (1)*, pages 814–833, 1999.

[SS00] Andrei Sabelfeld and David Sands. Probabilistic noninterference for multi-threaded programs. In *Proc. of CSFW 2000*, pages 200–214. IEEE Computer Society Press, 2000.

[SV04] A. Sokolova and E.P. de Vink. Probabilistic automata: system types, parallel composition and comparison. In C. Baier, B.R. Haverkort, H. Hermanns, J.-P. Katoen, and M. Siegle, editors, *Validation of Stochastic Systems: A Guide to Current Research*, volume 2925 of *Lecture Notes in Computer Science*, pages 1–43. Springer, 2004.

[SV06] Nandakishore Santhi and Alexander Vardy. On an improvement over Rényi's equivocation bound, 2006. Presented at the 44-th Annual Allerton Conference on Communication, Control, and Computing, September 2006. Available at http://arxiv.org/abs/cs/0608087.

[Var85] Moshe Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science*, pages 327–338, Portland, Oregon, 1985. IEEE Computer Society Press.

[WALS02] M. Wright, M. Adler, B. Levine, and C. Shields. An analysis of the degradation of anonymous protocols. In *ISOC Network and Distributed System Security Symposium (NDSS)*, 2002.

[YL92] Wang Yi and Kim G. Larsen. Testing probabilistic and nondeterministic processes. In *Proceedings of the 12th IFIP International Symposium on Protocol Specification, Testing and Verification*, Florida, USA, 1992. North Holland.

[ZB05] Ye Zhu and Riccardo Bettati. Anonymity vs. information leakage in anonymity systems. In *Proc. of ICDCS*, pages 514–524. IEEE Computer Society, 2005.