TEL-AVIV UNIVERSITY

RAYMOND AND BEVERLY SACKLER

FACULTY OF EXACT SCIENCES

SCHOOL OF COMPUTER SCIENCE

Hide and Seek:
Provable Anonymity in Computer Networks

Thesis submitted in partial fulfillment of the requirements for the M.Sc. degree in the

School of Computer Science, Tel-Aviv University

by

Ron Berman

The research work for this thesis has been carried out at Tel-Aviv University

under the supervision of Prof. Amos Fiat

December 2003

Abstract

Anonymous communication methods have been under heavy research in the last years. Specifically, with the advent of peer to peer networks, anonymity is grasped as a desired property of any well designed system for exchanging information between parties. Previous work dealing with anonymity and privacy is mostly application driven and intuitively based, paying more attention to implementation details rather than security analysis. This work focuses on communication protocols that exhibit unlinkability, which is one of the possible types of anonymity. As a preliminary step we use information theory to devise a set of equivalent definitions for unlinkability. These definitions motivate us towards the goal of proving the unlinkability of an efficient variant of David Chaum's classic protocol for anonymous data communication. The case of adaptive adversaries having both prior and no-prior information is studied. We conclude with the analysis of two possible applications of the protocol: anonymous web surfing and anonymous file retrieval from a small closed network.

Acknowledgments

I would like to express my gratitude to my supervisor Prof. Amos Fiat and to Dr. Amnon Ta-Shma for their invaluable help during the work on this research project. Their determination at constantly improving the results achieved has led this work to truly new regions.

In addition, I would like to acknowledge Prof. Fiat's recurring efforts to convince me to go sailing with him, as slim as my chances of acceptance were, and still are.

Contents

1 Introduction **1**

1.1 Motivation 2

1.2 Types of Anonymity 2

1.2.1 Chaumian Mix-Based Systems 4

1.2.2 Onion Routing 5

1.2.3 Attack Model and Attacks 6

1.2.4 Layer Mixing 7

1.3 Protocol Efficiency 7

1.4 Structure 8

1.5 Related Work 9

2 Adversary Models **11**

2.1 Passive Adversary 11

2.2 Adaptive Adversary 12

2.3 Malicious Adversary 12

2.4 Our Network Attack Model 13

3 Defining Unlinkability **14**

3.1 Equivalence of Definitions 16

3.2 Unlinkable Communication Protocols 18

4 An Anonymous Communication Protocol **20**

4.1 The Asynchronous Setting 23

5 Unlinkability Proof **25**

5.1 Proof Sketch 25

5.2 Definition and Properties of Obscurant Networks 26

5.3 Constructing Shallow Networks That Obscure Inputs 27
5.4 Finding Obscurity Within Executions of Our Protocol 29
5.4.1 Network Embedding 30
5.4.2 An Embedding Strategy 31
5.5 Unlinkability Without Prior Information 34
5.6 Unlinkability With Prior Information 36

6 Applications **41**
6.1 Real Adversaries 41
6.2 Anonymous Web Surfing 42
6.3 Privacy Preserving Data Exchange 42

7 Conclusions **45**
7.1 Future Work 46

Bibliography **48**

A Simple Statistical Analysis of Traffic **51**

B Information Theory and Its Usefulness **55**
B.1 Notations 55
B.2 Some Information Theory 55
B.3 Some More Motivation 57

Chapter 1

Introduction

Anonymous communication has been widely used in history in "real life" situations. Examples include the ability of a person to anonymously notify the police of an ongoing crime, the possibility to browse sensitive medical information such as the list of Alzheimer's Disease symptoms etc. When considering real-life situations outside the communications or computer-science world, the term "anonymity" is usually referred to as the property of being unidentifiable or indistinguishable when being put together with a group of people. The case for "real-life" communication holds still in the computer networking world, where different entities communicate using various means, and wish to keep their identities private.

A reasonable assumption is that the following simple methods may allow us to achieve anonymity:

- Introducing ambiguity into the communications pattern. *I.e.*, making our communications less specific and explicit.
- Joining a large group of entities and hiding our own unique communications pattern within a larger more random one.
- Letting others do the job for us, using a mediator, assuming that connecting the mediator with us is a harder task.
- Using encryption to hide the content of our communications.

This short list is naturally not complete, and as a matter of fact, not all methods or combination of methods provide the needed security. Following is an introduction to the problems we are faced with when designing communication systems allowing anonymity. As surprising as it may be, even the definition of anonymity is unclear in many situations, thus resulting in incomplete work, and more importantly, systems with unproven security properties.

1.1 Motivation

The first question arising from the above discussion regards the need for a rigorous analysis of methods for anonymous communication. If we cannot prove how hard (or actually easy) it is to single out a communications pattern from all the noise and how long it will take, then our claims have no practical need.

Appendix A describes the details of an experiment that consists of a computer network where all the nodes are divided into two types, "bad" and "good". All the content is also divided into the same two types. We assume that a good node always accesses good data, while bad nodes always access bad data. We also assume that all bad data is under control of an adversary, and that the network provider supplies some information about the users in each time step.

The appendix also describes the experiment's results and analysis. We can see that as low as $\Theta(\frac{N}{\sqrt{\epsilon}})$ experiments are enough to identify a bad node out of N nodes with ϵ error. The results clarify how fast and easy it is to distinguish "bad" guys from "good" guys[1] using simple passive methods.

1.2 Types of Anonymity

Taking a look at a dictionary's definition to the word "Anonymous" [MW] it is clear that translating the meaning into technical terms isn't trivial:

- not named or identified <an anonymous author> <they wish to remain anonymous>.
- of unknown authorship or origin <an anonymous tip>.
- lacking individuality, distinction, or recognizability <the anonymous faces in the crowd> <the gray anonymous streets – William Styron>.

The goal of anonymity is classically subdivided into smaller more specific objectives. In [PW87] anonymity is divided into three parts — sender anonymity, recipient anonymity and unlinkability of sender and receiver. Recent work found in [PK00] constitutes an attempt to put some order into the various interpretations of the term "anonymity" and create commonly accepted terms:

- An anonymity set of a certain item is the group of items that serve as an obscuring background for a certain property of the designated item. If, for example, the nodes in an

[1] The notion of "good" and "bad" is clearly a matter of taste.

anonymity set of a certain node can only send messages, then this designated node can only achieve sender anonymity.

- Unlinkability of two items means that these two items are not more related than before the system was run, given the posteriori knowledge. As unlinkability of two items depends on the relationship possible between those items and the relationship with the rest of the anonymity set, we receive the following subdivision:

  - Sender unlinkability means it is impossible to link a message with the identity of its sender.

  - Receiver unlinkability is the same as the above only the linkage is with the receiver of a message.

  - Relationship unlinkability means it is impossible to link a sender and a receiver as a transaction couple. This term is usually referred to as just "unlinkability".

- Unobservability is the state of items being indistinguishable from any item at all. When a query process achieves unobservability, *i.e.*, the difference between content of messages can't be determined we say we achieved "Query anonymity". The correct precise term, when referring to [PK00] would be message unobservability.

All of the mentioned types have been under research for quite some time. Section 1.5 serves as a brief introduction to previous work on these subjects.

The main part of this work deals with unlinkability. The subject is covered from the basics by showing a few equivalent definitions for unlinkability in Chapter 3.

A simple approach to unlinkability is to have every node in a communications network broadcast every data item it holds to all other nodes continuously. A node interested in a certain item just listens to the network continuously until the item sought for is broadcast. The source of the data is hidden by nullifying the source address in broadcast data. This requires that a broadcast mechanism will be available in the network without the requirement of a source address being attached to each message[2].

The broadcast solution is clearly not practical or efficient. It introduces major delays in service availability — each node may wait a long time until the required item is broadcast by someone else.

The majority of previous works has dealt with two types of communication operations: Sending a unidirectional message from a sender to a receiver (as in an e-mail message) or

[2]Several types of current technology's LANs posses this ability. Ethernet is just one example.

retrieving information from a known address by a retriever (as in browsing a web page). Another contribution of this work is a technique called "donor anonymity". In this case a retriever initiates a transaction where he retrieves information from a network without being able to identify the source of the answer. This differs from the case of receiver anonymity by the fact that the transaction is a two way operation. The retriever once serves as a sender, and then as a receiver. Donor anonymity preserves both sender and receiver anonymity *throughout* the entire retrieval operation. Section 6.3 illustrates situations in which we desire this property, and introduces a simple construction allowing it.

1.2.1 Chaumian Mix-Based Systems

In 1980 David Chaum introduced a novel approach to communications anonymity [Cha81]. Up until today Chaum's ideas serve as the basis for most practical implementations of communication systems featuring anonymity. Chaum's system is built by two fundamental building blocks.

A mix[3] is a mediating computer system between a sender of a message and its receiver. The purpose of a mix is to introduce ambiguity into traffic patterns, thus disallowing the ability to trace the path of a message from its sender to its receiver.

The mix system accepts batches of messages from senders, each one with its desired target address, and then forwards each message to its destination. The order of forwarding is arbitrary, *e.g.*, the lexicographic ordering of messages. By performing such an operation the mix achieves a couple of goals:

- The receiver cannot know the origin of the message, as all messages appear to be coming from the mix.
- Using encryption in the sender→mix part of the message path, combined with the different order of outgoing messages from the mix, an eavesdropper is not able to associate an incoming message into the mix with an outgoing message. This property makes it difficult for the adversary to link the origin of the message with its destination, thus achieving unlinkability.

Figure 1.1 illustrates the operation of the original mix as described in [Cha81].

An additional requirement imposed on a mix is to wait until at least two messages are buffered within it before forwarding them to their destinations. This requirement stems from the

[3] a.k.a. a Chaumian mix.

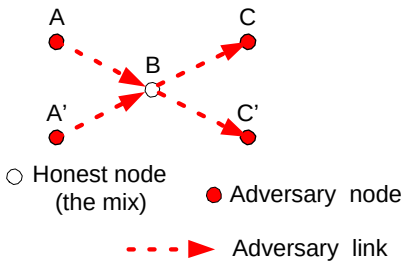

Figure 1.1: *Chaumian mixes: traffic analysis ambiguity is achieved by honest nodes that collect multiple incoming messages before sending out the "peeled" messages to the next stage. White vertices are honest, while filled-in vertices are under adversary control. All links are under adversary control.*

assumption that all communication in the system is under surveillance, and as such, ambiguity in traffic analysis can only be introduced inside a trusted mix. This assumption imposes harsh requirements on timing, choice of mixes and especially on the need for a large amount of activity in the network.

It is a fact that the above assumption also affects the anonymity strength of a mix. Traffic analysis in a traditional mix system is only avoided by obscuring the paths of honest messages in the system, and not ones created by the adversary. This results in high sensitivity to the amount of honest traffic needed to achieve unlinkability.

Section 1.5 describes some mix-based solutions all of which require large quantities of honest messages in the system. It is needless to say that these solutions are inefficient considering the communications volume they impose.

1.2.2 Onion Routing

The second feature of Chaum's anonymity preserving system is the use of public key cryptography in performing routing of messages through a cascade of mixes. The usage of message encryption between any two points in the system inhibits an adversary from following message paths based on their content. This effectively reduces the problem to dealing with traffic analysis.

When using a cascade of mixes the sender randomly chooses a list of mixes through which the message is to be routed to its final destination. The message is then encrypted multiple times, each time a new encryption "peel" is added. Each additional encryption layer encompasses within it the needed information for a specific mix in the cascade to route the message to the next mix in the list. Figure 4.1 supplies details of cascade mixing and demonstrates how it is used to route messages in a system. This type of message routing is nicknamed "Onion Routing" [GRS96].

The original system devised by Chaum allowed return messages to go back to an anonymous

return address. Practically, what this meant was that the message was constructed of two parts, the regular forward message "onion", and another separate "onion" containing the needed routing information and encryption keys that allowed backward return of answers. The second onion was transferred as is in the forward path from mix to mix, and used only in the return path.

1.2.3 Attack Model and Attacks

Many attempts were made to analyze the strength of the "onion routing" process. Chaum's model assumed that all communication lines, as well as some mix nodes are under control of an adversary. The classical description lacks a security analysis, and following work has proven the system to be insecure and highly sensitive to strong adversaries.

The analysis process usually consists of identifying a single attack, showing a case under which current systems give in. A solution then follows, which is tailored to fix the specific type of attack shown[4]. It is a very common case, though, that these solutions do not withstand new types of attacks which appear shortly after, and the process recurs yet again. It is hoped that the ideas and methods presented in this paper will form a strong basis for future rigorous analysis of privacy preserving systems, putting an end to the hack and attack circle. Following are a few examples of attack classes described in the literature.

Timing attacks are possible when the routes of messages in the system take different times, or when the timings of message arrival and departure give indication about its route. Suppose, for example, that a message had left a node and had come back to it after a time in which only 4 hops at maximum had been possible. If the network is not fully connected, the adversary knows a limit on the possible nodes to which the message was destined and came back from.

Mix flood attacks take advantage of the fact that Chaumian-mixes store a number of messages before forwarding each one to its destination. When the adversary floods a mix with her messages, making sure they compose the large majority of messages it stores, she forces a mix to output its stored messages earlier making her task of identifying the path of each foreign message easier.

Message tagging attacks allow an adversary the possibility of issuing special messages, or treating messages on their way through the network so as to follow their path and identify them at the destination or near it. This attack is extremely useful in case a network with low degree of connections between nodes is used, and the network map is unknown to the adversary, as it allows the mapping of available network connections.

[4]It is common to refer to such on-the-fly solutions as hacks.

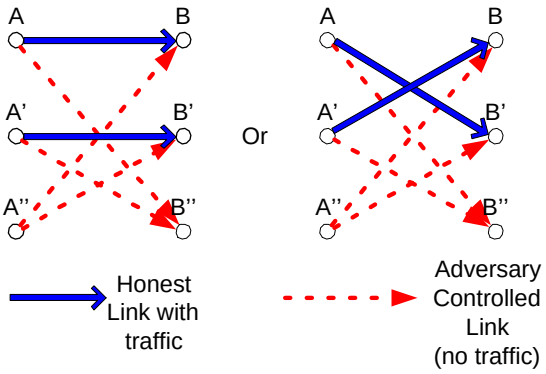

Figure 1.2: *Layer mixing: ambiguity is introduced because none of the four nodes* A, A', B, B' *and none of the four links* $A \mapsto B$, $A \mapsto B'$, $A' \mapsto B$ *and* $A' \mapsto B'$ *are under adversary control. The adversary is unsure whether* A *communicated with* B *or with* B'*. Notice that although the adversary has not seen any communication in this stage, the adversary knows that* A *and* B *together communicated with* A' *and* B'*, because of links he controls that were not used for communication.*

Section 1.1 gave general details of statistical traffic analysis. Full details and description appear in Appendix A. By applying several assumptions on the behavior of users, it is possible to employ intersection analysis on communicating groups, as well as different types of probabilistic assumptions and analysis.

1.2.4 Layer Mixing

The prime objective of this work is to show that there exists a simple, efficient and practical anonymous communication protocol with provable properties. As it turns out, it is possible to prove that a variant of Chaum's original "onion routing" protocol exhibits certain desirable properties. The *catch*, however, lies within the attack model. In order to prove unlinkability we need to assume that not all communication links are under adversary surveillance all the time. We call the process of messages going through such a network attaining unlinkability on the way — layer mixing.

Layer mixing is basically the idea of paths of two messages being obscured by the fact that an adversary does not listen on the communication links relevant to the passage of each message. Specifically, In order to achieve layer mixing, a structure of 2 source nodes, 2 target nodes and 4 links free of surveillance are needed. Figure 1.2 illustrates the details of the layer mixing idea.

1.3 Protocol Efficiency

Say we have N users in our system, and at each given time at most M users want to send a message. When measuring efficiency one may consider

- *delay*, also known as parallel time, *i.e.*, the number of rounds a protocol takes, or,

- *load*, *i.e.*, the total number of messages transmitted throughout the protocol, or,
- *both*.
- The term *overhead* is used to denote the load divided by M. This is the true overhead (measured in messages) for a protocol, imposed on each participating node.

Rackoff and Simon use delay as a measure of efficiency. They say a protocol is *efficient* if the maximum number of rounds is $O(\text{polylog}(N))$. Indeed, mix-based protocols in general, and Rackoff and Simon's algorithm in particular, require all honest nodes to participate at each round — irrespective of whether they have anything to send or not. Thus, while the number of rounds is small the total number of messages is very high and the system is highly inefficient in terms of message overhead.

In contrast, we define a protocol to be efficient only if both the delay is $O(\text{polylog}(N))$ and the load is at most $O(M \cdot \text{polylog}(N))$. Thus, the load on the system is proportional to its use.

1.4 Structure

The next section contains a survey of related work in the field of anonymous communication protocols, systems, analysis and more.

As this work is based primarily on the mathematical tools of information theory, we give a full definition of our notations, and a short introduction to the area of information theory in Appendix B. We use standard notations and definitions, as appearing in [CT91] and in [NC00].

We describe another short experiment later in the appendix that exhibits the difference and advantages of using the concepts of information theory, compared with standard probability manipulations.

Chapter 2 gives a formal definition of the attack model used in our system, while Chapter 3 rigorously defines what we would like our protocol to achieve, and what an unlinkable protocol is.

We later on describe our protocol in Chapter 4, after which we give construction details of a network where layer mixing is achieved in Section 5.2. Finally, Section 5.5 gives proof that layer mixing works and achieves unlinkability.

As this work is highly connected with the practical world of communications, we present some applications of our protocol, and discuss our conclusions and suggested future work in the last chapters.

1.5 Related Work

Chaum's Dining-Cryptographer networks (DC-Nets) [Cha88] allow both provable sender and receiver anonymity and unlinkability, secure against passive adversaries, as well as some forms of stronger adversaries. The scheme uses shared secret keys, requires secure and reliable broadcast mechanisms and all users need to participate at every message delivery.

Rackoff and Simon [RS93] also describe a simple system based on sorting networks, that is unlinkable against a *passive* adversary that controls all the communication links but *none* of the nodes. They generalize the system to *passive* adversaries that may control some of the nodes using secure multi-party communication. Their system requires all players to participate in each round.

A somewhat different system is Crowds [RR98]. Crowds has a set of intermediate nodes, some of them may be corrupt, but they are not used as mixes. Instead every node takes a probabilistic decision whether to send the message to its final destination, or to forward it to another intermediate node. Crowds supplies a formal analysis of the security it provides, but the security is very mild (it is proportional to the path length). The efficiency and security of a crowd grows as the number of its members grows. The system is highly applicable in peer to peer networks.

Abe ([Abe99]) proposed a mix-based protocol that uses a bulletin board and zero knowledge proofs. The interesting point about Abe's protocol is that it uses networks that have the property that for every given permutation the network's switches can be set so that the network implements the given permutation. In his paper Abe implicitly claims that if one takes a permutation network and uses each switch as a crossover (*i.e.*, each switch with probability half implements identity, and with probability half switches its inputs) then any permutation network perfectly obscures permutations. The claim is however faulty as was shown in [AH01]. The fix includes the requirement that participating mixes somehow coordinate the creation of permutation networks in advance between themselves.

Anonymous Multi Party Computation (AMPC) is introduced in [MP01] as a building block for an anonymous communications subsystem which does not utilize any complex cryptographic primitives. The system is robust against malicious modifications of data through executions of the protocol, and its strength is evaluated under a limited adversary model. The delay and load factors of the system highly depend of the specifics of the strength of the adversary. The system is applicable for achieving either sender anonymity or receiver anonymity, yet it is not clear if it achieves unlinkability.

Protocol	Attack Model: Resources under adversary control	Delay	Load	Simple?	Attacks?
Mix-nets	$O(1)$ fraction of nodes All links	polylog(N)	polylog(N)	Yes	**Yes**
DC-nets	Any fraction of nodes No links	$O(1)$	$O(N^2)$	No	—
[RS93]	$O(1)$ fraction of nodes All links	polylog(N)	$\tilde{O}(N)$	No	—
Crowds	$O(1)$ fraction of nodes No links	Dynamic	Delay	Yes	**Yes**
AMPC	$O(1)$ fraction of nodes No links	$O(N)$	$O(N^2)$	No	—
[BD03]	Any fraction of nodes All links	$O(N)$	$O(N^2)$	No	—
This paper	$O(1)$ fraction of nodes $O(1)$ fraction of links	polylog(N)	polylog(N)	Yes	—

Table 1.1: Unlinkability protocols for a network of size N. Delay is how long it takes for an anonymous message to arrive after it's been initiated. Load is the number of messages actually sent per anonymous message delivered.

An interesting new approach, quite different from Chaumian mixes, is taken in [BD03]. The system used is analogous to public transportation, where people are hard to track as they move along within large groups of people (*e.g.* a bus). The system has true proofs of security. Unfortunately, it also has a rather inefficient time/space tradeoff and thus is not "efficient" in our terminology.

Table 1.1 contains a summarized comparison of the results of this work with previous work.

We finally mention some mix-based implementations. The Anonymizer [Ano] is a single "trusted" mix implementation. Electronic Frontiers' Anonymous Remailer [EFG] is a full implementation of Chaum's ideas, filling in many details missing in Chaum's paper. Onion Routing [SGR97, RSG98] implements these ideas beneath the application level making the underlying communication network seamlessly anonymous.

The Free Haven Project [DFM00] aims to deploy a system for distributed, anonymous, persistent data storage which is robust against powerful adversaries. The project's site [FH] contains a large repository of anonymity related material.

There is a very significant body of work on mix-based system, and a good survey of this massive volume of work is the collection of papers [Fed01].

Chapter 2

Adversary Models

When coming to analyze the security properties of a system, the question of adversary strength always comes into mind. Past work, even when dealing specifically with the subject of anonymous communication, has introduced a great variety of adversary strengths, threat models and attack descriptions as the basic assumptions under which the protocol was analyzed.

The sheer importance of choosing the correct model as the basis of analysis cannot be stressed enough. This work describes three types of general adversaries which seem to be most relevant to the described protocol. The properties of each adversary are described and a comparison is made between models with regard to their harm potential to our system. We later describe the full details of our reference attack model.

2.1 Passive Adversary

A passive adversary, as its name implies, cannot initiate any traffic in the network apart from legitimate traffic. A passive adversary can come into existence in two points of a communications network. The first point is a communication link between any two nodes. In such a location, the adversary is able to determine if there was communication or there wasn't any at a specific point in time. The second location possible for a passive adversary is inside a node, either by taking control of it, or by placing specific "listening bugs" on its entry and exit communication links. In such a case, the adversary is able to follow all links connected with the node, and tell whether there was or wasn't any traffic on any such link at any specific time.

Passive adversaries have the option of recording and understanding message content as they pass through links under their surveillance. Classically, the two tasks are distinguished as "Traffic analysis", where the path of a message is of importance and not its content, and

"Content analysis" where the opposite is the case. Deployment of correctly used cryptographic methods allow the circumvention of content analysis, as described in Chapter 4.

We stress that we allow all types of adversaries to collude amongst themselves and share any information they acquire. This assumption can be reduced to the assumption that the system contains only one adversary which in turn compromises multiple honest nodes or takes control of multiple communication links either for passive or active uses. At a first glance, a passive adversary appears limited in its abilities. The reader is referred to Appendix A where it is shown how a well spread passive adversary is able to identify all communication paths under certain circumstances.

2.2 Adaptive Adversary

Adaptive adversaries embody the functionality of passive adversaries, with the extra ability of inserting new messages into the network. Practically, insertion of messages is made by having a node in the system under control of the adversary, following its instructions.

The way our protocol is designed allows it to withstand an adaptive adversary due to the use of encryption of messages passed on each link, as described in Chapter 4.

An interesting point, though, is to look at one specific attack an adaptive adversary can carry out, while a passive one cannot.

A *replay* attack takes place when a node in the system, or an entity controlling a communication link re-sends one (or more) of the messages that were previously sent through this node/link. This effectively replays a part or all of the followed message's path. If a specific path is followed upon, the adversary can use such replays to reproduce the communication pattern over and over again, making it statistically more prominent than others, thus isolating it. It is easy to find a remedy for this specific type of attack (using time-stamps or message identifiers for example), yet this revolves us back to the case of non-general solutions to specific problems. Preferably we would like a well designed protocol to be able to maintain unlinkability when being run many times. We call this scenario a multi-shot game. More on this issue will be discussed later.

2.3 Malicious Adversary

The third and most powerful type of adversary is called malicious. Generally speaking, this type of player has unlimited power within the network part it controls. A malicious player can inject new messages into the system, change messages as they pass through controlled nodes

and delete or delay message. We emphasize that such an adversary is not limited to the type of behavior described, and that there are surely many types of activities such an adversary can perform from controlled nodes we haven't yet thought of.

2.4 Our Network Attack Model

We have *nodes* and *communication links* in the system.

We assume nodes hold data items, and all data items are of the same length, or padded to the same length. Some nodes and links are under control of an adversary, others are not and are called *honest*. All links connected to a non-honest node are considered under control of the adversary. When the adjacent adversary is passive we assume all traffic on such a link is monitored and processed. When the adjacent adversary is adaptive or malicious we assume new messages may also originate through this link, even if they are not honest protocol messages.

The results presented in this work focus on networks in which all adversaries are considered to be adaptive. The protocol suggested withstands both passive and adaptive adversaries. The case of malicious adversaries is left for future work, and is briefly discussed in Section 7.1.

We assume that the system has a time bound Δ_{bound} on message delivery. Every message sent over a communication link arrives within that time bound. We also assume that all players have clocks, and all the clocks are within $\Delta_{accuracy}$ time of some global time, and that a public key infrastructure (PKI), *i.e.*, every node has a public key, and a public key directory is widely available. The most significant assumption we make is that at least a constant fraction of the communication links are honest.

Chapter 3

Defining Unlinkability

Say there are M active players and they wish to communicate with M distinct nodes[1]. Let π be the permutation that describes the communication pattern, *i.e.*, player i communicates with node $\pi(i)$, and let Π be the random variable whose value is π. If we have no prior information on the players' communication patterns Π is the uniform distribution over all $M!$ possible permutations. Now, let C be the random variable whose value is all the information available to the adaptive adversary, gathered from adaptive communication links and adaptive nodes. Specifically, C is a 0/1 matrix with rows indexed by time steps an columns indexed by edges and with $C_{t,e}$ being 1 iff there is some communication on edge e in time t. Simon and Rackoff require that (Π, C) is α–computationally close to some (Π, C') such that for all $\pi_1, \pi_2 \in S_M$ $|(C'|\Pi = \pi_1) - (C'|\Pi = \pi_2)|_1 \leq \alpha$.

One glaring omission in previous work is the implicit assumption that the adversary has no a-priori knowledge on communication patterns. This is explicit in the work of Rackoff and Simon, and implicit in the vast body of work on "applied" protocols. It seems clear that this assumption is typically false, *e.g.*, an adversary might know that residents of China tend to correspond with other residents of China. We also note that the prior knowledge the adversary has can have — in fact is likely to have — very small weight in a uniform world. *E.g.*, in a uniform world it is very unlikely that residents of Beijing correspond much more with residents of Shanghai than with residents of Tokyo.

We discuss prior knowledge in depth in Section 5.6. We believe that any reasonable definition that portends to say something about the real world should deal with prior knowledge.

We first define when a family of two correlated random variables $\{(\Pi, C)\} = \bigcup_n (\Pi_n, C_n)$

[1] If the M nodes are not distinct, then our protocol w.h.p. makes them distinct by adding a random identifier to each message. Protocol details are found Chapter 4.

is unlinkable. The definition is general, and also applies to scenarios other than communication over a network that is partially under adversarial control. We later on specialize to our case and define when a communication protocol is unlinkable. We begin with a definition of computational indistinguishability.

Definition 3.0.1. Let $A = \{A_n\}, B = \{B_n\}$ be two families of distributions. We say $d(A,B)_P \leq \delta(n)$, if for every family of polynomial-size Boolean circuits $\{T_n\}$, for every large enough n,

$$\left| \Pr_{x \in A_n}[T_n(x) = 1] - \Pr_{x' \in B_n}[T_n(x') = 1] \right| \leq \delta(n)$$

The following definition contains four alternative definitions.

Definition 3.0.2. A family of distributions $\{(\Pi, C)\} = \bigcup_n (\Pi_n, C_n)$ is $\alpha(n)$–unlinkable if,

- $\{(\Pi, C)\}$ is $\alpha(n)$–computationally close to some $\{(\Pi, C')\} = \cup_n (\Pi_n, C'_n)$.
  I.e., $d(\{(\Pi, C)\}, \{(\Pi, C')\})_P \leq \alpha(n)$, and,
- For every n, fix $\Pi = \Pi_n$, $C' = C'_n$ and $\alpha = \alpha(n)$. We require,
  - (Definition 1 [RS93]): for all $\pi_1, \pi_2 \in \Pi$, $|(C'|\Pi = \pi_1) - (C'|\Pi = \pi_2)|_1 \leq \alpha$.
  - (Definition 2, a variant on [RS93]): $\Pr_{\pi \in \Pi}[\ |(C'|\Pi = \pi) - C'|_1 \geq \alpha\ ] \leq \alpha$.
  - (Definition 3): $\Pr_{c \in C'}[\ |(\Pi|C' = c) - \Pi|_1 \geq \alpha\ ] \leq \alpha$.
  - (Definition 4): $I(\Pi : C') \leq \alpha$.

We prove:

Lemma 3.0.1. *let $\{(\Pi, C)\} = \bigcup_n (\Pi_n, C_n)$ be a family of arbitrary joint distributions, (Π_n, C_n) is distributed over some domain Λ_n.*

- *If $\{(\Pi, C)\}$ is $\alpha(n)$–unlinkable according to Definition 1, then it is $\alpha(n)$–unlinkable according to Definition 2. Conversely, If $\{(\Pi, C)\}$ is $\alpha(n)$–unlinkable according to Definition 2, then it is $2\alpha(n)$–unlinkable according to Definition 1.*
- *If $\{(\Pi, C)\}$ is $\gamma(n)$–unlinkable according to Definition 2 (Definition 3), then it is $\delta(n) = O(\log(|\Lambda_n|)\sqrt{\gamma(n)})$–unlinkable according to Definition 4. Conversely, If $\{(\Pi, C)\}$ is $\delta(n)$–unlinkable according to Definition 4, then it is $\gamma(n) = (2 \ln 2 \cdot \delta(n))^{1/3}$–unlinkable according to Definition 2 (Definition 3).*

3.1 Equivalence of Definitions

Before proving the equivalence we prove a lemma.

Lemma 3.1.1. *Let A and B be two arbitrary random variables, $0 \leq \alpha \leq 1$. Then*

$$\Pr_{b\in B}[\ |A-(A|B=b)|_1 \geq \alpha] \ \leq \ 2\ln 2 \cdot \frac{I(A:B)}{\alpha^2}$$

Proof.

$$\begin{aligned}
\Pr_{b\in B}[\ |A-(A|B=b)|_1 \geq \alpha] &= \Pr_{b\in B}[\ \frac{1}{2\ln 2}\cdot |A-(A|B=b)|_1^2 \ \geq \ \frac{1}{2\ln 2}\alpha^2] \\
&\leq \Pr_{b\in B}[\ D((A|B=b)\ ||\ A) \geq \frac{1}{2\ln 2}\alpha^2] \\
&\leq \frac{\mathbb{E}_{b\in B} D((A|B=b)\ ||\ A)}{\frac{1}{2\ln 2}\alpha^2}
\end{aligned}$$

where the first inequality is by Fact B.2.6 and the second one is Markov inequality. Thus, it suffices to prove that $\mathbb{E}_{b\in B}\ D((A|B=b)\ ||\ A) \ = \ I(A:B)$. Indeed,

$$\begin{aligned}
\mathbb{E}_{b\in B}\ D((A|B=b)\ ||\ A) &= \\
\sum_b \Pr(B=b) \sum_a \Pr(A=a|B=b)\cdot \log\left(\frac{\Pr(A=a|B=b)}{\Pr(A=a)}\right) &= \\
\sum_{a,b} \Pr(B=b)\Pr(A=a|B=b)\cdot \log\left(\frac{\Pr(A=a|B=b)\Pr(B=b)}{\Pr(A=a)\Pr(B=b)}\right) &= \\
D((A,B)||A\otimes B) = I(A:B)&
\end{aligned}$$

□

We now prove:

Proof. (Of Lemma 3.0.1) Fix n and set $\Pi = \Pi_n$, $C = C_n$, $C' = C'_n$.

Definition 4 → Definition 3: Suppose $I(\Pi : C') \leq \delta$. Set $\gamma = (2\ln 2\cdot\delta)^{1/3}$. We use Lemma 3.1.1 to get

$$\Pr_{c\in C'}[\ |\Pi-(\Pi|C'=c)|_1 \geq \gamma\ ] \leq 2\ln 2\cdot\frac{I(\Pi:C')}{\gamma^2} \leq 2\ln 2\cdot\frac{\delta}{\gamma^2} = \gamma.$$

Because $\gamma > \delta$ it follows that if $\{(\Pi, C)\}$ is δ-computationally close to $\{(\Pi, C')\}$ then $\{(\Pi, C)\}$ is also γ-computationally close to $\{(\Pi, C')\}$ and we are done.

Definition 4 $\rightarrow$ Definition 2 follows the same arguments simply by noting that $I(\Pi, C') = I(C', \Pi)$.

Definition 3 $\rightarrow$ Definition 4: We assume $\Pr_{c \in C'}[\ |(\Pi|C' = c) - \Pi|_1 \geq \gamma\ ] \leq \gamma$. We want to compare (Π, C') with $\Pi \otimes C'$. We see that:

$$\begin{aligned} d &= |(\Pi, C') - \Pi \otimes C'|_1 \\ &= \sum_c \sum_\pi |Pr(\Pi = \pi \mid C' = c\ ) \cdot Pr(C' = c) - Pr(\Pi = \pi) \cdot Pr(C' = c)| \\ &= \sum_c Pr(C' = c) \cdot |(\Pi|C' = c) - \Pi|_1 \ \leq\ 1 \cdot \gamma + 2 \cdot \gamma \leq 3\gamma. \end{aligned}$$

We want to use Fact B.2.4 with $(A, B) = (\Pi, C')$, and $(A', B') = (X, Y)$, (X, Y) chosen from $\Pi \otimes C'$. Because $H(\Pi \otimes C') = H(\Pi) + H(C')$ it follows that $I(A', B') = I(X, Y) = 0$ so we get that

$$|I(\Pi : C') - I(X, Y)| = I(\Pi : C') \quad \leq \quad 3 \cdot (3\gamma \cdot \log(|\Lambda|) + 3\gamma \cdot \log(\frac{1}{3\gamma}))$$

It follows that $I(\Pi : C') \leq \delta = O(\log(|\Lambda|)\sqrt{\gamma})$.

Definition 2 $\rightarrow$ Definition 4 is similar.

Definition 1 $\rightarrow$ Definition 2: We assume $\forall \pi_1, \pi_2 \in \Pi,\ |(C'|\Pi = \pi_1) - (C'|\Pi = \pi_2)|_1 \leq \alpha$. Fix $\pi \in \Pi$.

$$\begin{aligned} |(C'|\Pi = \pi) - C'|_1 &= |\sum_\sigma \Pr(\Pi = \sigma)(C'|\Pi = \pi) - \sum_\sigma \Pr(\Pi = \sigma)(C'|\Pi = \sigma)|_1 \\ &\leq \sum_\sigma \Pr(\Pi = \sigma) \cdot |(C'|\Pi = \pi) - (C'|\Pi = \sigma)|_1 \ \leq\ \alpha \ =\ \beta. \end{aligned}$$

Definition 2 $\rightarrow$ Definition 1 :

Start with Definition 2, $\Pr_{\pi \in \Pi}[\ |(C'|\Pi = \pi) - C'|_1 \geq \beta\ ] \leq \beta$. We now define a new joint distribution (C'', Π) that will satisfy both conditions of definition 1 as follows. We pick π according to the distribution Π. If the condition $|(C'|\Pi = \pi) - C'|_1 \geq \beta$ holds, we pick c'' according to the distribution $(C'|\Pi = \pi)$, otherwise we pick c'' according to the distribution C'.

As $d(X, Z)_P \leq d(X, Y)_P + d(Y, Z)_P$ for all families of distributions X, Y, Z, we have the following to satisfy the condition of computational closeness:

$$d(\{(\Pi, C)\}, \{(\Pi, C'')\})_P \leq d(\{(\Pi, C')\}, \{(\Pi, C'')\})_P + d(\{(\Pi, C)\}, \{(\Pi, C')\})_P \quad \leq \quad 2\beta$$

Also, by definition for every π_1, π_2,

$$|(C''|\Pi = \pi_1) - (C''|\Pi = \pi_2)|_1 \;\leq\; |(C''|\Pi = \pi_1) - C''|_1 + |C'' - (C''|\Pi = \pi_2)|_1 \;\leq\; 2\beta.$$

□

3.2 Unlinkable Communication Protocols

We now specialize to our case. We first define how unlinkable a specific protocol is.

Definition 3.2.1. We say a protocol is $\alpha(N)$–unlinkable according to definition i, $i \in \{1,2,3,4\}$, if for every N players, every choice of subsets S_N of honest players, and every distribution $\Pi_N(S_N)$ on their actual communication ($\Pi_N(S_N)$ is the prior knowledge) if we let $C_N(S_N)$ be the correlated random variable that contains the information known to the adversary, then $\bigcup_N(\Pi_N(S_N), C_N(S_N))$ is $\alpha(N)$– unlinkable according to definition i.

We now define what an *efficient* unlinkable protocol is, according to definition i, $i \in \{1,2,3,4\}$:

Definition 3.2.2. Let $i \in \{1,2,3,4\}$. Let P be a protocol parameterized by the number of users N in the system and the designated error function $\alpha(N)$. Let us denote $P = \bigcup_{N,\alpha(N)} P_{N,\alpha(N)}$, where $P_{N,\alpha(N)}$ is the protocol P when given N and $\alpha(N)$.

We say:

- P is *unlinkable* according to definition i, if for every possible error function $\alpha(N) \geq N^{-O(1)}$, $\bigcup_N P_{N,\alpha(N)}$ is $\alpha(N)$–unlinkable according to definition i.
- P is an *efficient* unlinkable protocol according to definition i, if in addition to P being unlinkable according to definition i, for every possible error function $\alpha(N) \geq N^{-O(1)}$, $P_{N,\alpha(N)}$ takes $T(N) = O(\text{poly}(\log(\frac{N}{\alpha(N)})))$ rounds, and $O(M \cdot T(N))$ messages, when M is the number of players who wish to send a message at a time.

Remark. We assume that $\alpha(N) \geq N^{-O(1)}$. Because our protocol is efficient we have that $T \leq \text{poly}(\log(N))$, and for concreteness we take $T \leq O(N)$.

Theorem 3.2.1. *A protocol P is efficiently unlinkable according to any one definition iff it is efficiently unlinkable according to all definitions.*

We therefore say a protocol is efficiently unlinkable if it is efficiently unlinkable according to any one of these definitions.

Proof. Let $P = \bigcup_{N,\alpha(N)} P_{N,\alpha(N)}$ be an efficiently unlinkable protocol according to some definition i. This means that:

- For every $\alpha(N) \geq N^{-O(1)}$, $\bigcup_N P_{N,\alpha(N)}$ is $\alpha(N)$–unlinkable according to definition i, and,
- The delay of the protocol $P_{N,\alpha(N)}$, $T(N, \alpha(N))$, is $\text{poly}(\log(\frac{N}{\alpha(N)}))$.

Let us denote $Q_{N,\alpha'(N)} = P_{N,\alpha(N)}$ where $\alpha'(N) = \log(|\Lambda_N|)\alpha(N)^{1/3}$ and $|\Lambda_N|$ is the size of the domain on which (C_N, Π_N) is distributed. We denote $Q = \bigcup_{N,\alpha'(N)} Q_{N,\alpha'(N)}$. We stress that we do not change the protocol only its indexing.

- By Lemma 3.0.1 $\bigcup_N Q_{N,\alpha'(N)}$ is $\alpha'(N)$–unlinkable according to definition j, and,
- For every $\alpha'(N) \geq N^{-O(1)}$, the delay for $Q_{N,\alpha'(N)}$ is the delay for $P_{N,\alpha(N)}$, which is $\text{poly}(\log(\frac{N}{\alpha(N)})$. However, $\alpha'(N) = \log(|\Lambda_N|)\alpha(N)^{1/3}$ and we show below that $\log(|\Lambda_N|) \leq O(N^3)$, and so $\alpha'(N) \leq \text{poly}(N) \cdot \alpha(N)^{1/3}$. We conclude that the delay of $Q_{N,\alpha'(N)}$ is $\text{poly}(\log(\frac{N}{\alpha'(N)}))$ as desired.

We conclude with the calculation of $\log(|\Lambda_N|)$. The random variable C_N tells for each dishonest link in the protocol whether there was communication on that link or not. There are $M \leq N$ vertices and so at most N^2 links per layer, and DN^2 links overall in a protocol of length D. For every such link, C_N may contain a zero/one answer whether there was traffic on that link or not. Altogether C_N has support 2^{DN^2}. Π_N has a smaller support. Altogether $\log(|\Lambda_N|) = O(DN^2)$. However, as noted above $D \leq T \leq N$. Altogether, $\log(|\Lambda_N|) \leq O(N^3)$. Thus, $\alpha'(N) \leq O(N^3)\alpha(N)^{1/3}$. □

Chapter 4

An Anonymous Communication Protocol

Our protocol is very similar to Chaum's [Cha81]. We first describe it in a synchronous system, and in Section 4.1 we explain how to adapt it to an asynchronous system. A wants to send a message $a \in \{0,1\}^S$ to B and get back an answer $b \in \{0,1\}^S$, where S is the length of data items in the system. It is crucial that all data items in the process are of the same length to disallow traffic analysis based on different lengths of items. A picks $T-1$ random nodes $v_1, \ldots, v_{T-1}$, and sets $v_0 = A$, $v_T = B$. A also picks T random strings $r_i \in \{0,1\}^S$, and $z_i \in \{0,1\}^{\ell_i}$ where ℓ_i is a security parameter for the encryption schemes E_i. We let $E_1, \ldots, E_T$ be the public encryption methods of the T nodes. We denote

$$a_i \;=\; E_{i+1}(r_{i+1}, z_{i+1}, v_{i+2}, E_{i+2}(\ldots E_{T-1}(r_{T-1}, z_{T-1}, v_T, E_T(r_T, a)))\ldots)$$

for $i = 0, \ldots, T-1$. The cascade of encryptions is depicted in Figure 4.1.

The way from A to B : A sends $(0, v_0, z_0, a_0)$ to v_1. In general, v_i sends the message (i, v_i, z_i, a_i) to v_{i+1}. When v_{i+1} receives (i, v_i, z_i, a_i) where $a_i = E_{i+1}(r_{i+1}, z_{i+1}, v_{i+2}, a_{i+1})$ it decrypts a_{i+1}, and sends $(i+1, v_{i+1}, z_{i+1}, a_{i+1})$ to v_{i+2}. It also records $v_i, v_{i+2}, z_i, z_{i+1}$ and r_{i+1}. $v_T = B$ recognizes it is the last on the path, and prepares an answer $b \in \{0,1\}^S$ to the message a it receives.

The way back : $B = v_T$ sends (v_T, z_{T-1}, b_T) to v_{T-1} where $b_T = b \oplus r_T$. In general, v_i receives a message (v_{i+1}, z_i, b_{i+1}). v_i recognizes the value z_i, the link (v_{i-1}, v_i) that precedes (v_i, v_{i+1}) and the values r_i, z_{i-1} that are associated with it. It then sends $(v_i, z_{i-1}, b_i =$

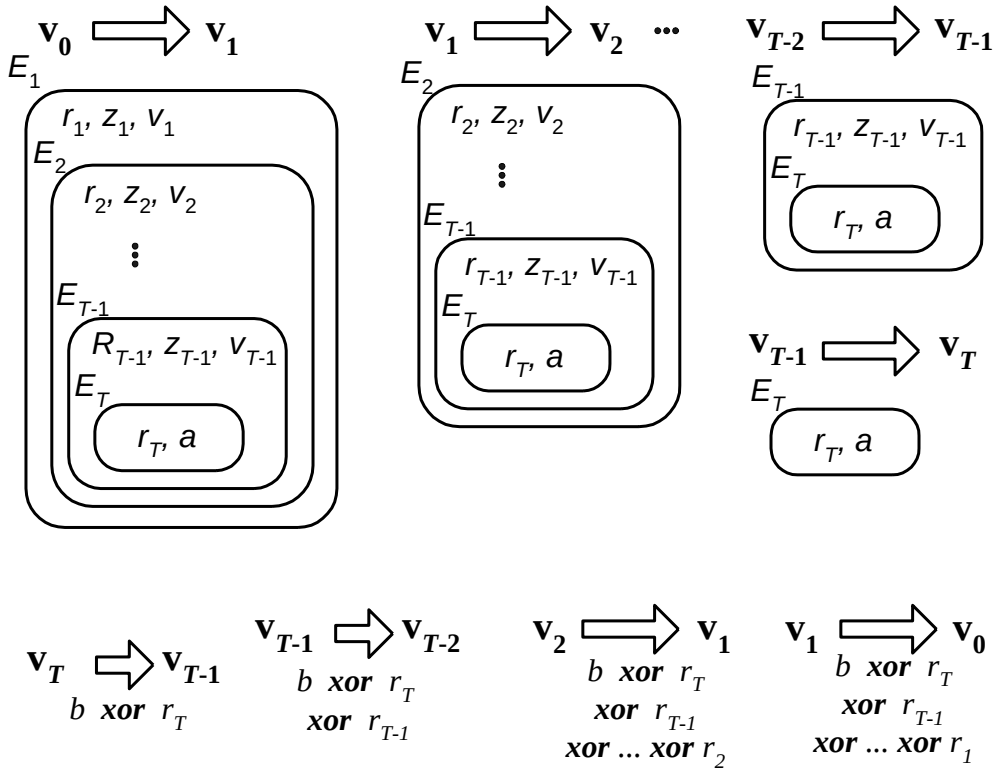

Figure 4.1: *The cascade of encryptions when sending a message a from v_0 to v_T via the path $v_0 \mapsto v_1 \mapsto v_2 \mapsto \cdots \mapsto v_{T-1} \mapsto v_T$, where E_i is the public key encryption associated with node v_i. Note that this process already prepares the symmetric keys (the r_i strings) used when sending the response b from v_T to v_0, if required, along the reverse path $v_T \mapsto v_{T-1} \mapsto \cdots \mapsto v_2 \mapsto v_1 \mapsto v_0$. The important data transmitted along the reverse path is also illustrated above.*

$b_{i+1} \oplus r_i$) to v_{i-1}. Finally, $A = v_0$ receives $(v_0, z_0, b_0 = b_1 \oplus r_1)$ from v_1. The value $b_0 \oplus r_1 \oplus \ldots r_T$ is the desired value b.

Remarks:

1. In the description above we've explicitly given v_{i+1} the identity of v_i. This is not really required in real protocols such as TCP/IP where the recipient knows the identity of the sender in any case.

2. The role of the z_i fields is simply to uniquely identify the returning message as associated with the forward message. This is here simply to allow the same node to participate in multiple simultaneous paths. It should be noted that this field can limit the adversary with the usage of various attacks such as the message replay attack described in Chapter 2.

3. The keys r_i are used as one time pads for the transferred data, and therefore need to be as long as the data items. It is, however, possible to use any other symmetric encryption algorithm with constant short-length keys, *e.g.*, DES or AES, assuming it is secure against the adversary. When measuring the data overhead imposed by our protocol on the length of a single message we notice that using short-length keys as suggested, only a constant amount of data is added to each data item, regardless of the data item size in the system. The only other factor affecting message length is the length of paths in the system. For paths of T rounds and data items of S bits, the size of messages is therefore $O(S + T)$ bits rather than $O(S \cdot T)$.

4. When looking carefully at the details of each packet transferred, we see that modification of messages or initiation of new messages does not allow the change of paths of returning messages. This is the reason we ignore the return path and analyze only the forward path, as the return path is identical. In addition, an adaptive adversary is not able to alter the forward path of a message, or its content, as the data belonging to the next nodes in the path is encrypted. The usage of encryption "onions" in a single run of our protocol inhibits an adversary from utilizing its ability to initiate new messages in order to single out a communication pattern.

Theorem 4.0.2. *Assume the above protocol runs in a network with N nodes, $\binom{N}{2}$ communication links, some constant fraction of which are honest, then the protocol is $\alpha(N)$–unlinkable when $T \geq \Omega(\log(N) \log^2(N/\alpha(N)))$.*

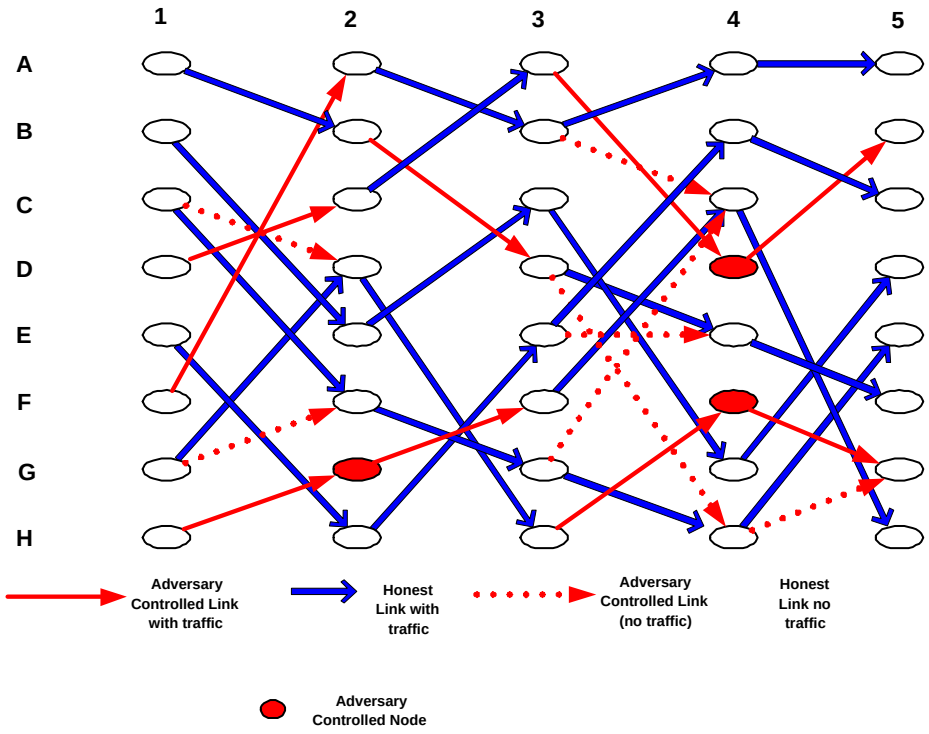

Figure 4.2: *Adversary controlled nodes imply that the adjacent links are also adversary controlled. We drew only the actual communication links used (with solid lines) going in and out of adversary controlled nodes. The nodes at coordinates* (x, y) *and* (x', y') *are not necessarily distinct.*

Our protocol does not make any sense in a network that is fully controlled by an adaptive adversary, as the adversary can match each message going into a node with the message going out of the node in the next time step. What we show is that if a non-negligible fraction of the communication links are honest, then our protocol is anonymous. We remind the reader that we count all connections entering or leaving an adaptive node as adaptive, and in addition there might be additional adaptive connections between honest nodes.

We picture a possible execution of the protocol in Figure 4.2.

4.1 The Asynchronous Setting

Recall that all players have clocks, and all the clocks are within $\Delta_{accuracy}$ time of some global time. Also, the system has a time bound Δ_{bound} on message delivery. We fix some arbitrary global time t_0. The protocol specifies that a player can send a message only at times $t_0 + i\Delta$, for $i \in Z$ and $\Delta = 2\Delta_{accuracy} + \Delta_{bound}$. The players then run the above protocol as if their

clocks are truly synchronized, and each step takes Δ time.

Let us denote $t^{(i)} = t_0 + i\Delta$. One immediate property is that if M arbitrary players $A_1, \ldots, A_M$ send a message on their local time $t^{(i)}$ to arbitrary M players $B_1, \ldots, B_M$, and players B forward the messages they receive on their next possible time step, then all these messages will be sent in local time $i+1$. This is regardless of the difference between local times of the different players, and the differences in delivery time. It turns out that this is all we need from our system, thus we get that Theorem 4.0.2 holds also in the asynchronous model.

Chapter 5

Unlinkability Proof

5.1 Proof Sketch

Generally speaking, in order to prove that the above protocol is secure, a process of structuring is needed to be done to the communication patterns, to allow for easy analysis and calculations.

To perform this process a special communications network is constructed, an "Obscurant Network" (See Section 5.2). Apart from the data flow properties of this network that allows anonymity, this network has a highly static and structured communication pattern, compared with the patterns created by our protocol.

In order to analyze the amount of data the adversary gathers from the pattern created by our protocol, we show that this pattern has enough honest links within it that together contain an "embedded" obscurant network. After describing what an embedding is and an algorithm to find one in Section 5.4, we prove that our protocol's communication pattern contains, w.h.p., such an embedding in Section 5.5 for the case of no-prior information.

Our proof makes use of another interesting technique. During most steps of the analysis, information is purposefully being revealed to the adversary regarding communication on links which are not under its control. This classifies the links in the network into two, ones where the adversary has full information of data flow, and ones where the adversary has absolutely no information about the flow of data. Showing both the existence of an embedding of an obscurant network as well as giving all other irrelevant data to the adversary allows for a simple proof in the case of no-prior information.

Prior information is dealt with in Section 5.6. A folding trick is used to reveal yet some more information to the adversary about the connection between information flowing from the sources of the message and the information arriving at the final destinations of messages. This

trick literally folds the communication pattern in half when observed from the adversary's point of view, reducing the analysis to the case of no-prior information, when the interesting layer of the protocol is the middle layer of communications. We then show that the middle layer does not convey enough information to the adversary, resulting in unlinkability. The result requires longer message paths in order to achieve a probable embedding of an obscurant network.

5.2 Definition and Properties of Obscurant Networks

A network is a layered directed circuit with the same number of vertices on each layer. We say a circuit is a crossover network, if every vertex has in-degree and out-degree one or two, and the circuit consists of crossover gates and simple connections only. An example of such a network is depicted in Figure 5.2.

We think of the following game: a pebble is put on some input vertex, say on the i'th vertex. If the vertex out-degree is one, we follow that link. Otherwise, we follow each of the crossover links with probability half. By the end of the game we get a distribution O_i over the output elements. We say the network ϵ–obscures the i'th input, if $|O_i - U_M| \leq \epsilon$. We say a network ϵ-obscures inputs, if it ϵ-obscures every input.

We can generalize this notion to crossover networks that obscure *permutations*. We think of the game where there are M pebbles, numbered from 1 to M. If a pebble lies on a vertex whose out-degree is one, the pebble follows that link. Otherwise, if two pebbles share a crossover, we choose each of the two possible matchings with equal probability and follow the matching. By the end of the game we get a distribution O over all possible permutations. We say the network ϵ–obscures permutations, if $|O - U_{M!}| \leq \epsilon$.

It is not true that a network that obscures inputs must also obscure permutations. An example for that is the butterfly network B_n which perfectly obscures inputs but does not obscure permutations. Notice that if we know where pebble 1 goes to, we also know that pebble 2 can go only to a subset of size $n/2$ of possible destinations. We do not know how to show that shallow crossover networks that obscure permutations exist. Section 1.5 describes an attempt to create such networks, yet a tremendous amount of coordination between different nodes and levels of the network is required.

In what follows we show a simple, shallow crossover network that obscures inputs. The construction is explicit. We call networks that obscure their inputs, obscurant networks.

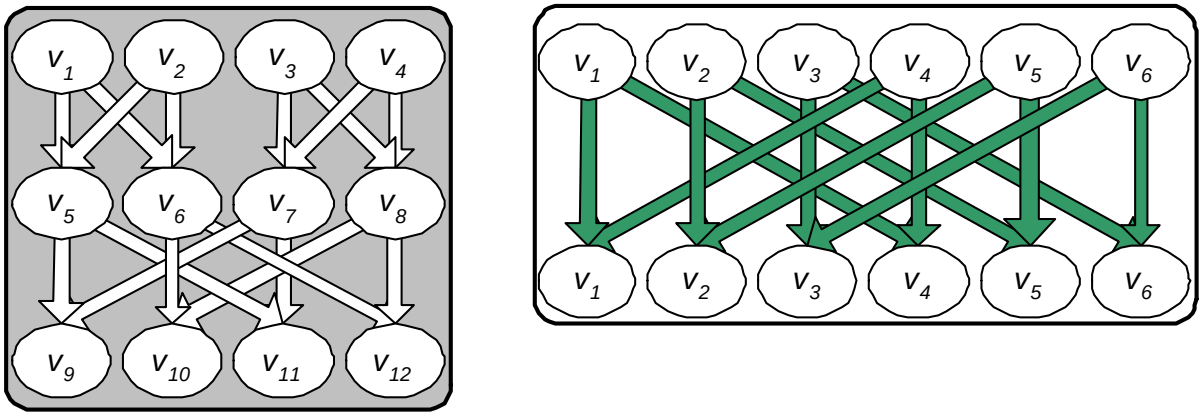

Figure 5.1: Example of components used to build an obscurant network. The left network is a butterfly with $Z = 4$ elements, B_4. The right network is P_{2k} with $k = 3$.

5.3 Constructing Shallow Networks That Obscure Inputs

We now build a shallow obscurant network on M inputs, for some given fixed integer M.

Let Z be the largest power of two not larger than M, *i.e.* $\frac{M}{2} < Z \leq M$. In the construction we use two components: a butterfly network B_Z with Z vertices in each layer, with comparators replaced with crossovers, and a network over $2k$ elements and two layers with k crossovers connecting vertex i in the first layer with both vertex i and vertex $k+i$ $(mod\ 2k)$ in the second layer, for $i = 1, \ldots, 2k$. We call this later network P_{2k}. Figure 5.1 illustrates an example of these components.

We distinguish between two cases. If $Z = M$ we put B_Z on the Z inputs. Otherwise, $\frac{M}{2} < Z \leq M$. We then do the following:

- For the first level, we put B_Z on the Z rightmost elements.
- For the second level, We put B_Z on the Z leftmost elements.
- For the third level, We put $P_{2(M-Z)}$ on the $2(M-Z)$ rightmost elements.
- For the fourth level, we put B_Z on the Z leftmost elements.
- We then iterate the third and fourth levels $\log(M) + \log \epsilon^{-1}$ times. See Figure 5.2.

We claim:

Lemma 5.3.1. *If the depth D of the network is $O((\log(M) + \log \epsilon^{-1}) \log(M))$, the network is ϵ–obscurant.*

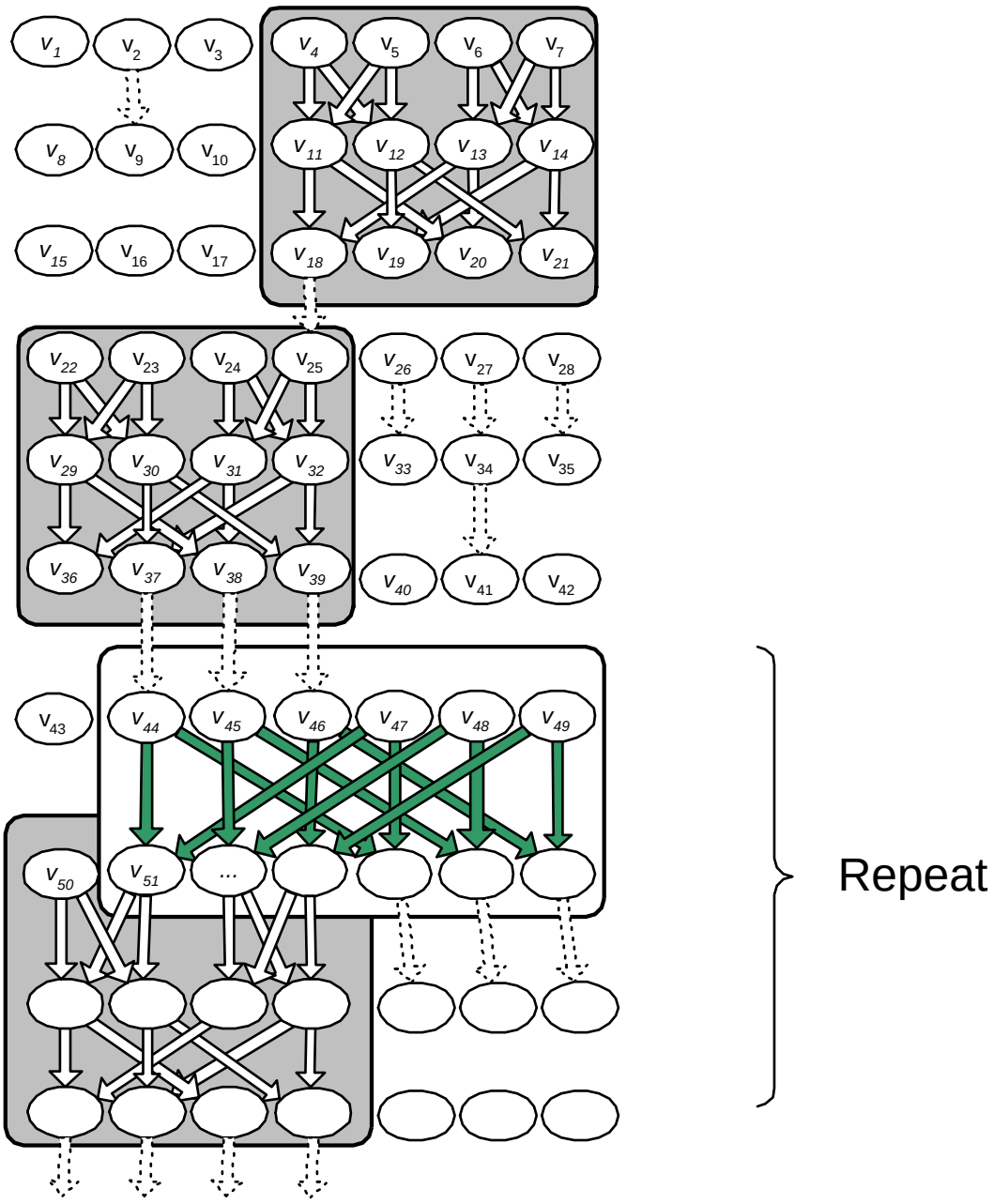

Figure 5.2: *An obscurant network for $M = 7$, $Z = 4$. Simple connections are often omitted. The boxes with the grey background are B_4 butterflies. The other boxed sub-circuit is P_6.*

Proof. If $M = Z$, then for every input vertex, i spans a tree. It follows that $O_i = U_M$ and the network is 0–obscurant.

Suppose $\frac{M}{2} < Z < M$. Let i be a starting vertex. Notice that B_Z gives equal weight to each of its Z outputs. It therefore follows that after the application of B_Z on the right followed by B_Z on the left, all the leftmost Z elements have one weight, ℓ_0, and all the remaining elements $M-Z$ (rightmost) elements have the same (possibly different) weight, r_0. One can observe that this property is an invariant that remains valid throughout the protocol, *i.e.*,

- The invariant: After applying the pair of transformations $P_{2(M-Z)}$ and B_Z i times, $i \geq 0$, the Z leftmost elements all have weight l_i and the $M - Z$ rightmost elements all have weight r_i.
- After applying the $P_{2(M-Z)}$ network of the i+1'st pair, all the $2(M-Z)$ rightmost elements have one weight $r_{i+1} = (r_i + l_i)/2$ and all the remaining $M - 2(M - Z) = 2Z - M = K$, $1 \leq K < Z$, leftmost elements remain at weight l_i.
- After applying the B_Z network of the $i + 1$'st pair, all the Z leftmost elements have one weight $\ell_{i+1} = \frac{K \cdot l_i + (Z-K) \cdot r_{i+1}}{Z}$ and all the $M-Z$ rightmost elements remain at weight r_{i+1}.

Our goal is to show that ℓ_i approaches r_i quickly.

$$\begin{aligned} |\ell_{i+1} - r_{i+1}| &= \left| \frac{K \cdot \ell_i + (Z-K) \cdot r_{i+1}}{Z} - \frac{r_i + \ell_i}{2} \right| = \left| \frac{(Z-K)r_i + (Z+K)\ell_i}{2Z} - \frac{Zr_i + Z\ell_i}{2Z} \right| \\ &= \left| \frac{K}{2Z}(\ell_i - r_i) \right| < \frac{1}{2}|\ell_i - r_i|. \end{aligned}$$

The last inequality follows as $K < Z$.

As $|\ell_0 - r_0| \leq 1$ it must be that $|\ell_t - r_t| < 2^{-t}$. We see that $1 = Z\ell_t + (M - Z)r_t < Z\ell_t + (M - Z)(\ell_t + 2^{-t}) = M\ell_t + (M - Z)2^{-t}$ and so $\ell_t \geq \frac{1-(M-Z)2^{-t}}{M}$. We similarly show that $\ell_t \leq \frac{1+(M-Z)2^{-t}}{M}$. Together, $|\ell_t - \frac{1}{M}| \leq \frac{(M-Z)}{M}2^{-t}$. To conclude the proof we note that $|O_i^{(t)} - U_M|_1 \leq 2M \cdot |\ell_t - \frac{1}{M}| \leq 2(M - Z)2^{-t}$. As $M - Z < M/2 < M$, we get that $|O_i^{(t)} - U_M|_1 \leq M2^{-t} \leq \epsilon$, for $t = \log(M) + \log \epsilon^{-1}$. □

5.4 Finding Obscurity Within Executions of Our Protocol

Say we have M honest active players that start sending messages according to the protocol. Say also that we have in mind an obscurant network G over M inputs and of depth D. Our goal is to show that if the M honest players run the protocol T steps, for some T large enough,

then the network G, in a sense, appears as a subgraph of the protocol. The precise notion of G appearing in the protocol is somewhat delicate and we explain it in detail soon.

We begin with a combinatorial lemma. The basic fact that we know about our system is that at least an f fraction of the links are honest. *I.e.*, if we let V be the set of all nodes and E the set of all honest links, then $|E| \geq f\binom{|V|}{2}$, $0 < f \leq 1$. The following combinatorial lemma asserts that if we choose four vertices a, b, c, d at random from V, then there is a crossover structure on the four vertices with probability at least f^4.

Fact 5.4.1. *([Alo01], Corollary 2.1) Let $G = (V, E)$ be a graph and assume $|E| \geq f\binom{|V|}{2}$. Then,*

$$\Pr_{a,b,c,d \in V} [\ \{(a,c),(a,d),(b,c),(b,d)\} \subseteq E\ ] \quad \geq \quad f^4$$

5.4.1 Network Embedding

We have two graphs. One is a crossover network G of depth D and width M, the other is the network P of depth T and width M originating from an execution of the protocol.

- We represent G as $G = (V_G, \boxtimes_G, I_G)$ where V_G is the set of DM vertices of G, $\boxtimes_G$ is the set of all groups of four vertices $(a, b; c, d)$ such that (a, c), (a, d), (b, c), (b, d) form a crossover of G, and I_G is the set of all edges of G not participating in any crossover.
- We represent P as $P = (V_P, T_P, C_P)$ where V_P is the set of TM vertices participating in the protocol, T_P is the set of all links carrying traffic in the execution of the protocol, and C_P the set of all links that are under adversary control (whether used to carry traffic or not).

Definition 5.4.1. A function $\phi : V_G \times \{0, 1\} \to V_P$ is an embedding if:

- The mapping ϕ respects T_P. *I.e.*,
  - Each direct (non crossover) link in G is mapped to a direct connection in P: $\forall_{e=(v,w)\in I_G}\ (\phi(v, 1), \phi(w, 0)) \in T_P$.
  - Each crossover of links in G is mapped to a crossover in P: $$\forall_{(a,b;c,d)\in\boxtimes_G} \left| \left\{ \begin{array}{ll} (\phi(a,1),\phi(c,0)), & (\phi(a,1),\phi(d,0)), \\ (\phi(b,1),\phi(c,0)), & (\phi(b,1),\phi(d,0)) \end{array} \right\} \cap T_P \right| = 2.$$
- Each vertex in G is mapped to a route in P: For every $v \in V_G$, $\phi(v, 0)$ and $\phi(v, 1)$ are connected in T_P. This means that the routes of data flow in G are preserved within P with expansions of some vertices into paths. The only property we utilize when searching

for anonymity is the lack of knowledge the adversary has regarding data flow on crossover edges of G, which are all mapped to crossovers in P. We do not require anything regarding the knowledge of how vertices in G are mapped into P, which permits us to tell all this information to the adversary. The notion of $\phi(v, 0)$ for $v \in V_G$ represents the entrance of communication links to a node in G, while $\phi(v, 1)$ is the exit point of communication links.

- The adversary does not know any link in any crossover. *I.e.*, for every $(v_1, v_2; w_1, w_2) \in \boxtimes_G$ and every $i, j \in \{1, 2\}$, $(\phi(v_i, 1), \phi(w_j, 0)) \notin C_P$.

Remark. During our analysis we notice that each layer of execution of our protocol contains exactly M distinct nodes, making a scenario where traffic goes out from one node to two destinations in one step impossible. *I.e.*, $\forall_{a,b,c \in V_P},\ |\{(a, b), (a, c)\}| \cap T_P| \leq 1$. This allows for the simplicity of the first condition.

We define $\phi_P(\boxtimes_G)$ to be the image of $\boxtimes_G$ under the embedding ϕ in the execution P. *I.e.*, the set of all $(u_1, u_2; u_3, u_4) \in V_P^4$ for which there exist $(v_1, v_2; v_3, v_4) \in \boxtimes_G$ s.t. $\phi(v_1, 1) = u_1$, $\phi(v_2, 1) = u_2$, $\phi(v_3, 0) = u_3$ and $\phi(v_4, 0) = u_4$.

As P is created randomly, and since C_P and T_P change during every instance of P, the embedding ϕ clearly depends on P. However, to be useful for us we need an additional property. We require that ϕ is independent of the communication that took place on the embedded copy of G. Formally,

Definition 5.4.2. Let G be a crossover network of depth D and width M. Let $\mathcal{P}$ be a protocol (*e.g.*, the protocol of Chapter 4) over M players and T steps. An embedding strategy for the protocol, with ϵ error, is an algorithm that given an execution $P = (V_P, T_P, C_P)$ of the protocol, outputs a function $\phi_P : V_G \times \{0, 1\} \to V_P$ such that:

- $\Pr_{coins\ of\ P}[\phi_P \text{ is an embedding}] \geq 1 - \epsilon$, and,
- For every two protocols P and P' that use the same sets of vertices, if $T_{P'}$ agrees with T_P on all edges not participating in $\phi_P(\boxtimes_G)$ then $\phi_P = \phi_{P'}$.

5.4.2 An Embedding Strategy

Let G be a crossover network of width M and depth D. Let $\mathcal{P}$ be our protocol from Chapter 4 with $T = 2kD$, for some k that we fix later. We now describe an embedding strategy. We are given an execution $P = (V_P, T_P, C_P)$ of the protocol, and we have to define an embedding $\phi_P : V_G \times \{0, 1\} \to V_P$ with the properties described above.

The strategy is composed of the following steps and algorithms:

- Label G to be used later when constructing ϕ_P:

  We label a vertex u of G by $u_i^{(d)}$, where $0 \leq d \leq D$ is the depth of u, and i comes from an arbitrary labelling of the d'th layer with labels $\{1, \ldots, M\}$, such that all edges in G are either of the simple form $(u_i^{(d)}, u_i^{(d+1)})$ or of the crossover form $(u_i^{(d)}, u_j^{(d+1)})$ where $(u_i^{(d)}, u_j^{(d)}; u_i^{(d+1)}, u_j^{(d+1)}) \in \boxtimes_G$.

- Label V_P, create the mapping ϕ_P and reveal all unneeded information to the adversary:

  Algorithm 5.4.2. (An algorithm for labelling V_P and constructing $\phi_P : V_G \times \{0,1\} \to V_P$)

  Initialization :

  Label the vertices in the bottom layer of P with $v_i^{(0)}$, where $i \in \{1, \ldots, M\}$ and the labels inside the layer are chosen arbitrarily (say, by lexicographic order on the identity of the vertex). Define $\phi_P(u_i^{(0)}, 0) = v_i^{(0)}$.

  Layer $0 \leq d \leq D$ **of** G : We go over the layers d of G, one by one, and find a mapping for it.

  With each new layer we perform the following:

  Set $ok(i) = false$ for every $i \in \{1, \ldots, M\}$. This tells us that we still have to take care of all vertices in the d'th layer of G.

  Go over the layers $t = d \cdot 2k + \ell$, $1 \leq \ell \leq 2k$ of vertices in P:

  Odd t : Reveal all communication on links going from a vertex in layer $t-1$ to a vertex in layer t. For every revealed edge $(v_i^{(t-1)}, w) \in T_P$ we label w with $v_i^{(t)}$.

  Even t : For every $i \in \{1, \ldots, M\}$ we do the following:

  - If $ok(i) = true$ it means that the edge going out of u_i^d has been embedded previously. We reveal the edge $(v_i^{(t-1)}, w) \in T_P$, and label w with $v_i^{(t)}$.
  - For $ok(i) = false$ we do the following:
    If $u_i^{(d)}$ belongs to a simple edge $(u_i^{(d)}, u_i^{(d+1)})$, then we reveal the edge $(v_i^{(t-1)}, w) \in T_P$, and label w with $v_i^{(t)}$. We also set $\phi_P(u_i^{(d)}, 1) = u_i^{(t-1)}$, $\phi_P(u_i^{(d+1)}, 0) = u_i^{(t)}$ and $ok(i) = true$.
    Otherwise, $u_i^{(d)}$ belongs to $(u_i^{(d)}, u_j^{(d)}; u_i^{(d+1)}, u_j^{(d+1)}) \in \boxtimes_G$. Let w and z be the vertices such that $(v_i^{(t-1)}, w), (v_j^{(t-1)}, z) \in T_P$.
    If one of the edges $(v_i^{(t-1)}, w)$, $(v_i^{(t-1)}, z)$, $(v_j^{(t-1)}, w)$ or $(v_j^{(t-1)}, z)$ is in C_P, it means that we have not found a clean crossover, where all of its nodes are

honest and are hidden from adversary eavesdropping. We thus reveal all the above four edges. We also label w with $v_i^{(t)}$.

If, on the other hand, all these four edges are honest, we label $\{w, z\}$ with the labels $\left\{v_i^{(t)}, v_j^{(t)}\right\}$ in an arbitrary order (say, by the natural order on i and j as numbers) and we set $\phi_P(u_i^{(d)}, 1) = v_i^{(t-1)}$, $\phi(u_i^{(d+1)}, 0) = v_i^{(t)}$, $\phi_P(u_j^{(d)}, 1) = v_j^{(t-1)}$, $\phi(u_j^{(d+1)}, 0) = v_j^{(t)}$, and both $ok(i) = true$ and $ok(j) = true$. We also say, then, that we have found the crossover $(u_i^{(d)}, u_j^{(d)}; u_i^{(d+1)}, u_j^{(d+1)}) \in \boxtimes_G$ in P.

Failure detection : The algorithm can stop and claim failure in case it has processed the entire layer t of vertices of P when $\ell = 2k$ without having $ok(i) = true$ for all i at the end. This would mean that the algorithm was not able to find an embedding.

The first and second conditions of Definition 5.4.1 hold directly from the way we choose the embedding ϕ_P.

The third condition holds whenever the algorithm stops with a positive successful answer, because we then embed every crossover of G in V_P in a way that uses only honest edges. We later show that the probability of a successful result is high. Figure 5.3 illustrates the embedding procedure.

Furthermore, Algorithm 5.4.2 is an embedding strategy. To see that, fix two executions P and P' of the protocol that use the same sets of vertices, and that agree on all communications over links not in $\phi_P(\boxtimes_G)$. As P and P' differ only on crossovers, and the labelling of the vertices at the second layer of the crossover depends only on a pre-determined order (not influenced by the actual communication over the crossover), the labelling in P is the same as in P'. It therefore follows that $\phi_P = \phi_{P'}$.

To complete the argument we show that with high probability (over the random coins of the protocol $\mathcal{P}$ from Chapter 4) we find all crossovers of $\boxtimes_G$ in P, and the algorithm ends successfully.

Claim. *For every crossover $(u_i^{(d)}, u_j^{(d)}; u_i^{(d+1)}, u_j^{(d+1)}) \in \boxtimes_G$, the probability we do not find it in an execution P of the protocol from Chapter 4 is at most $(1 - f^4)^k$.*

Proof. Fix $(u_i^{(d)}, u_j^{(d)}; u_i^{(d+1)}, u_j^{(d+1)}) \in \boxtimes_G$. For every time step $t = 2kd + \ell$, $2 \leq \ell \leq 2k$, look at the vertices $v_i^{(t-1)}, v_j^{(t-1)}, v_i^{(t)}, v_j^{(t)}$. Remember that in our protocol the vertices in each path are chosen at random. Furthermore, we reveal all edges going from even layers to odd layers. Thus, the vertices in the $t-1$ and t'th layers are chosen at random and independent of history. In particular, the above four vertices are chosen at random, and independent of history. By Fact

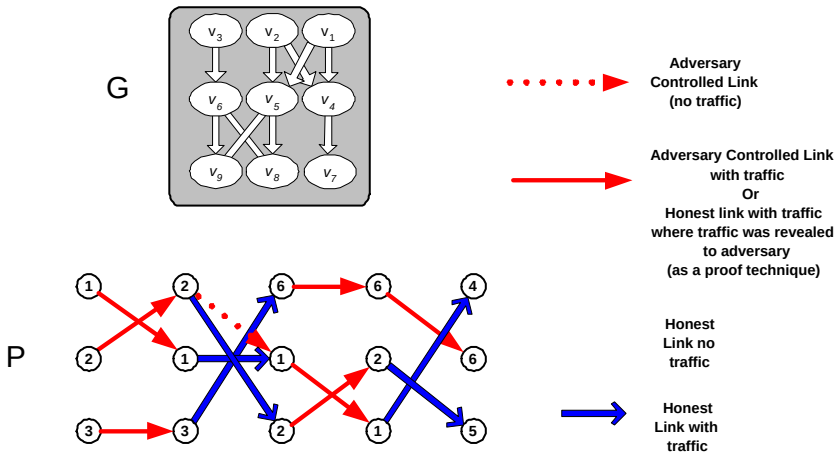

Figure 5.3: *We seek the crossover network G (depicted on the top in the shaded box) within the protocol network P depicted below it. Notice that every other layer in the protocol network is revealed.*
G has a single crossover in the first layer, going out of vertices 1 and 2. We do not find this crossover in the second layer of P because not all four edges of the crossover are honest. It does appear however in the fourth layer of P. Notice the relabelling of the vertices at the other side of the crossover.

5.4.1 we find a crossover with probability at least f^4. As different steps are independent, the probability we do not find a crossover in any of the k attempts is at most $(1-f^4)^k$. □

Using the union bound we see that:

Claim. *Let G be any crossover network with M inputs and depth D. Let us run the protocol of Chapter 4 with M honest nodes and for $T = 2kD$ steps. Let P be the resulting network. Then* $\Pr[G \text{ does not appear in } P] \leq DM(1-f^4)^k$.

5.5 Unlinkability Without Prior Information

Our goal now is to prove that our protocol is unlinkable. We first deal with the no prior knowledge case, *i.e.*, when the a-priori distribution is uniform. We then show in Section 5.6 how the no prior knowledge case implies the general case.

We show that given knowledge of how players $1, \ldots, j$ behave, the adversary does not know how player $j+1$ behaves. For every $j = 1, \ldots, M$, we display a *different* obscurant network G_j, over $M - j$ players, in the actual execution of the protocol.

Suppose there are M honest players sending messages in a network with N players, and

let $\alpha(N) > N^{-c}$. Let $G = G_M$ be an ϵ-obscurant network over M inputs and of depth $D = O(\log(\frac{M}{\epsilon})\log(M))$. Suppose we run the protocol for $T = 2Dk$ steps. We would like to set values for ϵ and k such that we receive $\alpha(N) - \textit{unlinkability}$ with our protocol.

We define the following random variables:

X : X contains all the actual information generated throughout the protocol. *I.e.*, for every link $(v_i^{(t)}, v_j^{(t+1)})$ it contains the information whether there was traffic on that link or not.

Π : $\Pi(i)$ contains the actual destination of the i'th honest player. The random variable $\Pi = \Pi(1)\ldots\Pi(M)$ contains the actual communication pattern between the M honest players and the M destinations.

C' : C' contains all the traffic information the adversary knows. *I.e.*, for every dishonest link $(v_i^{(t)}, v_j^{(t+1)})$ it contains the information whether there was traffic on that link during the t'th step or not.

Z : X and C' together determine whether the process described in Section 5.4 finds the crossover network G in the protocol or not. If we do, we let Z contain all the information available on links that do not belong to $\phi_{X,C'}(\boxtimes_G)$. *I.e.*, for every link $(v_i^{(t)}, v_j^{(t+1)})$ that does not belong to $\phi_{X,C'}(\boxtimes_G)$, it contains the information as to whether there was traffic on that link during the t'th step or not.

Notice that Z is correlated with X, Π and C'. Nevertheless, Fact B.2.2 tells us that $I(\Pi : C') \leq I(\Pi : C', Z)$. It would therefore suffice to show that $I(\Pi : C', Z) \leq \alpha(N)$. Now comes the crux of the argument, and we do it in detail.

Suppose the embedding strategy finds G in an execution P of the protocol. By Definition 5.4.2, all executions P' of the protocol that use the same set of vertices and agree with P outside $\phi_P(\boxtimes_G)$ result in the same embedding. As all edges revealed are outside $\phi_P(\boxtimes_G)$, the random variable Z has the same value in both cases. Also, C' has the same value in both cases as $\phi_P(\boxtimes_G)$ contains only honest edges. Thus, the adversary can not distinguish P from P'. As the a-priori probabilities of the executions P and P' are the same, both are equally likely from the adversary point of view. *I.e.*, any possible communication pattern on $\phi_P(\boxtimes_G)$ is equally likely.

Now, G is an ϵ-obscurant network. From the adversary point of view, any crossover is resolved to be identity with probability half, and a switch with probability half (because all possible communication patterns are equally likely), and so by the obscurant network properties $|(\Pi(1) \mid C', Z) - U_M|_1 \leq \epsilon$.

Using Claim 5.4.2 it follows that $\Pr_{c',z}[|(\Pi(1)|C'=c', Z=z) - U_M|_1 \geq \epsilon] \leq DM(1-f^4)^k = \epsilon$, when k is set to $log_{\frac{1}{1-f^4}}(\frac{DM}{\epsilon})$.

We now continue with standard manipulations. From Lemma 3.0.1 we see that $I(\Pi(1) : C', Z) \leq log(|\Lambda_N|) \cdot \sqrt{\epsilon} = O(TM^2\sqrt{\epsilon})$. Taking $\epsilon = \frac{\alpha^6(N)}{M^{12}}$, we receive $I(\Pi(1) : C', Z) \leq O(\frac{\alpha(N)}{M})$.

By Fact B.2.1,

$$I(C' : \Pi) \quad = \quad I(C' : \Pi(1)) + I(C' : \Pi(2)|\Pi(1)) + \ldots + I(C' : \Pi(M)|\Pi(1), \ldots, \Pi(M-1))$$

We can bound the j'th term $I(C' : \Pi(j) \mid \Pi(j-1), \ldots, \Pi(1))$ in this equation, by adding to the adversary the knowledge of the communication paths of the first $j-1$ players. We then see that we get a new game with only $M-j+1$ players. Our analysis from before shows that $I(C', Z : \Pi(j) \mid \Pi(j-1), \ldots, \Pi(1)) \leq O(\frac{\alpha(N)}{M})$. We therefore conclude that $I(C' : \Pi) \leq M \cdot O(\frac{\alpha(N)}{M}) \leq \alpha(N)$ as desired.

5.6 Unlinkability With Prior Information

In the general case the adversary knows that the actual communication that took place has a-priori distribution Π. In general, the adversary may use this knowledge to deduce things about the next to last layer, the one preceding it and so forth. Thus, the information the adversary sees flows both from bottom up (because the adversary knows who initiates messages, and follows whatever links he can), and from top down (because the adversary has some partial information about who sent who a message, and he follows links from top down). We note that we would like to deal with priors that have extremely low probability in a uniform world. *E.g.*, the adversary might know that residents of Kandahar tend to communicate with residents of Karachi. Such events have low probability in a uniform world.

One natural attempt for dealing with priors is to make use of Bayes formula. However, such a calculation gives very poor results when the prior has small probability in a uniform world. Another natural attempt, is bounding the amount of information the adversary gets from the prior alone, and from the communication alone and combining them. This approach is faulty as it is possible that two random variables A and B that convey no information about C, together reveal C, *e.g.*, if $C = A \oplus B$, where A and B are uniform.

To further clarify this issue, say G is a crossover network that β–obscures permutations. *I.e.*, if we start with the identity permutation, and randomize over the crossovers, the resulting distribution O is β close to uniform, $|O - U_{M!}|_1 \leq \beta$. Now say we have the prior knowledge that the resulting permutation is one from a subset P of possible permutations $P \subset S_M$. What we would like to have is that $|(O|P) - (U|P)|_1$ is small. However, if

- P maximizes the ℓ_1 distance, *i.e.*, $|O(P) - U(P)|_1 = \beta$, and,
- The probability P happens in O is the same as in the uniform case,

which is quite possible to arrange (in an abstract setting), then a simple calculation shows that $|(O|P) - (U|P)|_1 = \frac{M!}{|P|}\beta$, and so even when β is very small the distance in the presence of the prior can be arbitrarily close to 1. In a sense, the problem is that a fixed network might have a bad prior. The reason we can solve the problem in our protocol is that in our case the quantifiers are essentially reversed. *I.e.*, the adversary has to choose and fix a prior, and then the players make their random choices, effectively choosing a random network.

The way we technically show our protocol works is by concentrating on the *middle* layer. This is intuitively natural because the adversary knows the permutation at the beginning, and has partial information about the final permutation (given by the prior), but the middle layer seems to be masked by the random choices made throughout the protocol. We let $\Pi^{(T/2)}$ be the random variable whose value is the actual permutation that took place between the first and middle layer. We show that even in the prior knowledge scenario the adversary does not learn much about the middle layer, *i.e.*, that $I(C' : \Pi^{(T/2)}) \leq \beta$. To show that we give the adversary additional information so that to make the information flow only in one direction. Details follow.

Lemma 5.6.1. *Let Π be an arbitrary distribution. Suppose we run the protocol for $T = \Omega(\log(M)\log^2(\frac{M}{\alpha}))$ steps. Then $I(C' : \Pi^{(T/2)}) \leq \alpha$.*

Proof. We say a vertex $v^{(t)}$ from the t'th layer is associated with a vertex $w^{(T-t)}$ from the $T-t$'th layer, if the message that $v^{(t)}$ forwards eventually arrives at $w^{(T-t)}$. We also say the link (w, v) is associated with the link (v', w') if w is associated with w', and v is associated with v'.

For the proof, we give the adversary the extra knowledge about which vertex at level t is associated with which vertex at level $T - t$, for every $0 \leq t \leq \frac{T}{2}$, and we prove that even with that additional information the adversary does not know much about the middle layer. In Figure 5.4 we see all the additional information that is given to the adversary for the protocol network of Figure 4.2. We see that under this additional information the adversary gets to see M players playing our protocol for $T/2$ steps, and where a link $(v^{(t)}, v^{(t+1)})$ is honest iff both the link $(v^{(t)}, v^{(t+1)})$ and its associated link are honest. An example of such folding is given in Figure 5.5, where the network of Figure 4.2 is shown folded.

Thus, the only difference from the case of no prior knowledge is that now the probability each link is honest is f^2 rather than f. We therefore can use the theorem for no prior-knowledge

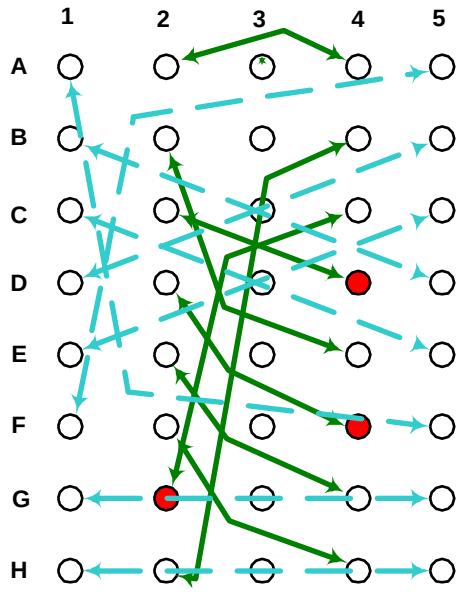

Figure 5.4: *The figure shows how the vertices of the protocol network in Figure 4.2 are matched to each other. We show that even with this additional information the adversary does not know much about the middle layer. The reader is encouraged to verify this matching based on the information given in Figure 4.2.*

and conclude that $I(C' : \Pi^{(T/2)}) \leq \alpha$ as desired. □

To complete the proof we show that since even given all what the adversary knows the adversary did not gain much information about the middle layer, it must be the case that the adversary did not gain much information about the last layer. *I.e.*,

Lemma 5.6.2. $I(C' : \Pi^{(T)}) \leq I(C' : \Pi^{(T/2)})$.

Proof. We represent the random variable C that contains the communication the adversary sees as $C = (C_1, C_2)$ where C_1 is the communication seen throughout the first $T/2$ steps, and C_2 is the communication seen throughout the last $T/2$ steps.

$$\begin{aligned} I((\Pi^{(T)} : C_1, C_2) &= I(\Pi^{(T)} : C_2) + I(\Pi^{(T)} : C_1|C_2) = I(\Pi^{(T)} : C_1|C_2) \\ &= I(f(\Pi^{(T/2)}, C_2) : C_1|C_2) \leq I(\Pi^{(T/2)} : C_1|C_2) \leq I(\Pi^{(T/2)} : C_1, C_2) \end{aligned}$$

The first equality and the last inequality are applications of the chain rule, Fact B.2.1.

To see the second equality, notice that $(C_2|\Pi^{(T)} = \pi)$ is the same distribution for all permutations π that are valid values of $\Pi^{(T)}$. This is because we can think of the protocol as if the

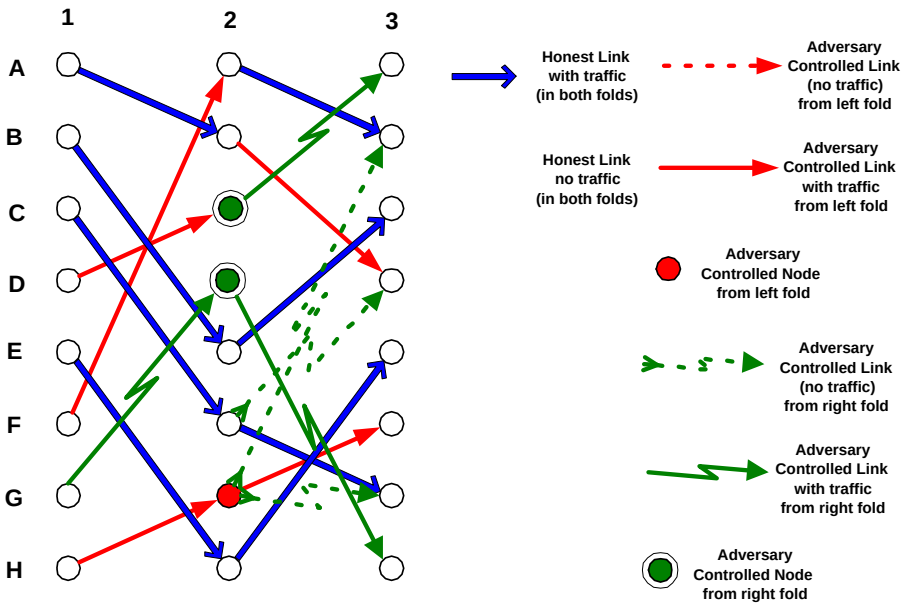

Figure 5.5: *The figure shows the folding of the protocol network in Figure 4.2 around its middle layer. In general we also reveal all the even layers. However, to keep the example small, the example shows folding when even layers are not revealed.*
The figure shows the first three layers of the protocol network of Figure 4.2, combined with additional adversary knowledge obtained from the folding:
1. Adversary knowledge of additional links with traffic is deduced from the folding. E.g., the traffic along the link from $(G,1)$ *to* $(D,2)$ *was deduced from the adversary controlled link with traffic from* $(F,4)$ *to* $(G,5)$ *in Figure 4.2, when taken along with the mapping* $(G,5) \mapsto (G,1)$ *and* $(F,4) \mapsto (D,2)$ *that we reveal to the adversary during the folding (Figure 5.4).*
2. Adversary knowledge of links with no traffic is also deduced from the folding. E.g., the fact that no traffic goes along the link from $(2,G)$ *to* $(3,B)$ *was deduced from the adversary controlled link with no traffic from* $(4,C)$ *to* $(3,B)$ *in Figure 4.2, when considered along with the folding information of Figure 5.4.*

players first pick $\pi \in \Pi^{(T)}$, then pick the top $T-1$ levels at random, and then complete the first layer to implement π. Thus, $I(\Pi^{(T)} : C_2) = 0$.

The crux of the argument is the first inequality. For it, we use the data-processing inequality, Fact B.2.5, and the probabilistic function $f(\sigma, c_2)$ that given $\sigma \in \Pi^{(T/2)}$ and $c_2 \in C_2$ chooses the permutation π with probability $\Pr(\Pi^{(T)} = \pi \mid \Pi^{(T/2)} = \sigma \wedge C_2 = c_2)$. The important thing to notice is that it suffices to know σ and c_2 alone to know the value of $f(\sigma, c_2)$, and we do not need to know c_1.

□

Remark. While we prove $I(C' : \Pi^{(T)}) \leq I(C' : \Pi^{(T/2)})$, it is not true that $I(C' : \Pi^{(T-1)}) \leq I(C' : \Pi^{(T/2)})$. *E.g.*, suppose prior information tells us that $\Pi^{(t)}$ is concentrated on the identity permutation. Then:

- $I(\Pi^{(T)} : C') = 0$.
- $\Pi^{(T-1)}$ is the uniform distribution over all permutations.
- Knowing C' and $\Pi^{(T)} = id$ usually reveals a lot of information about $\Pi^{(T-1)}$, and so $I(\Pi^{(T-1)} : C')$ is likely to be very large.
- We just proved that $I(\Pi^{(T/2)} : C')$ is very small.

and so it is quite possible that $I(\Pi^{(T-1)} : C') \geq I(\Pi^{(T/2)} : C')$, and in particular $I(\Pi^{(t)} : C')$ is not monotone in t.

One can then wonder what in our proof applies to $\Pi^{(T)}$ but does not apply to $\Pi^{(T-1)}$. Going over the proof one finds out that the equation $I(\Pi^{(T)} : C_2) = 0$ does not hold for $\Pi^{(T-1)}$, exactly for the reasons stated above. Indeed, if we pick $\pi \in \Pi^{(T-1)}$ we can choose the levels *below* it uniformly at random, but we have to complete the levels above it so that we get the constraints imposed by $\Pi^{(T)}$ (in our case we have to make it the identity permutation, so we are forced to choose π^{-1}). It therefore follows that the distributions $(C_2 | \pi^{(T-1)} = \pi)$ highly depend on π, and so the mutual information might be high.

Chapter 6

Applications

This paper has dealt with and is compared with communication protocols practiced and used even as these lines are being written. Giving no treatment to the connection with live networks and applications would miss a part of this paper's goals.

We illustrate three practical aspects of our protocol, related to both adversaries and users. We define a closed network as one where all nodes have the means to communicate directly with any other node (*i.e.* All nodes know addresses, numbers and keys as required by our protocol). After dealing with the subject of real adversaries of less limited power, we describe two applications that allow browsing for data on an exterior network (*e.g.* the Internet), and the interior closed network.

6.1 Real Adversaries

It is a simple reduction using the techniques of this paper to extend the adversary model and additionally allow the adversary to eavesdrop to the traffic on an honest link, 99% of the time, with the caveat that on a random 1% of the time, he fails to do so.

The results of this paper remain unchanged for this more powerful adversary model, as it only changes the path length needed to achieve unlinkability, according to the lower probability that a link is honest.

This seems a not-unreasonable model for communications in peer to peer networks over the Internet where (at least some) pairs of peers can communicate (at least some of the time) without being recorded.

6.2 Anonymous Web Surfing

Assume node A wants to retrieve data anonymously from Internet location x[1]. Node A randomly chooses a gateway node B from its closed network. A then sends the message $a =$ "$Please\ fetch\ data\ from\ x$" to node B. B browses location x on the Internet, and encrypts the returned data as answer b, sending it back to A. Figure 6.1 depicts an example usage of the protocol.

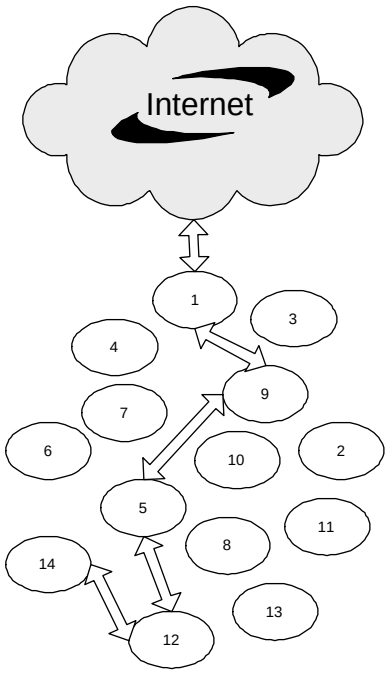

Figure 6.1: *Web browsing using our protocol. Node 14 tunnels its request through random nodes enroute to node 1 which serves as the selected gateway node. Node 1 then retrieves data from the Internet and returns it through the same path to node 14.*

A few technical details require special care when implementing Internet browsing using our protocol. As an example, the length of all messages (in both directions) must be uniform. The analysis found in [SSW+02] can serve as an introduction.

6.3 Privacy Preserving Data Exchange

It is now in our desire to allow browsing for information in a way that maintains unlinkability, with the identity of the data donor kept secret in addition. Following is a sketch of a new protocol that maintains donor privacy. All messages are sent using our protocol. In the scenario described, node A wants to retrieve item b from the network while remaining anonymous, as well as allowing the donor of the data to retain anonymity as well.

[1] *E.g. a site containing Alzheimer's disease symptoms list.*

In order to prove unlinkability in this scenario, we need to add an extra requirement to our unlinkability definitions from Chapter 3.

In previous definitions, it was naturally assumed that if the group of M message initiators contains adversary nodes, then unlinkability cannot be achieved. The following definition emphasizes the difference.

Definition 6.3.1. A protocol is *Fully Unlinkable* if it is unlinkable according to the definition in Section 3.2 with respect to any group of message initiators, including non-honest players.

We now show how to use the protocol from Chapter 4 to construct a network where *Full Unlinkability* may be achieved:

1. An arbitrary node (*e.g.* node number 1) is named BB and acts as a bulletin board for the entire system. This node receives requests from the entire system, yet does so in specific times only.
2. The initiator A picks a random intermediate node I.
3. A sends the message "Request item b to be placed at I" to BB.
4. BB stores the request in its memory.
5. Each node J in the system sends the message "Are there any new requests pending?" to BB in specific times.
6. BB concatenates any such requests into a long reply, and sends it to J.
7. When a certain J learns of a request for which it can supply the answer and wants to do so, it sends the message "Please store data item b" to I. b is the actual content of the item.
8. A requests b from I, and I either replies with it if it has it, or replies that it does not have it.

Remark. Correct timing of the messages is crucial in the above construction. Details were omitted to keep the presentation of this original idea simple.

Figure 6.2 illustrates one type of possible transaction using our new protocol. Proving the unlinkability properties of this new protocol is left for future work.

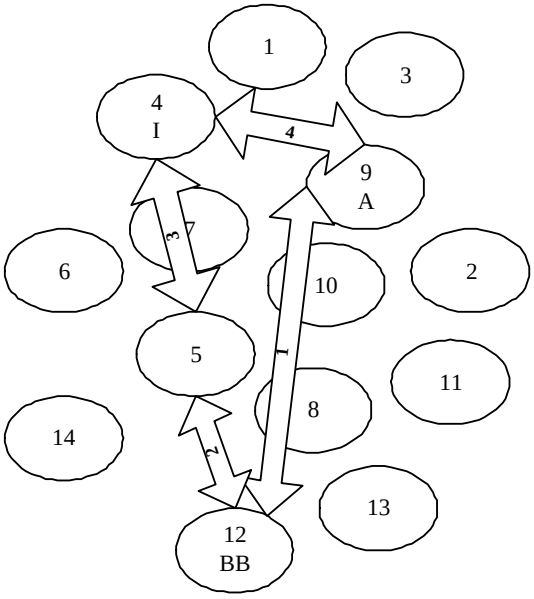

Figure 6.2: *Closed network browsing with donor privacy. Node 9 is the initiator A, node 12 is the bulletin board BB and node 4 is the intermediate node I.*
1. A places a request on BB. 2. Node 5 (as well as the entire network) sees the request on BB. 3. Node 5 answers A's request, placing the data on node I. 4. A fetches the data from I.

Chapter 7

Conclusions

This paper has dealt mainly with proving that Chaum's protocol, used for many years and considered anonymous is, in fact, anonymous.

We first examine the similarities and differences between our attack model and Chaum's attack model.

Are all links under adversary control? : The main difference between the two attack models is that in Chaum's attack model *no* communication link is honest, while in our attack model we allow 99% of the communication links to be controlled by the adversary, when the remaining 1% of the links are honest. The adversary can choose which of the links are under his control, but cannot change this set dynamically.

Time and time bound : There are several timing attacks on Chaum's system, discussed in [Ray01, RSG98]. Fixes involve a time bound. We include that time bound as an assumption on the system.

Malicious players : Chaum allows malicious nodes that may delete messages. Chaum's handling of malicious nodes, however, is basically the ability to "incriminate" a mix that "failed to properly process an item". We can also supply the same level of incrimination. However, in Section 7.1 we briefly show that this by itself does not guarantee unlinkability, neither in our system, nor in Chaum's system. In fact, Chaum's system does not even stand an adaptive adversary (see, *e.g.*, [Ray01], Sec 3.2).

During the course of the proof some new untouched aspects of the subject of anonymity were introduced and explored:

- A large survey of related work has allowed us to give clear and simple definitions for anonymity and adversaries. Proving those definitions are equivalent serves as a main contribution, as it allows us to measure and compare the quality of anonymity each type of system conveys.

- The significance of efficiency was illustrated with a comparison of many systems with ours. It was shown that one does not have to forfeit efficiency to achieve anonymity.

- The proof technique in the no prior information case, along with the definition of an obscurant network is interesting by itself, as it merges information theory into this area of study, simplifying analysis of protocols.

- Prior information was usually ignored in previous work. A surprising result is that anonymity can be achieved under a scenario in which the adversary has specific information regarding large parts of the communication.

- A large volume of work on anonymity deals with surfing the real world Internet without being identified. A bonus added here is the option of receiving information from a network, with its origin remaining private to the entire network, including the original requestor of information.

  The work presented here is far from complete. The next section introduces and partly discusses many issues to be solved in future work.

7.1 Future Work

Incomplete network graph : We assumed each node in the system has information about the name, address and public key of the rest of the nodes in the system. This assumption is hardly achievable in real world networks such as the Internet both because of the networks sizes, as well as because of their unstructured nature. Creating a provable anonymous protocol for this scenario is needed. It is quite easy to prove the our protocol does not and can not in its current form achieve anonymity against an adversary that knows the structure of such a network, due to possible timing attacks on it.

Section 1.2.3 describes the timing attack. This attack is very strong in our case of a non-complete network graph, as not every node is reachable from any other node within one hop. The message tagging attack also described in the same section can be used to map connections between nodes and discover the full map of the network.

Malicious behavior : Malicious adversaries are arbitrary, and in particular can inject, modify or delete messages. The security definition against malicious adversaries is identical to the non-malicious case, requiring low mutual information between collected data and the actual communication pattern. Achieving this in the malicious setting is much more complicated, however.

The ability to "incriminate" a mix that failed to properly process an item, does not, by itself, guarantee security. Say, for example, that M honest players send messages to M other players, and that K malicious players at the first level delete the messages sent to them. If the protocol goes on the adversary then learns how to associate as sets, the set of $M - K$ senders that their messages went through, and the set of $M - K$ receivers, thus learning quite a few bits of information.

A few possible solutions have been proposed in the past (see, *e.g.*, [RS93]). They include shutting down the entire network when such behavior is detected, and other similar drastic measures. These types of solutions are clearly not practical, and we see this direction of research as being of major importance for real advances in the field of anonymous communication.

Multi shot game : The anonymity we offer withstands only one shot of running our protocol. The question is whether it is possible to customize our protocol for scenarios where it is used multiple times.

Efficient secure broadcast : A few useful applications exist when using an unlinkable message protocol as a sub-layer. The application presented above currently uses an inefficient blackboard throughout the system, relying on its cooperation for advertising requests. An efficient way to securely broadcast a request to all participants preserving the privacy of the origin of request is required. A few solutions come to mind, such as secure gossiping. Their anonymity properties, however, remain yet to be proven.

Network management : Our protocol explicitly assumes that the network on which it runs is managed by a trustworthy outsider, as well as the assumption that the network is static most of the time. This is not the case (yet again) for real world applications. The problems of joins, leaves and key management must find a practical and proven secure solution before we put our system to the test.

Bibliography

[Abe99] M. Abe. Mix-networks on permutation networks. In *Advances in Cryptology - ASIACRYPT '99, International Conference on the Theory and Applications of Cryptology and Information Security, Singapore, November 14-18, 1999, Proceedings*, volume 1716 of *Lecture Notes in Computer Science*, pages 258–273, 1999.

[AH01] Masayuki Abe and Fumitaka Hoshino. Remarks on mix-network based on permutation networks. *Lecture Notes in Computer Science*, 1992:317–324, 2001.

[Alo01] N. Alon. Testing subgraphs in large graphs. In *42nd IEEE Symposium on Foundations of Computer Science*, pages 434–439, 2001.

[Ano] The anonymizer. http://anonymizer.com.

[BD03] Beimel and Dolev. Buses for anonymous message delivery. *JCRYPTOL: Journal of Cryptology*, 16, 2003.

[Cha81] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the Association for Computing Machinery*, 24(2):84–88, February 1981.

[Cha88] David Chaum. The Dining Cryptographers Problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.

[CT91] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley Series in Telecommunications. John Wiley & Sons, New York, NY, USA, 1991.

[DFM00] Roger Dingledine, Michael J. Freedman, and David Molnar. The free haven project: Distributed anonymous storage service. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*. Springer-Verlag, LNCS 2009, July 2000.

[EFG] Anonymous remailer information. http://anon.efga.org/Remailers.

[Fed01] Hannes Federrath, editor. *Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 25-26, 2000, Proceedings*, volume 2009 of *Lecture Notes in Computer Science*. Springer, 2001.

[FH] The free haven project. http://www.freehaven.net.

[GRS96] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Hiding routing information. In *Information Hiding*, pages 137–150, 1996.

[MP01] Dahlia Malkhi and Elan Pavlov. Anonymity without 'cryptography' (extended abstract). In *FC: International Conference on Financial Cryptography*. LNCS, Springer-Verlag, 2001.

[MW] Merriam-webster online dictionary. http://www.m-w.com.

[NC00] M. Nielsen and I. Chuang. *Quantum Computation and Quantum Information*. Cambridge, 2000.

[PK00] Andreas Pfitzmann and Marit Köhntopp. Anonymity, unobservability, and pseudonymity — A proposal for terminology. In Hannes Federrath, editor, *Designing Privacy Enhancing Technologies*, volume 2009 of *Lecture Notes in Computer Science*, pages 1–9, Berkeley, CA, USA, July 2000. Springer-Verlag, Berlin Germany.

[PW87] Andreas Pfitzmann and Michael Waidner. Networks without user observability. *Computers and Security*, 6(2):158–166, April 1987.

[Ray01] Jean-François Raymond. Traffic analysis: Protocols, attacks, design issues, and open problems. *Lecture Notes in Computer Science*, 2009:10–29, 2001.

[RR98] Michael K. Reiter and Aviel D. Rubin. Crowds: anonymity for Web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, November 1998.

[RS93] Charles Rackoff and Daniel R. Simon. Cryptographic defense against traffic analysis. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on the Theory of Computing*, pages 672–681, San Diego, California, 16–18 May 1993.

[RSG98] Michael G. Reed, Paul F. Syverson, and David M. Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications*, 16(4):482–494, May 1998.

[SGR97] Paul F. Syverson, David M. Goldschlag, and Michael G. Reed. Anonymous connections and onion routing. In *1997 IEEE Symposium on Security and Privacy*, pages 44–54, 1997.

[SSW+02] Qixiang Sun, Daniel R. Simon, Yi-Min Wang, Wilf Russell, Venkat Padmanabhan, and Lili Qiu. Statistical identification of encrypted web browsing traffic. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 19–30, Oakland, CA, May 2002. IEEE Computer Society, Technical Committee on Security and Privacy, IEEE Computer Society Press.

Appendix A

Simple Statistical Analysis of Traffic

We imagine the following scenario: in a computer network the nodes are divided into two types, "bad" and "good". All the content is also divided into the same two types. We assume that a good node always accesses good data, while bad nodes always access bad data. We also assume that all bad data is under control of an adversary, and that the network provider supplies some information about the users in each time step. Specifically, we assume that in each time step we receive the following information:

1. List of nodes that performed data access, either good or bad.
2. Number of bad data items accessed.

Putting things more rigorously on paper, we define the following:

- N — Number of nodes in network.
- B — Number of bad nodes in network. We assign $B = \beta \cdot N$.
- P — Number of nodes that access data items in each experiment. We assign $P = \alpha \cdot N$. The actual list of nodes that access the data is chosen randomly out of N in each experiment. This comes to simulate random access behavior from the nodes.
- M — Number of experiments in the game until final decision is given.

The rules of the experiment consist of:

- Game: We divide time into fixed length slices which we call experiments. In each experiment a batch of P accesses are made to data items. After each experiment, we supply an observer with the knowledge of who the P nodes were, and out of which, how many

queries were for bad items. The observer doesn't know who accessed bad items out of the P nodes.

- Observer: The observer keeps a counter for all nodes in the network, or a subset of them, on which he wants to tell whether they are bad or not. After each experiment the observer increments the counter of each of the nodes he observes which were part of the P nodes by the amount of bad requests which originated from the P nodes in the given experiment. The observer also knows the values of N and B.

- Final Cut: After M experiments the observer decides according to the counters of the observed nodes whether they are bad or not. A trivial condition is being used for deciding, and we describe it later.

- Question: How many experiments does the observer need to correctly recognize a bad node w.h.p., using the described algorithm?

Claim. *Given the above definitions for N, α and β, an observer can identify bad nodes with $1-\epsilon$ probability after $M = \Theta(\frac{N}{\sqrt{\epsilon}})$ experiments.*

Proof. We assign two random variables:

- X — The random variable giving the amount of addition to the counter of a bad node when it is a part of the P nodes.

  The value for this variable ranges from 1 to B. The probability distribution of this variable is hyper-geometric. Specifically:

$$Pr(X = k) = \frac{\binom{N-B}{P-k} \cdot \binom{B-1}{k-1}}{\binom{N-1}{P-1}}$$

- Y — Same variable, only for a good node. The value for this variable ranges from 0 to B, and the probability distribution is the following:

$$Pr(Y = k) = \frac{\binom{N-B-1}{P-k-1} \cdot \binom{B}{k}}{\binom{N-1}{P-1}}$$

The condition used for distinguishing good nodes from bad nodes is simple. After M experiments, the observer divides each counter held for each node by M. If the counter is closer to $\mathbb{E}(X)$ than to $\mathbb{E}(Y)$, then the node is declared as bad.

We would like to show next that the probability that a good node is declared bad is very small.

Let Y_i be the outcome of experiment $i = 1 \ldots M$ for a certain good node for which a counter is kept. We assign $S_M = Y_1 + \ldots + Y_M$.

The chances of the observer being wrong are:

$$Pr\left(S_M > M * \mathbb{E}(Y) + \frac{M}{2} \cdot (\mathbb{E}(X) - \mathbb{E}(Y))\right) =$$
$$Pr\left(\frac{S_M - M \cdot \mathbb{E}(Y)}{\sigma(Y) \cdot \sqrt{M}} > \frac{\sqrt{M}}{2 \cdot \sigma(Y)} \cdot (\mathbb{E}(X) - \mathbb{E}(Y))\right)$$

We now perform two steps. The first is to plug in $\mathbb{E}(X)$, $\mathbb{E}(Y)$, $\sigma(Y)$, while the second is making an assumption that N is large, meaning, *e.g.*, that $N \approx N - 1$ etc. We then require that the probability of being wrong is small:

$$Pr\left(S_M > M * \mathbb{E}(Y) + \frac{M}{2} \cdot (\mathbb{E}(X) - \mathbb{E}(Y))\right) \approx$$
$$Pr\left(\frac{S_M - M \cdot \mathbb{E}(Y)}{\sigma(Y) \cdot \sqrt{M}} > \frac{\sqrt{M} \cdot \sqrt{1-\alpha}}{2 \cdot \sqrt{\beta \cdot \alpha \cdot (1-\beta) \cdot N}}\right) \rightarrow$$
$$\frac{1}{\sqrt{2\pi}} \cdot \int_{\frac{\sqrt{M} \cdot \sqrt{1-\alpha}}{2 \cdot \sqrt{\beta \cdot \alpha \cdot (1-\beta) \cdot N}}}^{\infty} e^{-t^2} dt \leq$$
$$\frac{1}{\sqrt{2\pi}} \cdot 4 \cdot \frac{2^4 \cdot \beta^2 \cdot \alpha^2 \cdot (1-\beta)^2 \cdot N^2}{M^2 \cdot (1-\alpha)^2} \leq \epsilon$$

The second transition is an application of the central limit theorem. As $e^{-t^2} \leq t^{-4}$ for all $t > 0$, we have bounded the error probability of the observer in the first inequality.

We finally see that if $M \geq 8 \cdot \frac{\beta\alpha(1-\beta)}{1-\alpha} \cdot \frac{N}{\sqrt{\epsilon}}$ we fulfill our requirements. □

Numerical calculations show that the worst case scenarios, *i.e.*, the times when the largest numbers of M are needed, appear when $\beta = 0.5$. In addition, the larger the amount of communication taking place in every experiment, *i.e.*, the larger α is, the harder it is for the observer to identify bad nodes with low error probabilities. Figure A.1 shows the relation between M and α when at least 99% accuracy is required.

Taking example numbers for obtaining such a result, we fix $\alpha = 0.01$, $\beta = 0.1$, $N = 1,000,000$. In order to achieve 99% accuracy in such a case, roughly $M = 20,000$ experiments are needed. This means that around 200 messages on average are needed to be sent by each node. It should be noted that the approximation was done assuming that all nodes are being

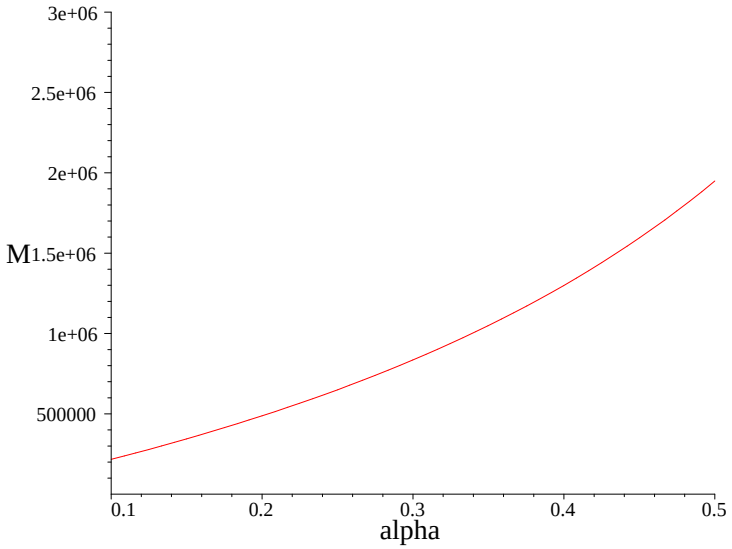

Figure A.1: Number of required experiments (M) versus experiment fraction of total number of nodes (α) when requiring 99% precision. β equals 0.1 and the number of nodes N equals $1,000,000$.

judged independently and at the same step. It is obvious that determining bad nodes in earlier stages makes things easier for the observer in identifying bad nodes later.

Appendix B

Information Theory and Its Usefulness

B.1 Notations

A discrete distribution D over Λ is a function $D : \Lambda \to [0..1]$ such that $\sum_{x\in\Lambda} D(x) = 1$. For $S \subseteq \Lambda$ we denote $D(S) = \sum_{s\in S} D(s)$. We identify a random variable A that takes values from Λ with the distribution A such that $A(x)$ is the probability the random variable A takes value x.

We measure distance between distributions defined over Λ with the ℓ_1 norm (also known as the variational distance):

$$|D_1 - D_2|_1 \stackrel{\text{def}}{=} \sum_{x\in\Lambda} |D_1(x) - D_2(x)| = 2 \max_{S\subseteq\Lambda} \ (\ D_1(S) - D_2(S) \ )$$

Let A and B be two events. We denote by $A \otimes B$ their product distribution, *i.e.*,

$$\Pr(A \otimes B = (a, b)) \quad = \quad \Pr(A = a) \cdot \Pr(B = b)$$

and by (A, B) their joint distribution, *i.e.*,

$$\Pr((A, B) = (a, b)) \quad = \quad \Pr(A = a \wedge B = b)$$

B.2 Some Information Theory

The entropy of a distribution D is

$$H(D) \quad \stackrel{\text{def}}{=} \quad \sum_{x\in\Lambda} D(x) \log(\frac{1}{D(x)})$$

The mutual information function is

$$I(A:B) \stackrel{\text{def}}{=} H(A) + H(B) - H(A,B)$$

when

$$H(AB) = \sum_{a \in A, b \in B} \Pr((A,B) = (a,b)) \log(\frac{1}{\Pr((A,B) = (a,b))})$$

The mutual information function can also be expressed as

$$I(A:B) = H(A) - H(A|B)$$

where

$$H(A|B) \stackrel{\text{def}}{=} \mathbb{E}_{b \in B} H(A|B=b)$$

We also define

$$I(X:Y|Z) \stackrel{\text{def}}{=} H(X|Z) + H(Y|Z) - H(X,Y|Z)$$

The mutual information function respects the following chain rule:

Fact B.2.1. *[CT91]* $I(A : B_1 \dots B_n) = I(A : B_1) + I(A : B_2|B_1) + \dots + I(A : B_n \mid B_1 \dots B_{n-1})$

Also, the mutual information function is monotone in the following sense:

Fact B.2.2. *[CT91, NC00] For every random variables* A, B *and* C, $I(A : BC) \geq I(A : B)$.

The entropy function, and hence the mutual information function, are continuous functions. This is captured in:

Fact B.2.3. *([NC00], Box 11.2, page 512) Assume* D_1, D_2 *are distributed over* Λ. *Let* $\delta \stackrel{def}{=} |D_1 - D_2|_1$ *and assume* $\delta \leq \frac{1}{e}$. *Then*

$$|H(D_1) - H(D_2)| \leq \delta \log(|\Lambda|) + \delta \log(\frac{1}{\delta})$$

As $I(A : B) = H(A) + H(B) - H(AB)$ we see that:

Fact B.2.4. *Assume* (A, B) *is distributed over* Λ. *Let* $\delta \stackrel{def}{=} |(A, B) - (A', B')|_1$ *and assume* $\delta \leq \frac{1}{e}$. *Then*

$$|I(A : B) - I(A' : B')| \leq 3 \cdot (\delta \log(|\Lambda|) + \delta \log(\frac{1}{\delta}))$$

One way to think about the mutual information function $I(A:B)$ is that it measures the amount of information A gets to know about B, or saying it differently, how much knowing the value of A affects our knowledge of the distribution B. An important property of the mutual information function is that A can not increase this knowledge without communicating with B. This is captured in the data processing inequality:

Fact B.2.5. *(The data processing inequality) (see [NC00], section 11.2.4) For every deterministic or probabilistic function f,*

$$I(f(A,C):B|C) \quad \leq \quad I(A:B|C)$$

We stress that f can be determined by A and C alone. Also, f might be probabilistic, i.e., uses random coins.

Finally, we define the relative entropy function of two random variables A and B distributed over the same domain:

Definition B.2.1. [CT91, NC00] Let A and B be distributed over the same domain Λ. Their relative entropy is

$$D(A||B) \quad = \quad \sum_{x\in\Lambda} \Pr(A=x)\cdot\log(\frac{\Pr(A=x)}{\Pr(B=x)})$$

The relative entropy is not symmetric, *i.e.*, $D(A||B)$ might be different from $D(B||A)$. The relative entropy is always non-negative, is zero iff $A=B$ and respects the following inequality:

Fact B.2.6. *[CT91]* $D(A||B) \geq \frac{1}{2\ln 2}|A-B|_1^2$.

A simple fact is that

Fact B.2.7. *[CT91, NC00]*
$I(A:B) \;=\; D((A,B)||A\otimes B)$.

B.3 Some More Motivation

The above definitions give rise to the question of what the mutual information definition is good for. One would like to say that revealing additional information about the communications can only help the adversary. The problem arises when formally stating this property. When defining anonymity, Rackoff and Simon make use of the ℓ_1-norm, see Chapter 3. A technical problem that arises is that the one-norm is not monotone, whereas mutual information is.

To illustrate this issue, consider the following experiment: Toss a 0/1 uniform probability coin 4 times, where $R_i, i = 1, \ldots, 4$, is the result of the i'th coin toss, and $S = \sum_{\ell=1}^{4} R_\ell$. We would like to say that an adversary that knows (R_1, R_2, R_3) "knows" more about S than an adversary that knows only (R_1, R_2). One intuitive way of stating this is that the ℓ_1 distance between the distribution known to the adversary and the binomial distribution for S gets bigger when the adversary learns more. However, this is not necessarily true when we consider specific values of these random variables. Imagine that the adversary already knows $R_1 = R_2 = 0$ and learns that $R_3 = 1$.

$$\begin{aligned}
|(S|R_1 = 0, R_2 = 0) - S|_1 &= \sum_{\ell=0}^{4} \left| \begin{array}{l} \text{Prob}(S = \ell) - \\ \text{Prob}((S|R_1 = 0, R_2 = 0) = \ell) \end{array} \right| \\
&= 0.875; \\
|(S|R_1 = 0, R_2 = 0, R_3 = 1) - S|_1 &= \sum_{\ell=0}^{4} \left| \begin{array}{l} \text{Prob}(S = \ell) - \\ \text{Prob}((S|R_1 = 0, R_2 = 0, R_3 = 1) = \ell) \end{array} \right| \\
&= 0.75.
\end{aligned}$$

Thus, the "little birdy" principle, that revealing additional information about the communications can only help the adversary, does not always work with this definition.

An alternative way of stating that the adversary knows more about S with R_3 than without R_3 is to say that $I(S : (R_1, R_2)) \leq I(S : (R_1, R_2, R_3))$:

$$\begin{aligned}
I(S : (R_1, R_2)) &= H(S) + H((R_1, R_2)) - H((S, R_1, R_2)) \\
&= \sum_{\ell=0}^{4} \binom{4}{\ell} / 2^4 \cdot \log_2 \left(2^4 / \binom{10}{\ell} \right) \\
&\quad + \log_2(4) - 4 \cdot \sum_{\ell=0}^{2} \binom{2}{\ell} / (4 \cdot 2^2) \cdot \log_2 \left((4 \cdot 2^2) / \binom{2}{\ell} \right) \approx 0.53 \\
I(S : (R_1, R_2, R_3)) &= H(S) + H((R_1, R_2, R_3)) - H((S, R_1, R_2, R_3)) \\
&= \sum_{\ell=0}^{4} \binom{4}{\ell} / 16 \cdot \log_2 \left(16 / \binom{4}{\ell} \right) \\
&\quad + \log_2(8) - 8 \cdot \sum_{\ell=0}^{1} \binom{1}{\ell} / (8 \cdot 2) \cdot \log_2 \left((8 \cdot 2) / \binom{1}{\ell} \right) \approx 1.03
\end{aligned}$$

Here, the "little birdy" principle works fine. This is always true (Fact B.2.2, Section B.2).

In fact, what we learn from that is that *on average* the little birdy principle works with ℓ_1 norm. *I.e.*, while some values may pull us to the wrong direction, most values push towards the right direction. In essence, this is the meaning of the equivalence stated in Lemma 3.1.