Design and Analysis of Efficient Anonymous-Communication Protocols

Aaron Johnson
Department of Computer Science
Yale University
aaron.johnson@yale.edu

Abstract

Research into protocols for anonymous communication in networks has yielded many designs and a wide range of functionality and performance options. Of these, only a few have been successful in practice. The experience in fielded systems emphasizes the following criteria for designing anonymous-communication protocols: communication functionality, anonymity, latency, and message complexity. Therefore in this thesis we explore understanding and improving the performance of anonymous-communication protocols according to these criteria.

The approach we take is to formally model such protocols and rigorously analyze them according to precise definitions of our criteria. The protocols are designed to operate in an adversarial environment, and we would like to analyze the properties they possess against all likely adversaries. It is difficult to perform a convincing analysis of this kind informally, and the ability to perform such an analysis is a main benefit of our theoretical approach.

The first subject of our analysis is the onion-routing protocol. Onion routing is the most successful protocol for anonymous communication in practice, and it has seen several publicly-available implementations. However, the anonymity provided by onion routing has not received much rigorous analysis. Rather, the protocol has been studied mostly as a practical solution for anonymous communication. Therefore, in the first half of the thesis, we model the network environment and the onion-routing protocol, and we analyze the resulting anonymity properties.

Our results show that onion routing provides unsatisfactory anonymity against a reasonable adversary that controls some fraction of the network. The exact anonymity depends on the size of the network and the behavior of the users, but it faces an upper limit that depends on the size of the adversary. Therefore, in the last half of the thesis, we consider two ways to get past this limit: trust information and a new prototol design.

We first suppose that users have external trust information about the network that helps them avoid parts that are controlled by the adversary. Then we consider using this information to improve the performance of onion routing. Under a model of trust, we come up with practical and provable recommendations.

Next we consider a new protocol that avoids the main flaw in onion routing: timing attacks. An adversary that can observe traffic coming into the network and flowing out of the network can use timing patterns in the traffic, natural or induced, to link a user with his destination. We give a protocol that involves explicit

timing instructions and redundancy, and prove that it provides arbitrarily good anonymity that grows with the user population. Finally, this protocol requires adding some artificial delays that depend on the behavior of the network. Therefore, to estimate the magnitudes of these delays in practice, we measure timing in a live anonymity network. Our results suggest that the added delays are likely to result in a small constant factor increase over onion routing.

Contents

Chapter 1

Introduction

Protocols for anonymous communication allow users to communicate while hiding their identities from one another and from third parties. Research into such protocols has yielded many different designs, which offer a variety of functionality and performance, and which are often designed under different assumptions about the network environment. The functionality provided by anonymous-communication protocols spans the spectrum of possibilities - protocols exist that provide one-way communication, that allow a user to send one message and receive a single response, and that provide two-way connection-oriented communication. The performance of these protocols varies widely as well. There are high-latency protocols that can take arbitrarily long to deliver a message, and there are low-latency protocols that guarantee no more than a constant-factor increase in delivery time over direct communication. Some protocols require n^2 messages to be sent for every anonymous message, where n is the number of users, and others require a number of extra messages that is only a small constant. Protocols exist for very strong and very weak adversaries, for synchronous and with asynchronous communication, for complete and for incomplete communication graphs.

We would like to design such protocols with strong anonymity guarantees but without incurring high communication overhead or much added latency. The designs that have been proposed meet these goals to varying degrees.

Of the many anonymous-communication design proposals, onion routing [29] has had notable success in practice. Several implementations have been made [29, 65, 18], and there was a similar commercial system, Freedom [7]. As of August 2007, the most recent iteration of the basic design, Tor [18], consists of over 1000 routers, collectively processing an average of 110MB/s, with an estimated total user population of 200,000 [69]. Because of this popularity, we believe it is important to improve our understanding of the protocol.

Onion routing is a practical anonymity-network scheme with relatively low overhead and latency. It provides two-way, connection-based communication and does not require that the destination participate in the anonymity-network protocol. These features make it useful for anonymizing much of the communication that takes place over the Internet today, such as web browsing, chatting, and remote login.

Onion routing was designed as a practical protocol, however, and the precise anonymity and performance it provides are not clear. Systems for communication anonymity generally have at most one of two desirable properties: provable security and practicality. Systems that one can prove secure require assumptions that make them impractical for most communication applications. Practical systems are ultimately the ones we must care about, because they are the ones that will actually be used. However, their security properties have not been rigorously analyzed or even fully stated. This is no surprise, because practical anonymity systems have been deployed and available to study for perhaps a decade, while practical systems for communications confidentiality and/or authenticity have been in use almost as long as there have been electronic communications. It often takes a while for theory and practice to catch up to each other. Thus, in the first half of this thesis we formally model and analyze the anonymity and performance of onion routing. In Chapters 2 and 4 we formally model the network environment and describe the onion-routing protocol. In Chapters 5 and 6 we perform analyses of the anonymity provided by onion routing.

Our results indicate that onion routing does not provide very adequate anonymity. In the last half of this thesis we describe two protocols that attempty to improve the anonymity of onion routing while maintaining its efficiency properties. In Chapter 7, we consider the case that users have some outside knowledge about which routers in the network are more trustworthy and less likely to compromise anonymity. We give new strategies for path-selection in onion routing to take advantage of this trust information. In Chapter 8, we describe a new protocol that provides arbitrarily good anonymity, assuming that enough users use the system at roughly the same time. This protocol requires adding some delays that depend on the network, an dso we perform a measurement study of the timing properties of the Tor network to evaluate what kind of performance the system might achieve in practice.

Chapter 2

Models and Definitions

2.1 Introduction

The first step towards rigorously analyzing and designing anonymity protocols is to build an appropriate model of the environment in which they will operate. Our environment consists of the network, users, and the adversary. We make our modeling decisions in order to best capture the relevant features of the environment as they have been revealed in the research and practice of low-latency anonymous communication. The second step is to choose the criteria with which to evaluate the protocols and make precise definitions of them. The criteria that we will consider are functionality, anonymity, latency, and message complexity.

2.2 Models

We give a basic model here that will serve as a basis for all of the protocols we describe and analyze. Features will be added to this model in later chapters as we seek better understanding of and designs for anonymous communication.

2.2.1 Input/Output Automata

We express our model and protocols using input/output automata [41]. Several different formalisms have been used to formalize anonymous-communication protocols, including an ACP-style process algebra [43], the universally composable framework [8], and a run-and-systems logical semantics [30]. The main obstacle to useful anonymous communication is preventing *traffic analysis*, that is, hiding traffic patterns of traffic from the adversary. I/O automata allow us to focus on this problem rather than the cryptographic aspects of the protocols. Also, I/O automata are quite general and allow us to model asynchronous computation and communication.

2.2.2 Executions

We use standard notions of *execution* and *fairness*. An execution is a possible run of the network given its initial state. Fairness for us means that any message an automaton wants to send will eventually be sent and every sent message is eventually received.

We introduce the notion of a *cryptographic* execution. This is an execution in which no agent sends a control message encrypted with active keys it doesn't possess before it receives that message. We restrict our attention to such executions, and must require our encryption operation to prevent an attacker from outputting a control message, with more than negligible probability, when it is encrypted with keys he doesn't possess. This is reasonable because we can easily create a ciphertext space that is much larger than a limited control message space P. Note that this precludes the use of public key encryption to encrypt the packets because such messages can easily be constructed with the public keys of the routers.

Definition 1. *An* execution *is a sequence of states of an IO automaton alternating with actions of the automaton. It begins with an initial state, and two consecutive states are related by the automaton transition function and the action between them. Every action must be* enabled*, meaning that the acting automaton must be in a state in which the action is possible at the point the action occurs.*

Because there are no internal actions in the automata, all actions are message sends or message receives. Frequently, we will treat an execution as a sequence of actions, because the states are implicit from these and the initial states.

Definition 2. *A finite execution is* fair *if there are no actions enabled in the final state. Call an infinite execution* fair *if every output action that is enabled in infinitely many states occurs infinitely often.*

Definition 3. *An execution is* cryptographic *if an agent sends a message containing $\{p\}_{k_1,\dots,k_i}$ only when it possesses all keys $k_1, \dots, k_i$, or when for the largest j such that the agent does not possess k_j, $1 \leq j \leq i$, the agent has already received a message containing $\{p\}_{k_1,\dots,k_j}$.*

2.2.3 Network

We model onion routing as a fully-connected asynchronous network of I/O automata. The network is composed of FIFO channels. There is a set of users U, a set of routers R, and a set of destinations D. Let $N = U \cup R \cup D$. The term *agent* refers to any element of N.

The network delivers messages in the message space $\mathcal{M}$. $\mathcal{M}$ must satisfy the following properties:

1. It contains the integers, that is, $\mathbb{Z} \subseteq \mathcal{M}$.

2. Message tuples are valid messages, that is, $\mathcal{M}^k = \mathcal{M}$, $k \in \mathbb{N}$.

Our network model provides asynchronous communication between every pair of agents. Asynchrony is an important feature, because synchronizing the actions of different users can be used to make them indistinguishable to an adversary and is a key technique used by anonymous communication protocols [19]. The network model does not include, for now, any more explicit notion of time than the ordering on actions.

2.2.4 Users

We model user behavior very simply. At the start of the protocol each user wants to send a message to one destination and to receive one message back.

User behavior is an important feature of our model because it has a big effect on anonymity. When there are very few users that communicate, for example, good anonymity is simply not achievable. As another example, if each user only communicates with a single unique destination, and the adversary knows this, then the user cannot communicate anonymously unless the adversary can't tell if the user is communicating at all.

Our model of user behavior is an intentionally simple example of bidirectional communication. We will show that many of the common anonymous-communication protocols do not have satisfactory performance even for such basic user behavior. We will then focus on protocols with better performance, which will then allow us to consider the effect of more complicated, and more realistic, users.

There are many features of user behavior that are not addressed by our basic model. Users may choose destinations in a predictable way. They may contact multiple destinations simultaneously. Users and destinations may participate in bidirectional streams of messages rather than single message/response pairs. Instead of communicating just once, users may participate in many communication sessions over time.

These ways in which plausible user behavior differs from our model may be significant in an anonymous communication protocol. An adversary might be able to learn a user's frequent communication partners if he chooses them somewhat predictably over time. Moving to stream communication may open up additional opportunities for an adversary to determine the destination. The simple user behavior model already leads to interesting problems in evaluating and designing anonymous communication protocols. Also, we do examine more complicated user behavior in Chapters 6 and 8. These show that our simple model is robust in some ways.

2.2.5 Adversary

The adversary in our system is a set of users and routers $A \subseteq U \cup R$. The adversary is *active* in the sense that the automata running on members of A are completely arbitrary. We call an agent *a compromised* if $a \in A$. We assume that the adversary knows the number of users, routers, and destinations.

Note that we do not allow the adversary to compromise any of the destinations. It does seem possible in several situations this could be the case - for example, communication with a known hostile destination or a destination that is under surveillance. However, the case that this doesn't happen generally captures protocol performance and protocol design challenges, and therefore that is the case we focus on. Because our design criteria will require that all protocols use just one router to communicate with the destination, the case that this router is compromised presents the same situation as the case that the destination itself is compromised. Therefore many of our analyses extend naturally to the case that the adversary can compromise destinations.

We choose to make the adversary *local*, in that he controls a subset of the network. In particular, this limits him to only a partial observation of network activity. This contrasts with a *global* adversary, popular in analyzing mix networks, that can observe the entire network. This adversary is too strong for the large network such as the Internet. This is a very helpful restriction for designing effective protocols; we can confuse a local adversary just by making sure that some network events occur outside of his view. A global adversary precludes this, and seems to require protocols that have high resource requirements.

We also make the adversary active in the network. This is opposed to a *passive* adversary, that is, an adversary that obeys the protocol and just makes inferences from his observations, which is considered often in the literature. A passive adversary is interesting for several reasons: it is a trivial behavior for the adversary to implement, it is hard to detect and protect against because it is behaving the same as a normal user, and there are mechanisms (for example, zero-knowledge proofs) that help enforce correct adversary behavior. However, an active adversary is much more realistic, and ideally a protocol should work against such an adversary.

Because we are trying to keep some knowledge hidden from the adversary, it is important to specify the knowledge that the adversary starts with. Our basic knowledge assumption is that the adversary knows the number of users that exist in the network. This is already probably too strong, in that it gives the adversary more information than he likely has. However, in the protocols we consider, the adversary can make good statistical guesses about the number of users, routers, and, in some cases, destinations, just by observing the operation of the protocol at the router he controls.

2.3 Criteria

The criteria that we will use to evaluate anonymous communication protocols are functionality, anonymity, latency, and message complexity. The popularity of onion routing can be attributed in part to the fact that it has reasonable performance according to all of these criteria, while the many competing protocols usually perform badly in at least one. Note, however, that good performance on each of these may not be sufficient for a useful protocol in many of the situations in which anonymous communication might be applied. For example, one might want the user to be *unobservable*, that is, that the adversary is unable to determinine whether or not the user is communication with anyone at all. More weakly, one might want the protocol to be hard to distinguish from other popular protocols on the same network. However, it does seem that for many of the Internet applications that onion routing is used to anonymize - for example, Web browsing, file sharing, chat, and email - any protocol that succeeds against these four criteria is likely to be useful.

2.3.1 Functionality

The communication functionality that we want our protocol to provide is to, first, deliver a message from a user to a given destination and, second, to deliver a possible response message from the destination to the user. Formally, the protocol must implement the operation AnonymousMessage(d,m). It takes as inputs a destination $d \in D$ and a message $m \in \mathcal{M}$, and it returns a number $n \in \mathbb{N}$ that identifies that use of the operation. After a user u calls AnonymousMessage, some router $r \in R$ in the network must eventually deliver the message $[g(n), m]$ to d, where $g(n) : \mathbb{N} \rightarrow \mathbb{N}$ is an injective map. Then, if at some point d sends the message $[g(n), m']$ to r, $m' \in \mathcal{M}$, the network must eventually deliver the message $[n, m']$ to u. Note that the protocol is only required to implement these actions when all the agents correctly execute it.

Our functionality requirement is deliberately simple to avoid complicating the analysis of protocols according to all of our criteria. Also, it allows us to directly compare the protocols for anonymous communication in the literature, which generally implement this basic operation.

We require that only one router communicate with the destination for a given communication from the user. This requirement is motivated by the observation in practice that it is difficult to get hosts to support a special protocol for anonymous communication. Therefore, the destination communicates using standard point-to-point protocols with one router that acts as an anonymous proxy for the user. Removing this requirement would make it easier to design effective protocols, but it is a real-world constraint that we feel is important to adopt.

2.3.2 Anonymity

There are multiple types of anonymity involved in anonymous communication and several plausible ways of measuring each one. Indeed, there is a growing body of research solely concerned with identifying and comparing various notions of anonymity. Our approach is to pick an important type of anonymity and study how to implement it very well under our chosen criteria, evaluating anonymity with some reasonable metric. We are mostly not concerned with examining diferent types of anonymity and determining the best metrics.

Therefore we emphasize one type of anonymity evaluated according to one metric. We will focus primarily on *relationship anonymity*, that is, on how well the adversary can determine the communication partners of a user. To measure this, we primarily use the probability that the adversary assigns to the user's destination after observing the protocol, under probabilistic extensions our model.

Types

In general, any action that a user can perform can be performed anonymously. This means that if the adversary can tell that a certain action was performed, then he cannot tell which user was the actor. We will also consider the user anonymous, in the case that adversary cannot tell whether or not the action was performed.

There are several types of anonymity associated with network communication that may be of interest:

- *Sender anonymity*: A message has sender anonymity if the adversary cannot tell which user sent that message.

- *Receiver anonymity*: A message has receiver anonymity if the adversary cannot tell which destination a specific message is sent to.

- *Relationship anonymity*: A user and destination have relationship anonymity if the adversary cannot tell which destination a user communicates with.

In general, there is a weak logical relationship about these three notions - if user u and destination d have relationship anonymity, then, for every message m sent by u, m has sender anonymity or receiver anonymity. Otherwise, for some message the adversary would be able to determine that it was sent by u to d. If u and d do not have relationship anonymity, we cannot say anything about the anonymity of their messages. It is possible, for example, that every message has both sender and receiver anonymity, and yet the adversary can still determine that u sent at least one of them to d.

We focus on relationship anonymity because it has been the primary goal of fielded anonymous communication systems. Also, providing receiver anonymity is made fundamentally difficult by our functionality requirement that one final router send a user's message to its destination. This final router may be compromised and will observe the message and its recipient. Conversely, providing sender anonymity seems very close to providing relationship anonymity, because in our model every user communicates with exactly one destination. Therefore for u to have relationship anonymity, we only need there to be one message with sender anonymity for which u is among the possible senders and that is to a destination other than d.

Metrics

There are several reasonable metrics with which to measure anonymity. Unlike the primitives in cryptography, which have similar information-hiding goals to anonymous communication and are in several cases closely related [34], the proper definitions for anonymity have not been settled. This seems to be because the kinds of very strong guarantees that cryptographic constructions can satisfy have not been achieved by useful constructions for anonymous communication. The way to measure the anonymity that existing constructions do achieve is then less clear.

In an anonymous-communication protocol, the adversary makes a series of observations of the network. There is a set of possible executions, including the execution that actually occurred, that are consistent with his observations. Metrics that are functions on this set are *possibilistic* in that they only consider which situations are possible. For a given agent and action, they include

- *Boolean*: whether it is possible that the agent didn't perform the action
- *Anonymity set*: the number of actors that may have performed the action

Possibilistic anonymity is a very weak notion of anonymity, as the adversary may have some external information that allows him to conclude that certain executions are very unlikely. To model this uncertainty, we can add a probability measure to executions. Then the adversary can infer a conditional probability distribution on executions given his adversary. Metrics that take into account the probability measure are *probabilistic*. For a given agent and action, the probabilistic metrics that have been proposed include

- *Agent probability*: the conditional probability that the adversary assigns to the event that the correct agent performed the action
- *Entropy*: the entropy of the conditional distribution over agents
- *Min-entropy*: the min-entropy of the conditional distribution over agents
- *Variation distance*: the variation distance between a prior distribution over agents for a given action and a posterior distribution (that is, the conditional distribution)

Probabilistic metrics take into account both the randomness that is a part of many anonymous-communication protocols and predictable behavior of users that is reasonably modeled as probabilistic. From the metrics that have been proposed, we select agent probability for most of the thesis. It emphasizes how well a protocol hides specific actions by a specific user, and it is relatively easy to work with mathematically.

We use different definitions of anonymity for the different models and protocols that we consider. Therefore will defer for now giving precise definitions of anonymity.

2.3.3 Latency

The latency of a protocol measures how long it takes for a message to be received by its intended recipient after it is sent. Because our basic model lacks an explicit notion of time, we give here a definition of latency based on the longest sequence of message that must be sent and received after a message is sent until the destination finally receives it.

For a given execution, the actions form a partial order in which, for actions a_1 and a_2, $a_1 \prec a_2$ if a_1 and a_2 are actions of the same agent's automaton and a_1 was performed by that automaton before a_2. We define the latency of an anonymous message in a given protocol execution to be the length of the longest chain in the partial order from the action in which a user sends the anonymous message (that is, calls AnonymousMessage) to the action in which the message is sent. If no such chain exists, we define the latency to be infinity. Then we can define the general latency of a protocol.

Definition 4. *Let $\mathcal{E}$ be the set of fair, cryptographic executions when all agents run the given protocol. The* latency *of a protocol is the minimum over all executions $\alpha \in \mathcal{E}$ of the protocol of the latency of α.*

We take the smallest latency over all executions because it capture the latency inherent in the protocol. If we were to take the largest possible latency, executions that include long sequences of messages unrelated to delivering the anonymous message could yield unrealistically large values.

Our definition of latency assumes that all agents faithfully run the protocol. Our model of the adversary allows him to run abitrary programs, however. This is an important issue, because a protocol in which the adversary can easily delay or deny the delivery of messages is not very usable. If we assume that the adversary's goal is to deanonymize communication, however, assuming that he employs strategies that are most disruptive to message delivery may overstate the problem. Therefore we separately consider latency under adversarial misbehavior.

Definition 5. *Let $\mathcal{E}_A$ be the set of fair, cryptographic executions when the honest agents run the given protocol and the adversary runs A. The latency against A is the minimum over all executions $\alpha \in \mathcal{E}_A$ of the latency of α. The* adversarial latency *is the maximum over all adversaries A of the latency against A.*

It is true that delaying message delivery may aid in anonymization. If users repeatedly attempt to send an anonymous message until it is successfully delivered, the adversary can try to prevent delivery until attempts at deanonymization succeed. This issue is explored by Borisov et al. in [6]. We do not consider anonymity under this kind of user behavior.

2.3.4 Message Complexity

In general, we care about the total communication requirements of a protocol. Important metrics include the number of messages, message sizes, and the message contention [41]. These metrics can be considered for the system overall or for individual agents. We want our protocols to have reasonable communication requirements for each agent, and so we are interested in the per-agent metrics. Moreover, for the protocols we consider, message size and contention are not a problem in theory or practice. Therefore we will only consider the number of messages sent by the agents in the protocol.

Definition 6. *Let $\mathcal{E}$ be the set of fair, cryptographic executions of a protocol when all agents run the protocol. The* message complexity *of the protocol is the minimum over all $\alpha \in \mathcal{E}$ of the number of messages sent in α.*

Similar to defining latency, we choose the minimum over executions because we want the message complexity to represent the fundamental resource requirements of the protocol.

As before, we cannot assume that the adversary faithfully runs the protocol, and we want to consider by how much his malicious behavior can increase the message complexity of the protocol.

Definition 7. *Let $\mathcal{E}_A$ be the set of fair, cryptographic executions when the honest agents run the given protocol and the adversary runs A. The message complexity against A is the minimum over all executions $\alpha \in \mathcal{E}_{\mathcal{A}}$ of the number of messages sent in α. The* adversarial message complexity *is the maximum over all adversaries A of the message complexity against A.*

Chapter 3

Related Work

3.1 Formalization of Anonymous Communication Protocols

Numerous papers have informally discussed the security of the design of onion routing and related systems, as well as theoretical and experimentally demonstrated attacks. There have been numerous formalizations of anonymous communication. However, formal analyses have primarily been of systems other than onion routing, *e.g.*, DC nets and Crowds.

Recent papers have formalized systems similar to onion routing but without persistent circuits. Camenisch and Lysyanskaya [8] prove that the cryptography their protocol uses doesn't leak any information to nodes in the path other than the previous and next nodes, but leave open what anonymity it provides. This question is answered in part by Mauw et al. [43], who formalize a similar connectionless protocol in an ACP-style process algebra and, under a possibilistic definition of anonymity, show that it provides sender and receiver anonymity against a global passive adversary. Cryptography is dealt with using high level assumptions, similar to our work. Their model and analysis has much in common with this paper, but it does differ in several important ways. First, the protocol they investigate is connectionless: each data "onion" stores the identities of the routers it will pass through. This is significantly different from onion routing, which is circuit-based. Second, the analysis is done with respect to a passive adversary, which exhibits only a subset of the behavior of an active adversary. Third, in their model agents choose destinations asynchronously and the adversary must take into account every onion he has seen when trying to break anonymity. In our model all agents choose a single destination, which gives the adversary more power. In some ways, our work extends theirs, and several of the differences noted here appear in [43] as suggestions for further research.

Ours is not the first formalization of anonymous communication. Earlier formalizations used CSP [58], graph theory and possible worlds [33], and epistemic logic [67, 30]. These earlier works focused primarily on formalizing the high-level concept of anonymity in communication. For this reason, they applied their formalisms to toy examples or systems that are of limited practical application and can only provide very strong forms of anonymity, e.g., dining-cryptographers networks. Also, with the exception of [30], they have at most a limited ability to represent probability and probabilistic reasoning. We have focused in [24] on formalizing a widely deployed and used, practical, low-latency system.

Halpern and O'Neill [30] give a general formulation of anonymity in systems that applies to our model. They describe a "runs-and-systems" framework that provides semantics for logical statements about systems. They then give several logical definitions for varieties of anonymity. It is straightforward to apply this framework to the network model and protocol that we give in [24]. Our possibilistic definitions of sender anonymity, receiver anonymity, and relationship anonymity then correspond to the notion of "minimal anonymity" as defined in their paper. The other notions of anonymity they give are generally too strong and are not achieved in our model of onion routing.

Even earlier formalizations of substantial anonymous communication systems [8, 43] have not been directly based on the design of deployed systems and have focused on provability without specific regard for applicability to an implemented or implementable design. Also, results in both of these papers are for message-based systems, rather than systems that create a cryptographic circuit prior to communication. Thus, while illuminating, they are not likely to be applicable to low-latency communications, and, despite the title of [8], are not analyses of onion routing.

Other works have presented probabilistic analysis of anonymous communication [56, 60, 72, 12, 14, 42, 37] and even of onion routing [65]. The work of Shmatikov and Wang [62] is particularly similar to ours. It calculates relationship anonymity in mix networks and incorporates user distributions for selecting destinations. However, with the exception of [60], these have not been formal analyses. Also, whether for high-latency systems such as mix networks, or low-latency systems, such as Crowds and onion routing, many of the attacks in these papers are some form of intersection attack. Several of these papers set out frameworks for making that more precise. In particular, [12], [14], and [42] constitute a progression towards quantifying how long it takes (in practice) to reveal traffic patterns in realistic settings.

We are not concerned herein with intersection attacks. We are effectively assuming that the intersection attack is done. The adversary already has a correct distribution of a user's communication partners. We are investigating the anonymity of a communication in which a user communicates with one of those partners in the distribution. This follows the anonymity analyses performed in much of the literature [37, 43, 56, 65], which focus on finding the source and destination of an individual communication. Our analysis differs in that we take into account the probabilistic nature of the users' behavior.

3.2 Improved Anonymous-Communication Protocols

In Chapters 7 and 8, we present two new protocols for anonymous communication. Our goal in each case is to improve anonymity while maintaining efficiency in latency and message complexity.

3.2.1 Trust

Chapter 7 explores improving onion routing when users have outside knowledge of which routers to trust. Trust has many meanings and applications in computer security [22, 64, 5, 1, 9, 61, 57, 15]. Much of the literature is concerned in one way or another with propagation or transfer of trust from where it is to where it needs to be. Our concern is not with the transfer of trust information but with what it means in the context of onion routing and how to make use of it. We consider how trust associated with network nodes or links might be used to protect (or reveal) information that would undermine the anonymity of communicants.

3.2.2 Preventing Timing Attacks

Timing attacks are a major challenge in low-latency anonymous communication [38]. They have been observed in some of the earliest low-latency systems [2], including initial versions of onion routing [29]. These attacks are also closely related to traffic analysis in mix networks [54].

In a passive timing attack, the adversary observes timing patterns in a network flow, and then correlates them with patterns in other traffic that it observes. If the adversary is able to observe both the user and the destination, he can thereby link the two. The ability of the adversary to perform this correlation has been experimentally demonstrated several times [74, 38, 52, 50, 3].

A solution to passive timing attacks is to get rid of identifying patterns in the traffic by padding and delaying it. The drawbacks to such an approach are added latency and bandwidth overhead. Our protocol relies on the existence of some acceptable and effective padding scheme. Constant-rate padding, in which traffic is sent at a constant rate by filling in the gaps with dummy packets, is probably the most obvious such scheme. It has appeared multiple times in the literature [65, 27, 38]. Levine et al. [38] propose a "defensive dropping" mechanism which adds dummy packets at the start of the circuit and drops them at various routers before the end. This reduces the correlation between any patterns in the incoming streams and patterns in the outgoing streams. Shmatikov and Wang [63] propose a variable-rate padding scheme. In their scheme, packets from the user are forwarded with little delay, and dummy packets are added by the intermediate routers according to a probability distribution on the packet inter-arrival times. Wang et al. [71] describe a link-padding scheme for low-latency systems, but their system is designed for a situation in which the adversary is not active and the destination participates in the protocol. This situation does not reflect the kind of Internet communications that have proven useful and that we target.

All of these schemes are vulnerable to an adversary that actively delays packets from the user. Yu et al. [73] show that this can be done in a way that makes it difficult for the user or the system to detect that the attack is occurring. According to the results of Shmatikov and Wang, their variable-rate scheme of inserting padding packets throughout the network provides the most protection against such an attack. However, in the scenario we are designing for, the final destination does not participate in the protocol. Therefore there must be one last router in the system that provides the point of contact to the destination. This final router can identify the dummy packets, and therefore can observe timing patterns that the adversary created earlier in the stream.

Simply delaying packets that pass directly through adversarial routers isn't the only active timing attack that has been demonstrated. Murdoch and Danezis [49] show that in onion routing the adversary can actively add delay patterns to the data by sending bursts of traffic through a router. This can be used to determine the routers on a given circuit. A queuing scheme has been suggested by McLachlan and Hopper [45] to mitigate this problem. Fu et al. [26] describe how the presence of a flow on a router can also be determined by ping times to the router, and suggests the simple countermeasure of treating ping packets equally. We do not address such congestion attacks in this paper, and it is outside of our model. Borisov et al. [6] look at the case that the adversary doesn't just delay, but drops packets in a denial-of-service (DoS) attack aimed at forcing users to move to circuits that the adversary can deanonymize. This is an easy behavior to observe, although proving the culprit may be difficult, and the authors suggest that reputation mechanisms can be used to limit such attacks. Such an attack was also discussed by Dingledine et al. [19]. Mittal and Borisov [47] examine the use of active attacks to subvert the host lookup mechanism in the Salsa and AP3 peer-to-peer protocols. This isn't a timing attack, but they uncover a tradeoff offered by the protocol between preventing active and passive attacks in the lookup that is very similar to the tradeoff we examine in our use of redundancy.

Hopper et al.[32] explore several attacks based on the network latency between hosts. In particular, they show the latencies from multiple hosts to a user can be very identifying. The user in our protocol communicates with several routers as a first hop, and in light of this attack we took care not to allow the adversary to infer these latencies.

One key feature of our protocol is the use of a layered mesh to provide redundancy. The use of redundancy for several purposes has been also explored in previous protocols. Syverson [66] suggests using router "twins," pairs of routers that share the same key, to provide a backup in case a router fails. Two redundancy schemes to manage failures, K-Onion and Hydra-Onion, are proposed by Iwanik et al. [35]. In the K-Onion scheme, onions are encoded in a way that provides more than one option for the next hop to route around host failures. In the Hydra-Onion scheme, onions get forwarded to two routers at every hop instead of one. The purpose of this scheme is to prevent an active attack in which the adversary kills messages from a known user and observes which destination stops receiving traffic. Redundancy to support failure is not our goal, and such schemes are in some ways complementary to our own. However, the redundancy in our protocol does protect against honest node failures as well as malicious ones. Our analysis, however, only considers malicious node failures. The aim of the Hydra-Onion scheme is similar to ours, although they only analyze the ability of the adversary to kill a connection. Morever, the scheme is designed for an adversary that controls links, and not one that controls routers.

Another critical feature of our protocol is the use of explicit timing to coordinate the users and routers. This is similar to the timing instructions of Stop-and-Go mixes [37]. Such mixes are given a time window within which the packets must arrive, and they delay forwarding by an exponentially-distributed amount of time. Although the techniques are similar, this scheme is designed for mix networks and not stream communication, and this scheme does give the adversary some slack time within which a timing signature could possibly be placed. Moreover, the lack of any redundancy means that any slowdowns within the network, even of benign origin, can quite easily kill a connection. The protocol is also susceptible to DoS attacks.

The timing approach we take is also similar to the use of synchronicity in mix networks by Dingledine et al. [19]. They describe synchronous batching on free routes and show that it typically provides better anonymity than cascades. Our scheme can be viewed as low-latency synchronous batching, In their case the synchronicity is provided by long delays in the mix network, while in our case synchronicity is provided by the padding schemes. They are able to disregard active attacks in which messages are dropped or delayed, because each mix is able to wait for enough time that any missing packets must be maliciously withheld. They rely on reputation systems [20] to eventually find and punish such action. With low-latency communication, on the other hand, we cannot assume that such delays will not happen. Moreover, our solution does not rely on the use of an external reputation mechanism. This is an advantage, because reputation mechanisms in general only detect large-scale misbehavior and must tolerate smaller-scale targeting of users.

Chapter 4

Formalizing Onion Routing

4.1 Introduction

Low latency and other performance characteristics of Tor can be demonstrated experimentally; anonymity-preserving properties cannot. Also, even with careful design, vulnerabilities can persist. The initial Tor authentication protocol had a cryptographic-assumption flaw that left it open to man-in-the-middle attacks. The revised authentication protocol was then proven to be free of such flaws [28]. As Tor is increasingly relied upon for sensitive personal and business transactions, it is increasingly important to assure its users that their anonymity will be preserved. Long-established components of such assurance in system security include a formal model, proving security guarantees in that model, and arguing that the model captures essential features of the deployed system. These are what we provide in this paper.

An onion-routing network consists of a set of onion routers and clients. To send data, a client chooses a sequence of routers, called a *circuit*, and constructs the circuit using the routers' public keys. During construction, a shared symmetric key is agreed upon with each router. Before sending data, these keys are used to encrypt each packet once for each router in the circuit and in the reverse of the order that the routers appear in the circuit. Each router uses its shared key to decrypt the data as it is sent down the circuit so it is fully decrypted at the end. Data flowing up to the client has a layer of encryption added by each onion router, all of which are removed by the client. The layered encryption helps hide the data contents and destination from all but the last router and the source from all but the first router. The multiple encryption and decryption also makes it harder for an observer to follow the path the data takes through the network.

Anonymity has not yet been rigorously proven of onion routing. We thus propose a formal model of onion routing based on the Tor protocol with which analyze the anonymity it provides.

4.2 Automata

We give the automata descriptions for the users and routers that are based on the Tor protocol [18]. We have simplified the protocol in several ways. In particular we do not perform key exchange, do not use a stream cipher, have each user construct exactly one circuit to one destination, do not include circuit teardowns, eliminate the final unencrypted message forward, and omit stream management and congestion control. Section 5.3.7 discusses the effects of changing some of these features of our protocol.

We use circuit identifiers to simplify the definitions and proofs of anonymity in our protocol. Circuit identifiers mimic the effects of a timing attack in onion routing. They have the added advantages of making it clear when this power is used. Theorem 5 shows that they are easily simulated by an adversary, and therefore they do not affect the anonymity of the protocol.

We assume that each router-and-user pair shares a set of secret keys; however, the router does not know which of its keys belong to which user. This separates, for now, key distribution from the rest of the protocol. We assume that all keys in the system are distinct. Let K be the keyspace. The triple (u, r, i) will refer to the ith key shared by user u and router r.

Let P be the set of control messages that are used in the protocol, and $\bar{P}$ be the extension of P by encryption with up to l keys. We assume that $\bar{P} \subseteq \mathcal{M}$. The control messages will be tagged with a link identifier and circuit identifier when sent, so let the protocol message space be $M = \mathbb{N} \times \mathbb{N} \times \bar{P}$. We denote the encryption of $p \in P$ using key k with $\{p\}_k$, and the decryption with $\{p\}_{-k}$. For brevity, the multiply encrypted message $\{\{p\}_{k_1}\}_{k_2}$ will be denoted $\{p\}_{k_1,k_2}$. Brackets will be used to indicate the list structure of a message (*i.e.* $[p_1, p_2, \ldots]$).

During the protocol each user u iteratively constructs a circuit to his destination. u begins by sending the message $\{\text{CREATE}\}_{k_1}$ to the first router, r_1, on his circuit. The message is encrypted with a key, k_1, shared between u and r_1. r_1 identifies k_1 by repeatedly trying to decrypt the message with each one of its keys until the result is a valid control message. It responds with the message CREATED. (Note that this is different from the implemented Tor protocol, in which the CREATE message would be encrypted with the public key for r_1 rather than one of the shared keys it holds.) Given a partially-constructed circuit, u adds another router, r_i, to the end by sending the message $\{[\text{EXTEND}, r_i, \{\text{CREATE}\}_{k_i}]\}_{k_{i-1},\ldots,k_1}$ down the circuit. As the message gets forwarded down the circuit, each router decrypts it. r_{i-1} performs the

CREATE steps described above, and then returns the message $\{\text{EXTENDED}\}_{k_{i-1}}$. Each router encrypts this message as it is sent back up the circuit.

Link identifiers are used by adjacent routers on a circuit to differentiate messages on different circuits. They are unique to the adjacent pair. Circuit identifiers are also included with each message and identify the circuit it is traveling on. They are unique among all circuits.

The user automaton's state consists of the sequence of routers in its circuit, the destination, a number that identifies its circuit, and a number that indicates the state of its circuit. The user automaton runs two threads, one to extend a circuit that is called upon receipt of a message and the other to start circuit creation that is called at the beginning of execution. We express these in pseudocode rather than IO automata, but note that the state changes in a particular branch occur simultaneously in the automaton. $k(u, c, b)$ refers to the key used by user u with router c_b in the bth position in circuit c. The automaton for user u appears in Automaton 1.

Automaton 1 User u

```
$c \in \{(r_1, \dots, r_l) \in R^l | \forall_i r_i \neq r_{i+1}\}$; init: arbitrary    ▷ User's circuit
$d \in D$; init: arbitrary    ▷ User's destination
$i \in \mathbb{N}$; init: random    ▷ Circuit identifier
$b \in \mathbb{N}$; init: 0    ▷ Next hop to build
procedure START
    SEND($c_1$, $[i, 0, \{\text{CREATE}\}_{k(u,c,1)}]$)
    $b = 1$
end procedure
procedure MESSAGE($msg$,$j$)    ▷ $msg \in M$ received from $j \in N$
    if $j = c_1$ then
        if $b = 1$ then
            if $msg = [i, 0, \text{CREATED}]$ then
                $b{+}{+}$
                SEND($c_1, [i, 0, \{[\text{EXTEND}, c_b, \{\text{CREATE}\}_{k(u,c,b)}]\}_{k(u,c,b-1),\dots,k(u,c,1)}]$)
            end if
        else if $b < l$ then
            if $msg = [i, 0, \{\text{EXTENDED}\}_{k(u,c,b-1),\dots,k(u,c,1)}]$ then
                $b{+}{+}$
                SEND($c_1, [i, 0, \{[\text{EXTEND}, c_b, \{\text{CREATE}\}_{k(u,c,b)}]\}_{k(u,c,b-1),\dots,k(u,c,1)}]$)
            end if
        else if $b = l$ then
            if $msg = [i, 0, \{\text{EXTENDED}\}_{k(u,c,b-1),\dots,k(u,c,1)}]$ then
                $b{+}{+}$
                SEND($c_1, [i, 0, \{[\text{DEST}, d, HELLO]\}_{k(u,c,b-1),\dots,k(u,c,1)}]$)
            end if
        end if
    end if
end procedure
```

The router automaton's state is a set of keys and a table, T, with a row for each position the router holds in a circuit. Each row stores the previous and next hops in the circuit, identifying numbers for the incoming and outgoing links, and the associated key. There is only one thread and it is called upon receipt of a message. In the automaton for router r, we denote the smallest positive integer that is not being used on a link from r to q or from q to r as $minid(T, q)$. The automaton for router r appears in Automaton 2.

Automaton 2 Router r

```
$keys \subseteq K$, where $|keys| \geq |U| \cdot \lceil \frac{l}{2} \rceil$; init: arbitrary        ▷ Private keys
$T \subset N \times \mathbb{N} \times N \times \mathbb{Z} \times keys$; init: $\emptyset$        ▷ Routing table
procedure MESSAGE($[i, n, p], q$)        ▷ $[i, n, p] \in M$ received from $q \in N$
    if $[q, n, \emptyset, -1, k] \in T$ then        ▷ In link created, out link absent
        if $\exists_{s \in R-r, b \in P} p = \{[\text{EXTEND}, s, b]\}_k$ then
            SEND($s, [minid(T, s), b]$)
            $T = T - [q, n, \emptyset, -1, k] + [q, n, s, -minid(T, s), k]$
        else if $\exists_{d \in D} p = \{[DEST, d, b]\}_k$ then
            SEND($d, [minid(T, d), b]$)
            $T = T - [q, n, \emptyset, -1, k] + [q, n, d, minid(t, d), k]$
        end if
    else if $[s, m, q, -n, k] \in T$ then        ▷ In link created, out link initiated
        if $p = \text{CREATED}$ then
            $T = T - [s, m, q, -n, k] + [s, m, q, n, k]$
            SEND($s, [i, m, \{\text{EXTENDED}\}_k]$)
        end if
    else if $\exists_{m>0}[q, n, s, m, k] \in T$ then        ▷ In and out links created
        SEND($s, [i, m, \{p\}_{-k}]$)        ▷ Forward message down the circuit
    else if $[s, m, q, n, k] \in T)$ then        ▷ In and out links created
        SEND($s, [i, m, \{p\}_k]$)        ▷ Forward message up the circuit
    else
        if $\exists_{k \in keys} p = \{\text{CREATE}\}_k$ then        ▷ New link
            $T = T + [q, n, \emptyset, -1, k]$
            SEND($q, [i, n, \text{CREATED}]$)
        end if
    end if
end procedure
```

Chapter 5

Possibilistic Anonymity in Onion Routing

5.1 Introduction

We give precise definitions of anonymity of onion routing as formalized in Chapter 4. Using these, we provide necessary and sufficient conditions for anonymity to be provided to a user in our model. It should be noted that we do not analyze the validity of the cryptography used in the protocol and instead base our proofs on some reasonable assumptions about the cryptosystem.

We only consider possibilistic anonymity here. An action by user u is considered to be anonymous when there exists some system in which u doesn't perform the action, and that system has an execution that is consistent with what the adversary sees. The actions for which we consider providing anonymity are sending messages, receiving messages, and communicating with a specific destination. More refined definitions of anonymity, [59, 16], incorporate probability. We apply such a definition to our system in Chapter 6, by defining a probability measure over executions or initial states.

The main result we show is that the adversary can determine a router in a given user's circuit if and only if it controls an adjacent router, with some other minor conditions. In particular, the adversary can determine which user owns a circuit only when the adversary controls the first hop. The set of users which have an uncompromised first hop form a sender "anonymity set," among which the adversary cannot distinguish. Similarly, the adversary can determine the last router of a circuit only when it controls the last router or the penultimate router. Such circuits provide receiver anonymity. Also, a user and his destination have relationship anonymity when he has receiver anonymity or his sender anonymity set includes another sender with a destination that is different or unknown to the adversary.

The first-hop/last-hop attack is well-known [65], but we state it in full detail and show that, in a reasonable formal model, it is the only attack that an adversary can mount. Also, our results persist with or without some of the nontrivial design choices, such as multiple encryption and stream ciphers. This doesn't imply that these features are unnecessary – they may make attacks more difficult in practice and meet other goals such as data integrity, but it does illuminate their effect on the security of the protocol.

5.2 Anonymity Analysis

Definition 8. *A* protocol configuration $C : U \rightarrow \{(r_1, \ldots, r_l, n) \in R^l \times D \times \mathbb{N}_+ | \forall_i r_i \neq r_{i+1}\}$ *maps each user to the circuit, destination, and circuit identifier in his automaton state.*

For the remainder of this chapter, we will refer to a protocol configuration simply as a *configuration.*

The actions that we want to be performed anonymously are closely related to the circuits that the users try to construct during an execution and their destinations. Therefore, in order to prove that certain actions are performed anonymously in the network, we show that the adversary can never determine this circuit and destination information. This is a possibilistic notion of of anonymity. We do this by identifying classes of adversary-indistinguishable configurations.

Because $i \in N$ only sees those messages sent to and from i, an execution of a configuration C may appear the same to i as a similar execution of another configuration D that only differs from C in the circuit positions and destinations that are not adjacent to i and in circuit identifiers that i never sees. To be assured that i will never notice a difference, we would like this to be true for all possible executions of C. These are the fair cryptographic executions of C, and likewise the executions of D should be fair and cryptographic. We will say that these configurations are indistinguishable if, for any fair cryptographic execution of C, there exists a fair cryptographic execution of D that appears identical to i, *i.e.* in which i sends and receives what appear to be the same messages in the same order.

Agent i's power to distinguish among executions is weakened by encryption in two ways. First, we allow a permutation on keys to be applied to the keys of encrypted or decrypted messages in an execution. This permutation can map a key from any router other than i to any other key of any other router other than i, because i can only tell that it doesn't hold these keys. It can map any key of i to any other key of i, because

i doesn't know for which users and circuit positions its keys will be used. Second, i cannot distinguish among messages encrypted with a key he does not possess, so we allow a permutation to be applied to control messages that are encrypted with a key that is not shared with i. This second requirement must be justified by the computational intractability of distinguishing between encrypted messages with more than a negligible probability in our cryptosystem.

Definition 9. *Let D_A be a relation over configurations indicating which configurations are indistinguishable to $A \subseteq N$. For configurations C and C', $C \sim_{D_A} C'$ if for every fair cryptographic execution α of C, there exists some action sequence β such that the following conditions hold with C' as the initial state:*

1. *Every action of β is enabled, except possibly actions done by members of A.*

2. *β is fair for all agents, except possibly those in A.*

3. *β is cryptographic for all agents.*

4. *Let Ξ be the subset of permutations on the active keyspace $U \times R \times \lceil \frac{l}{2} \rceil$ such that each element restricted to keys involving $a \in A$ is a permutation on those keys. We apply $\xi \in \Xi$ to the encryption of a message sequence by changing every list component $\{p\}_{(u,r,i)}$ in the sequence to $\{p\}_{\xi(u,r,i)}$.*

   Let Π be the subset of permutations on $\bar{P}$ such that, for all $\pi \in \Pi$,

   (a) π is a permutation on the set $\{\{p\}_{k_1,\ldots,k_i}\}_{p \in P}$

   (b) $\pi(\{p\}_{k_1,\ldots,k_i,k_a}) = \pi(\{p\}_{k_1,\ldots,k_i})$ when k_a is shared by the adversary

   We apply $\pi \in \Pi$ to a message sequence by changing every message $\{p\}_{k_1,\ldots,k_i}$ in the message sequence to $\pi(\{p\}_{k_1,\ldots,k_i})$.

   Then there must exist $\xi \in \Xi$ and $\pi \in \Pi$ such that applying ξ and π to the subsequence of α corresponding to actions of A yields the subsequence of β corresponding to actions of A.

If $C \sim_{D_A} C'$, we say that C is *indistinguishable* from C' to A. It is clear that an indistinguishability relation is reflexive and transitive.

5.2.1 Anonymity

The sender in this model corresponds to the user of a circuit, the receiver to the destination of the circuit, and the messages we wish to communicate anonymously are just the circuit control messages. The circuit identifiers allow the adversary to link together all the messages initiated by a user and attribute them to a single source. (Recall that in our model, users open a single circuit to a unique destination at one time.) Therefore sender anonymity is provided to u if the adversary can't determine which circuit identifier u is using. Similarly, receiver anonymity is provided to d for messages from u if the adversary can't determine the destination of the circuit with u's identifier. Also, relationship anonymity is provided to u and d if the adversary can't determine u's destination.

Definition 10. *User u has* sender anonymity *in configuration C with respect to adversary A if there exists some indistinguishable configuration C' in which u uses a different circuit identifier.*

Definition 11. *Destination d has* receiver anonymity *on user u's circuit, in configuration C, and with respect to adversary A, if there exists some indistinguishable configuration C' in which a user with u's circuit identifier, if one exists, has a destination other than d.*

Definition 12. *User u and destination d have* relationship anonymity *in configuration C if there is some indistinguishable configuration C' in which the destination of u is not d.*

5.3 Indistinguishable Configurations

Now we will show that sometimes the adversary cannot determine the circuit, destaniotn, or circuit identifier. More specifically, an adversary can only determine which user, router, or destination occupies a given position in a circuit when the adversary controls it or a router adjacent to it on that circuit. Also, when the adversary controls no part of a circuit it cannot determine its identifier.

In order to do this, we must show that, given a pair of configurations (C, C') that are indistinguishable by these criteria, for every execution of C there exists an execution of C' that appears identical to the adversary. To do this we will start with a fair cryptographic execution of C, describe how to transform it, and prove that this transformed sequence forms a fair, cryptographic, and indistinguishable execution of C'. We will also show that a pair of configurations that are distinguishable by the described criteria allow no such transformation.

5.3.1 Message Sequences

To start, we observe that, in spite of the arbitrary actions of the adversary, the actions of the uncompromised users and routers are very structured. The protocol followed by the user and router automata defines a simple sequence of message sends and receives for every circuit. A user or router will only send messages from the part of such a sequence consisting of its actions.

The user automaton gives this subsequence for users. It consists of messages between the user and the first router on its circuit, and is parameterized by the user u, the circuit c, the destination d, and the circuit identifier. We will refer to this sequence as $\sigma_U(u, c, d, j)$. Because the user automaton ignores the circuit identifier on received messages, we use an asterisk to indicate that any value is valid. Let $\sigma_U(u, c, d, j)$ be:

Step	From	To	Message
1	u	c_1	$[j, 0, \{\text{CREATE}\}_{k(u,c,1)}]$
2	c_1	u	$[*, 0, \text{CREATED}]$
3	u	c_1	$[j, 0, \{[\text{EXTEND}, c_2, \{\text{CREATE}\}_{k(u,c,2)}]\}_{k(u,c,1)}]$
4	c_1	u	$[*, 0, \{\text{EXTENDED}\}_{k(u,c,1)}]$
$1 + 2i$	u	c_1	$[j, 0, \{[\text{EXTEND}, c_{i+1}, \{\text{CREATE}\}_{k(u,c,i+1)}]\}_{k(u,c,i),\ldots,k(u,c,1)}]$
$2 + 2i$	c_1	u	$[*, 0, \{\text{EXTENDED}\}_{k(u,c,i),\ldots,k(u,c,1)}]$
$2 \leq i < l$			
1+2l	u	c_1	$[j, 0, \{[\text{DEST}, d, \text{HELLO}]\}_{k(u,c,l),\ldots,k(u,c,1)}]$

Lemma 1. *For user u to be enabled to send a message in an action sequence under configuration C, the the following conditions must be satisfied:*

1. *The send appears in $\sigma_U(u, C(u))$.*

2. *The prefix of $\sigma_U(u, C(u))$ ending before the send has appeared in the sequence.*

3. *This prefix is the longest such prefix to appear.*

Proof. Every message sent by Automaton 1 appears in $\sigma_U(u, C(u))$. This proves that (1) is satisfied.

We show inductively that (2) is satisfied. Assume that (1)-(3) hold for the sends that appear in step $1 + 2(i - 1)$, $0 < i \leq l$. Given some message that u is enabled to send, by (1) it appears in some step $1 + 2i$ in $\sigma_U(u, C(u))$. Examining where this message is sent in Automaton 1, we observe that step $2i$ must have occurred. Also, we observe that the automaton state variable b must equal i. For b to be set to i, the message in step $1 + 2(i - 1)$ must have been sent, because b is only incremented after that event. By our inductive assumption, when that send occurred, steps 1 to $2(i - 1)$ had occurred. Putting these together, we get the prefix in $\sigma_U(u, C(u))$ before step $1 + 2i$, showing (2). The inductive base case is trivially true, because the prefix of step 1 is empty.

Now, to show (3), we just observe that once step $1+2i$ occurs, b is incremented. Because b is never decremented, the condition guarding the send message of step $1+2i$ is never again satisfied. □

Similarly, the router automaton defines the action sequence that a router performs during the creation of a circuit. A different sequence exists for every router r, user u, circuit position $1 \leq i \leq l$, circuit c, destination d, and link identifiers $m, n, p \in \mathbb{N}$. We will denote a particular sequence $\sigma_R(r, c, d, u, i, m, n)$. Frequently we will drop parameters that we don't care about, for example, referring to $\sigma_R(r, c, d, u, i)$ when the specific link identifiers don't matter, and may abuse this notation by treating it as one sequence rather than a family of sequences. We use $k(u, c, i)$ as before.

If $i < l$, the sequence $\sigma_R(r, c, d, u, i, m, n)$ is:

Step	From	To	Message
1	c_{i-1}	r	$[j_1, n, \{\text{CREATE}\}_{k(u,c,i)}]$
2	r	c_{i-1}	$[j_1, n, \text{CREATED}]$
3	c_{i-1}	r	$[j_2, n, \{[\text{EXTEND}, c_{i+1}, s]\}_{k(u,c,i)}]$
4	r	c_{i+1}	$[j_2, m, s]$
5	c_{i+1}	r	$[j_3, \text{CREATED}]$
6	r	c_{i-1}	$[j_3, \{\text{EXTENDED}\}_{k(u,c,i)}]$

If $i = l$, the sequence $\sigma_R(r, c, d, u, i, m, n)$ is:

Step	From	To	Message
1	c_{i-1}	r	$[j_1, n, \{\text{CREATE}\}_{k(u,c,i)}]$
2	r	c_{i-1}	$[j_1, n, \text{CREATED}]$
3	c_{i-1}	r	$[j_2, n, \{[\text{DEST}, d, s]\}_{k(u,c,l)}]$
4	r	c_{i+1}	$[j_2, m, s]$

Using the σ_R sequences, we can characterize which messages a router can send at any point in an action sequence. Let α be a finite execution, and τ_i be the length i prefix of some sequence $\sigma \in \sigma_R(r)$. We say that τ_i has *occurred* in α when τ_i is the longest prefix of σ that appears as a subsequence of α, and, at the point at which step 1 (4) occurs in α, n (m) must be the smallest number not yet in r's table as an entry to or from c_{i-1} (c_{i+1}), the agent that sent (received) the message in step 1 (4).

Lemma 2. *For r to be enabled to send a message μ at the end of α, one of three cases must apply for some $\sigma \in \sigma_R(r)$:*

1. *The message is part of the circuit-building protocol. Sending μ is the $(2i)^{th}$ step in σ, $1 \leq i \leq 3$. Then τ_{2i-1} must occur in α and τ_{2i} must occur in the execution created when μ is appended to α.*

2. *The message is a forward down the circuit.* $\mu = [m, p]$. *r is to be sending μ to c_{i+1}. σ has occurred in α. The link identifiers to c_{i-1} and c_{i+1} are n and m, respectively. α contains the action of the message $[n, p]$ being sent to r from c_{i-1} after σ occurs. Also, the number of such receives from c_{i-1} is greater than the number of sends of μ to c_{i+1} that happen after σ occurs.*

3. *The message is a forward up the circuit.* $\mu = [n, p]$. *r is to be sending μ to c_{i-1}. σ has occurred in α. The link identifiers to c_{i-1} and c_{i+1} are n and m, respectively. α contains the action of the message $[m, p]$ being sent to r from c_{i+1} after σ occurs. Also, the number of such receives from c_{i+1} is greater than the number of sends of μ to c_{i-1} that happen after σ occurs.*

Proof. Suppose r is enabled to send μ at the end of α. This must occur as one of the send operations in Automaton 2, which occur in steps 9, 15, 18, 20, and 24.

If the send is enabled in line 6,9, 15, or 24, then it appears as Step 4, 4, 6, or 2, respectively, of a $\sigma_R(r)$ sequence. Then the message is part of a circuit-building protocol, and the case (1) will apply. For each step $2i$ in $\sigma_R(r)$, $1 \leq i \leq 3$, the argument is the same. The condition guarding the send in Automaton 2 includes the receipt of the message in step $2i - 1$ of $\sigma_R(r)$. It also requires the table to have an entry that is only made when the message in step $2i - 2$ is sent. Thus τ_{2i-1} must be a prefix in α for r to be enabled to send μ. Moreover, once μ is sent, the table row is changed, preventing μ from being sent again. Therefore τ_{2i-1} is the longest prefix of τ_i. Lines 6 and 22 of Automaton 2 ensure that the minimum-identifier condition was satisfied when 4 and 1 of $\sigma_R(r)$ were executed, respectively. Therefore τ_{2i-1} occurred in α. Sending μ executes the next step of $\sigma_R(r)$. It respects the minimal-identifier condition as part of the automaton specification, and because τ_{2i} becomes the new longest prefix, τ_{2i} occurs.

If the send is enabled in line 18 Automaton 2, it is a forward down the circuit, and case (2) will apply. Take m, n, p, c_{i-1}, and c_{i+1} in the statement of the case to be n, m, p, r, and s in Automaton 2, respectively. In order for the send to be enabled in line 18 of Automaton 2, there must exist the entry in the table T that is inserted, in line 10 or line 14, when the message in step 4 or step 6 of $\sigma_R(r)$ is sent, respectively. Thus, by case (1), which we have already shown, a σ_R sequence must have occurred in α. The MESSAGE procedure gets called once for every time a message $[n, p]$ is sent to r from c_{i-1}, and therefore line 18 must have been executed fewer times than the number of receives from c_{i-1} of μ after σ occurs.

The case for a send enabled by line 20 of Automaton 2 is a forward up the circuit, and case (3) applies. The argument is exactly the same as for case (2), with c_{i-1} and c_{i+1} switched. □

We can use these lemmas to partition the actions performed by an agent in an execution of configuration

C. We will use these partitions to construct executions of indistinguishable configurations and prove that they satisfy the requirements of Definition 9.

For a user u we create two partitions. The first is formed by the maximal prefix of $\sigma_U(u, C(u))$ such that each receive in the partition causes the b variable of u's state to be incremented. The condition on the receives is required for a unique maximal prefix to deal with the case that an adversary sends sequence responses multiple times. The second partition is formed from all of u's other actions. By Lemma 1 this is composed of receiving unnecessary messages due to adversarial actions, and we will call this the "junk" partition.

For a router r, we create a partition for each entry in its routing table at any point in the execution and an extra junk partition. For a given routing table entry we create a partition out of the maximum-length subsequence of some $\sigma_R(r)$ sequence, say, σ, such that each receive modifies the same entry in the routing table. We also include every send and receive of a forward performed using that entry. This partition is said to be associated with σ. Every other action done by the router is put in a junk partition, and, by Lemma 2, this partition is composed only of receives.

5.3.2 Indistinguishable Users

Now we prove that an active adversary cannot determine which user creates a given circuit unless the first router on that circuit is controlled by the adversary or the owners of all the other circuits have been determined. That is, an adversary cannot distinguish between a configuration C and the configuration C' that is identical to C except for two circuits with uncompromised first routers that are switched between their owners. In order to do so, we must show that, for any fair, cryptographic execution of C, there exists some action sequence of C' satisfying the indistinguishability requirements of Definition 9. To do so, we simply swap between the switched users the messages that pass between them and the first routers on their circuits and switch the encryption keys of these messages.

Theorem 1. *Let u, v be two distinct users such that neither they nor the first routers in their circuits are compromised (that is, are in A). Let C' be identical to C except the circuits of users u and v are switched. C is indistinguishable from C' to A.*

Proof. Let α be a fair, cryptographic execution of C. To create a possible execution of C', first construct α' by replacing any message sent or received between u (v) and $C_1(u)$ ($C_1(v)$) in α with a message sent or received between v (u) and $C_1(u)$ ($C_1(v)$). Then let ξ be the permutation that sends u to v and v to u and other users to themselves. Create β by applying ξ to the encryption keys of α'.

1. *Every action by an agent in $N \setminus A$ in β is enabled.* All receives are enabled in β, because sends and corresponding receives are modified together.

   For any user $w \notin \{u, v\}$, all messages in $\sigma_U(w, C(w))$ go to or from w, so none are added or removed from α in α'. Also none of the messages in this sequence would be modified by ξ because they are encrypted with a key of w, and ξ doesn't convert messages in α' to be messages of $\sigma_U(w, C(w))$. Therefore if a message is enabled to be sent from w in β it was enabled in α.

   For user u, when u sends a message to $C_1(v)$ in β, it corresponds to v sending a message to $C_1(v)$ in α. v is enabled to do so in α so at that point it has sent and received exactly those messages of $\sigma_U(v, C(v))$ necessary to enable that send. In β we have changed those messages to be messages between u and $C_1(v)$ while modifying the encryption keys, so the necessary $\sigma_U(u, C'(u))$ messages have appeared to enable the send. No additional messages in $\sigma_U(u, C'(u))$ could have appeared in β, because u and v do not communicate on link identifier 0 in α. Therefore u is enabled to send the message. A similar argument works for v.

   For a router $r \notin A \cup \{C_1(u), C_1(v)\}$, the only change in messages to or from r between α and β is from the permutation ξ applied to the encryption keys of the messages. Applying ξ preserves the occurrence of some prefix of $\sigma_R(r, C'(w), w)$ at any point in the execution, for any $w \notin \{u, v\}$. For $w = u$, applying ξ turns the occurrence of some $\sigma_R(r, C(u), u)$ into an occurrence of $\sigma_R(r, C'(u), v)$, and vice versa for $w = v$. It also preserves the appearance of messages to forward and the actual forwarding. Thus any action performed by r in β is enabled because it corresponds to a similar enabled action in α.

   Now consider a message μ sent from $C_1(u)$ in β.

   It may be that μ is part of a $\sigma_R(C_1(u), C(w), w)$ sequence for some $w \notin \{u, v\}$ in α. Then μ is enabled in β since none of the messages in $\sigma_R(C_1(u), C(w), w)$ come from u or v and none involve u or v in the encryption keys, so all exist in β that did in α and no additional ones do. It could also be that, in α, μ is a forward in some circuit not belonging to u or v. Then μ is still enabled in β for a similar reason, recognizing that although it might involve the encryption keys of u or v, the content of messages is ignored in forwards.

   Another case is that, in α, μ is part of some $\sigma_R(C_1(u), C(u), u, i)$. Then in β, μ is part of $\sigma_R(C_1(u), C'(v), v, i)$. This is because every message from u to $C_1(u)$ is changed to a message from v to $C_1(u)$ and every encryption key involving u changes to one involving v. It can be seen by inspecting the messages in a

σ_R sequence that consistently replacing the user in the encryption keys in a σ_R sequence and (when $i = 1$) the previous hop from u to v, as is done to create β, transforms a $\sigma_R(C_1(u), C(u), u, i)$ sequence into a $\sigma_R(C_1(u), C'(v), v, i)$ sequence. No additional messages can enter into this sequence in β because they must be encrypted with a key of v, and any such message will have appeared with a key of u in α and will have filled the same spot in the $\sigma_R(C_1(u), C(u), u, i)$ sequence there. Thus μ is enabled in β. Also, for similar reasons, if μ is a forward in u's circuit in α, then it is a forward for v's circuit in β.

The final case is when, in α, μ is in a $\sigma_R(C_1(u), C(v), v)$ sequence or is a forward on v's circuit. Since v does not communicate directly with $C_1(u)$ as a user in α, it must be that $C_1(u)$ is some intermediate router. Then the only changes to the $\sigma_R(C_1(u), C(v), v)$ messages in α are the encryption keys, which are applied consistently to all the sequence messages and are not interfered with by messages in α already using the target keys since they are also modified. Therefore if μ corresponds to a $\sigma_R(C_1(u), C(v), v)$ send in α, it is a $\sigma_R(C_1(u), C'(u), u)$ message enabled in β. Also, for similar reasons, if μ was a foward in v's circuit in α, it is an enabled forward on u's circuit in β.

The analogous argument works for $C_1(v)$.

2. *β is fair for agents in $N \setminus A$.*

   For any user $w \notin \{u, v\}$, every $\sigma_U(w, C'(w))$ message received in β in its non-junk partition is the same message in α. Therefore every send w is enabled to perform in β it is enabled to perform in α. Because α is fair for w so is β.

   Now consider u. The transformation properly changes the messages from $C_1(v)$ to v in $\sigma_U(v, C(v))$ to messages sent to u that are in the same position in the $\sigma_U(u, C'(u))$ sequence. No extra messages can appear since they must be encrypted using u's keys, and then they would have been encoded with v's keys in α and been part of the $\sigma_U(v, C(v))$ sequence there. Therefore every send that u is enabled to perform in β, v is enabled to perform in α. Since α is fair for v, then β is fair for u. The analogous argument works for v.

   For router $r \notin A \cup \{C_1(u), C_1(v)\}$ to be enabled to perform a send in β but not α, there must be a message in some sequence $\sigma_R(r, C'(w), w, i)$ that r receives in β but doesn't in the corresponding sequence in α. This cannot be for any $w \notin \{u, v\}$, because the messages in this σ_R are not modified, except possibly the content of forwards which doesn't affect their validity. All messages in β that are to r and are in some $\sigma_R(r, C'(u), u)$ are also are sent to r in α and are part of some $\sigma_R(r, C'(v), v)$. Therefore if such a message enables r to send something in β there exists a similar message enabling

r to send something in α. Also forwards along u's circuit in β exist as forwards along v's circuit in α. The analogous argument works for messages of some sequence $\sigma_R(r, C', v)$.

For $C_1(u)$ to be enabled to perform a send in β but not α, there must be a message in some sequence $\sigma_R(C_1(u), C'(w), w)$ or forward that $C_1(u)$ receives in β but doesn't in α. There can not be such a message in the $\sigma_R(C_1(u), C'(w), w)$ sequence for any $w \notin \{u, v\}$, because the messages in this sequence are not modified in the transformation and no new messages encrypted with w's key are created. Also the sender and recipient of forwards from w aren't modified, and the content of forwards which doesn't affect their validity. Now suppose $w = v$. For any message that appears at the end of some $\sigma_R(C_1(u), C'(v), v, i)$ in β that $C_1(u)$ doesn't respond to there must not be an analogous message in $\sigma_R(C_1(u), C(u), u, i)$ in α or $C_1(u)$ would be enabled at that point as well. But this message must be encrypted with v's keys and would be modified by the ξ permutation and thus play the same role for $C_1(u)$ in α. Again, the content of forwards doesn't matter and any forward on v's circuit in β corresponds to a forward on u's circuit in α. The analogous argument works for the case $w = u$. Therefore every send enabled for $C_1(u)$ in β is enabled in α, and β is fair for $C_1(u)$. The analogous argument works for $C_1(v)$.

3. *β is cryptographic.*

   We've already shown that uncompromised routers and users perform enabled actions in β. Since the automatons only allow sending messages encrypted with keys the agent doesn't possess after receiving them, the actions of these agents do not prevent β from being cryptographic. For a compromised user or router, let's say that a control message encrypted with a foreign key is sent before being received at least once. If the encryption key doesn't involve u or v, then the same message gets sent in α before being received, contradicting the fact that α is cryptographic. If the key does involve u, then in α it involves v, in which case if the message is received in α beforehand, it must have received it in β since the key permutation takes v to u. Likewise for messages encrypted with one of v's keys. The fact that α is cryptographic then implies that β is cryptographic.

4. *We can find a $\xi \in \Xi$ and $\pi \in \Pi$ that turn α into a sequence that agrees with β in all the adversary actions.*

   ξ is simply the user permutation used to create β, transposing users u and v, and π is the identity on all messages. Applying these to α yields a sequence that agrees with β everywhere except for messages between u (v) and $C_1(u)$ ($C_1(v)$), which we assumed are not adversarial.

$\square$

5.3.3 Indistinguishable Routers and Destinations

Now we prove that an adversary cannot determine an uncompromised router or destination on a given circuit unless it controls the previous or next router on that circuit. The proof is similar to that of Theorem 1, although it is complicated by the fact that the identities of routers in a circuit are included in multiple ways in the circuit creation protocol. Given an execution of C, we identify those messages that are part of the circuit creation sequence of the modified circuit and then change them to change the router or destination. Then we show that, in the sense of Definition 9, from the adversary's perspective this sequence is indistinguishable from the original and could be an execution of C'.

Theorem 2. *Say there is some user $u \notin A$ such that u's circuit in C contains three consecutive routers, $r_{i-1}, r_i, r_{i+1} \notin A$. Let C' be equal to C, except r_i is replaced with r_i' in u's circuit, where $r_i' \notin A \cup \{r_{i-1}, r_{i+1}\}$. C' is indistinguishable from C to A. The same holds for uncompromised routers (r_i, r_{i+1}) if they begin u's circuit and are replaced with (r_i', r_{i+1}), uncompromised routers (r_{i-1}, r_i), if they end u's circuit and are replaced with (r_{i-1}, r_i'), or uncompromised router and destination (r_{i-1}, r_i) if they are replaced with (r_{i-1}, r_i').*

Proof. Let α be some fair cryptographic execution of C, and let $h(C(u), i)$ denote the number of occurrences of the ith router in the circuit $C(u)$ among the first i routers. We modify α in steps to create an indistinguishable sequence β:

1. If r_i is a router, replace all message components of the form $[\text{EXTEND}, r_i, \{\text{CREATE}\}_{(u,r_i,h(C(u),i))}]$ with $[\text{EXTEND}, r_i', \{\text{CREATE}\}_{(u,r_i',h(C'(u),i))}]$. If r_i is a destination, replace all message components of the form $[\text{DEST}, r_i, b]$ with $[\text{DEST}, r_i', b]$.

2. Consider the partition of router r_{i-1}'s actions that are associated with a $\sigma_R(r_{i-1}, C(u), u, i-1)$ sequence. Replace all messages in this partition that are to and from r_i with the same messages to and from r_i'. Modify the link identifiers on these messages so that they are the smallest identifiers in use between r_{i-1} and r_i' at that point in α. Increase link identifiers that are in use between r_{i-1} and r_i' to make room for these new connections and decrease link identifiers that are in use between r_{i-1} and r_i to fill in the holes created by the removed connections. Perform similar modifications for routers r_i and r_{i+1}.

3. Replace all encryption keys of the form $(u, r_i, h(C(u), i))$ with $(u, r_i', h(C'(u), i))$. Increment as necessary the third component of the encryption keys used between u and r_i' to take into account that r_i'

appears once more in $C'(u)$ than it does in $C(u)$. Also decrement as necessary the third component of the keys used between u and r_i to take into account that r_i appears once less in $C'(u)$ than it does in $C(u)$.

Now we show that the action sequence thus created, β, is a fair cryptographic execution of C':

1. *Every action by an agent in $N \setminus A$ in β is enabled.*

   It is easy to see that all receives are enabled in β since sends and corresponding receives are modified together.

   Our strategy to show that all sends in β are enabled will be to consider the separate non-junk partitions of α after the transformation. First we will show that no sends from uncompromised agents appear in β outside of these transformed partitions. Then we show that any given non-junk partition of α is transformed into a subsequence that is "locally" enabled under C'. A user (router) action sequence is locally enabled if each send satisfies the conditions of Lemma 1 (2) applied just to that sequence. Then we show that it is "globally" enabled in the sense that the sends in the transformed user (router) partition continue to satisfy Lemma 1 (2) when considered over the entire sequence β.

   It is easier to proceed this way since going from a locally to globally-enabled sequence just requires that certain actions *don't* exist in the larger sequence. For users, none of the sends in the transformed non-junk partition can appear again in the larger sequence between being enabled and being sent in the partition. This must also be true for transformed router partitions, and additionally the link identifiers used must be unique and minimal at the point of link creation. Via Lemmas 1 and 2 a locally-enabled action sequence satisfying these global conditions contains only enabled sends in β.

   The fact that there are no sends from uncompromised agents in β outside of the transformed non-junk α partitions helps us prove that actions are globally enabled. By inspecting the three changes made to α, it is clear that no actions are added or deleted from α, and that sends (receives) in α are sends (receives) in β. Since every send in α from an agent $a \in N \setminus A$ is part of one of its non-junk partitions, every send by an uncompromised agent in β is part of one of the transformed partitions.

   We can use this to show that a given sequence is globally enabled. If the sequence is a locally-enabled transformed user partition, it is automatically globally enabled because there are no sends outside the partition to interfere with it. Similarly, for locally-enabled transformed router partitions, we automatically satisfy the send non-inteference property in β as long as we satisfy the second requirement on the link identifiers.

This second requirement for routers is slightly simpler to achieve by noting that all CREATE messages in β were transformed from CREATE messages in α. Therefore we only need to show that the link IDs used in β are unique and minimal among the link creations in α after transformation.

For user $v \neq u$ the non-junk partition has not been modified, and therefore, by Lemma 1, every send is locally enabled in β. Therefore every action by v is enabled.

Now consider the user u's non-junk partition in α. We've modified steps $2i-1$,$2i$,$2i+1$, and $2i+2$ as necessary to change the $\sigma_U(u, C(u))$ prefix to a $\sigma_U(u, C'(u))$ prefix. All these are enabled by Lemma 1, and so this is locally enabled. No sends appear outside of this transformed partition in β. Thus the partition is globally enabled.

Now consider a router $r \notin \{r_{i-1}, r_i, r'_i, r_{i+1}\}$ and a partition of r in α. The partition consists of a prefix of a $\sigma_R(r, C(w))$ sequence, for some user w, and possibly some forwards. The only changes made to the partition are key relabelings and some modification of the messages of forwards. The relabeling turns the $\sigma_R(r, C(w))$ prefix into some $\sigma_R(r, C'(w))$ prefix of the same length, and so sends in this sequence are locally enabled. Forwards are enabled regardless of content, and so they are also locally enabled. No link identifiers of r have changed, and so they are still unique and minimal. Therefore the whole partition is globally enabled.

Now consider r_{i-1}. Take some non-junk partition of α that does not contain a sequence in which r_{i-1} is the $(i-1)$th circuit router of u, that is, that does contain a $\sigma_R(r_{i-1}, C(w), w, j)$ sequence, $w \neq u$ or $j \neq i-1$. For the same reasons as the preceding case, it is transformed into a sequence associated with a $\sigma_R(r_{i-1}, C'(w), w)$ sequence. Thus it is locally enabled. The partition that is a prefix of $\sigma_R(r_{i-1}, C(u), u, i-1)$ can be seen by inspection to be modified to be a locally-enabled sequence associated with $\sigma_R(r_{i-1}, C'(u), u, i-1)$. The link identifiers used in every transformed partition of r_{i-1} are unique and minimal in β because the original partitions had unique and minimal IDs in α, we haven't changed the IDs or neighbors of any partitions not connecting with r'_i or r_i, and we have changed the ID in partitions connecting with r'_i or r_i to make IDs unique and minimal after changing a partition to connect with r'_i instead of r_i. Thus the whole sequence is globally enabled. The analogous arguments work for r_{i+1}, r_i and r'_i.

2. *β is fair for agents in $N \setminus A$.*

   To show this we will again consider the transformed partitions of α. We have shown that they form enabled sequences, and now need to show that no messages from the transformed junk partition belong in these sequences. For users, this means that the next step in a transformed σ_U partition isn't received. For routers, it means that the next step in a transformed σ_R partition isn't received, no new forwards on a created circuit are received, and that no new valid CREATE messages are received.

Consider a user $w \neq u$. Every receive action by w in β is from a receive action by w in α. The messages received by w are never modified to use one of w's keys, so a message encrypted with w's keys in β uses the same keys in α. Also the content of an encrypted message is never changed to be a message that appears in $\sigma_U(w, C'(w))$. Therefore any receive that is a step of $\sigma_U(w, C'(w))$ in β is the same step in $\sigma_U(w, C(w))$ in α. Therefore β is fair for w.

Now consider user u. As shown, the transformed non-junk partition in α is a locally-enabled sequence in β. For u to have an unperformed enabled action in β, the next message in the $\sigma_U(u, C'(u))$ sequence must come from the junk sequence and be unanswered. $\sigma_U(u, C(u))$ and $\sigma_U(u, C'(u))$ differ in steps $(2 + 2j)$, $i \leq j \leq l$. These only differ in the encryption keys, and these have been renumbered in such a way that the $(2 + 2j)$th step in $\sigma_U(u, C'(u))$ appears if and only if the $(2 + 2j)$th step appears in $\sigma_U(u, C(u))$. Thus an enabling receive in β implies that such a receive exists in α. Therefore β is fair for u.

Now consider a router $r \notin \{r_{i-1}, r_i, r'_i, r_{i+1}\}$. For a given transformed partition, no new messages of the associated $\sigma_R(r)$ sequence can appear in β. This is because $\sigma_R(r)$ messages are all encrypted for r and we have created no such messages, nor have we modified such messages so as to create a new message in the $\sigma_R(r)$ sequence. For forwards, first we recognize that the transformation maintains source and link ID consistency for r in the sense that if we were to group r's receives in α by their source and link ID the transformed groups would be the same as the same groups created in β. Therefore for an incoming message to be transformed into a forward on a created circuit, it must previously be sent with the link identifiers of the incoming link, but since content in forwards doesn't matter, this would be a forward in α as well. Finally there are no valid CREATE messages received in β that aren't received in α. No new messages are sent to r, no messages are transformed into a CREATE, no keys have been modified to belong to r, and r's link IDs have been consistently changed. Therefore β is fair for r.

Now consider r_{i-1}. For a transformed partition of r_{i-1}, suppose that some receive extends the associated $\sigma_R(r_{i-1})$ or acts as a forward on the created circuit. This receive must be from the junk partition of r_{i-1} since we have shown its that non-junk partitions form enabled sequences. This message can't be from a router not in $\{r_i, r'_i\}$ because all such messages existed in α with the same link ID, source, and destination, and the content is either the same or is a forward, which would still be a forward in α. It can't come from r_i. This router is uncompromised and therefore properly uses link identifiers. If the message were part of the associated $\sigma_R(r_{i-1})$ sequence in β, it would exist in α with the ID in use by r_{i-1} and r_i in the sequence in α and with the same content, so this can't be the case. If the message were a forward, again it would exist in α with the link ID in use between r_{i-1} and r_i in the partition, and would therefore be a forward in α as well. Similar arguments work for messages from r'_i. Finally, no new partitions can be created, because CREATE messages to r_{i-1} on unique link IDs in β are the same in α. Therefore β is fair for r_{i-1}. The analogous arguments work for r_{i+1}.

Now consider r_i. Because only link identifiers to r_{i-1} and r_{i+1} have been changed, and those routers are uncompromised, all messages in β to r_i from a given router and with a given link ID are transformed from all messages in α from the same router and of a given (possibly different) link ID. Suppose a message receive in β enables a send in the associated $\sigma_R(r_i)$ that is not enabled in the related sequence in α, or acts as a new forward. It must have been sent in α on the link ID in α of that partition. Because messages aren't redirected to r_i and senders aren't changed, it must have been sent in α by the same sender. Because content doesn't change in the $\sigma_R(r_i)$ messages and doesn't matter in forwards this message would perform the same function in α, contradicting the fairness of α. No new partitions can be created because new CREATE messages aren't made by the transformation, senders are the same, and link identifiers are renumbered in such a way that distinct link IDs from a router in α are distinct in β. Therefore β is fair for r_i.

A similar argument works for r'_i over its partitions in α, but we do reassign a partition of r_i to r'_i, which we must also consider. Notice that the messages redirected to r'_i exist on unique link IDs in β with r_{i-1} and r_{i+1} in β. Therefore these cannot enable actions on the other transformed partitions, and vice versa. Also no junk messages of r'_i can enable actions in this transformed partition because the connecting routers, r_{i-1} and r_{i+1}, are uncompromised and will have sent these messages on a link ID that is different from the ID of the transformed new partition in β. Finally, we show that the only valid CREATE messages received by r'_i in β are those in transformed partitions of α. Every CREATE in β is a CREATE in α. Every valid CREATE to r'_i in α becomes a valid CREATE in β because it is part of a transformed partition and we have shown that these become enabled sequences. The only messages redirected to r'_i belong to r_i's reassigned partition, which forms a fair sequence in α and maintains this after transformation. The final possibility for a new CREATE is a CREATE message from r'_i's junk partition that was sent to r'_i in α but was encrypted with a key of r_i, which then gets changed in the transformation. The only such message is $\{\text{CREATE}\}_{(u,r_i,h(C(u),i))}$. Only r_{i-1} could send this message in α because α is cryptographic, and u only produces such a message encrypted with the key of r'_{i-1}. It would only send this message if it were to receive the message $\{\text{EXTEND}, r'_i, \{\text{CREATE}\}_{(u,r_i,h(C(u),i))}\}_{(u,r_{i-1},h(C(u),i-1))}$, which u never sends in α. Again by the cryptographic property of α r_{i-1} never sends this CREATE to r'_i in α. Thus no valid CREATE messages are received by r'_i in β that are not transformed from partitions in α, and we have shown that these transformed partitions are fair. Therefore β is fair for r'_i.

3. *β is cryptographic.*

   For uncompromised routers, the fact that all sends are enabled in β guarantees cryptographic sends, because the protocol ensures this property. Compromised routers send and receive all the same messages, but to which the transformation function has been applied. Therefore because α is cryptographic, β is.

4. *We can find key and message permutations that turn β into a sequence that agrees with α in all adversary actions.*

   No messages are redirected towards or away from $a \in A$ when constructing β. We apply the message permutation to β of transposing $\{[\text{EXTEND}, r'_i, \{\text{CREATE}\}_{(u,r'_i,h(C'(u),i))}]\}_{k_1,\ldots,k_j}$ and $\{[\text{EXTEND}, r_i, \{\text{CREATE}\}_{(u,r_i,h}$ $1 \leq j \leq l$, where k_j isn't shared by the adversary. (If we replaced a destination we instead transpose $\{[DEST, r'_i, b]\}_{k_1\ldots,k_j}$ and $\{[DEST, r_i, b]\}_{k_1\ldots,k_j}$.) We also apply the key permutation that sends $(u, r'_i, h(C'(u), i))$ to $(u, r_i, h(C(u), i))$ and undoes the renumbering of the r'_i and r_i keys. Then the subsequence of actions by a in β is identical to the subsequence in α.

□

5.3.4 Indistinguishable Identifiers

Theorem 3. *Say there is some uncompromised user u such that all routers and the destination in $C(u)$ are uncompromised. Then let C' be a configuration that is identical to C, except that u uses a different circuit identifier. C' is indistinguishable from C to A.*

Proof. Let α be a fair, cryptographic execution of C. To create β, simply change every occurrence of u's circuit identifier in C ($C_{l+2}(u)$) to its identifier in C'. β is enabled, fair, and cryptographic for C' because no message containing $C_{l+2}(u)$ gets sent to the adversary in α and the protocol itself ignores circuit identifiers except to forward them on. It appears the same to A for the same reason. □

5.3.5 Distinguishable Configurations

Let D'_A be the transitive closure of pairs that are indistinguishable by Theorems 1, 2, and 3. Theorems 1, 2, and 3 will allow us to characterize exactly which configurations are indistinguishable.

Lemma 3. *D'_A is an equivalence relation on indistinguishable configurations.*

Proof. If $C_1 \sim_{D_A} C_2$, for some configurations C_1 and C_2, then, for every execution α of configuration C_1, there exists an indistinguishable execution β of C_2. Then if $C_2 \sim_{D_A} C_3$, for some configuration C_3, there must exist an execution γ of C_3 that is indistinguishable from β. α and γ are then indistinguishable, and thus indistinguishability of configurations is transitive. Therefore taking a transitive closure of indistinguishable pairs, as we do to create D'_A, preserves the indistinguishability of all pairs.

Theorems 1, 2, and 3 are all symmetric, in that if, for configurations C_1 and C_2, $C_1 \sim C_2$ by one of the theorems, then $C_2 \sim C_1$ by the same theorem. Therefore D'_A is symmetric. D'_A is transitive by construction. D'_A is reflexive because a configuration is trivially indistinguishable from itself. Therefore D'_A is an equivalence relation. □

We introduce some notation to conveniently refer to the configurations related by D'_A.

Definition 13. *For configurations C and D, we say that $C \approx_{D_A} D$ if C and D are related by a chain of configurations that are indistinguishable by Theorems 1, 2, and 3.*

We can easily tell which configurations are in the same equivalence class using the following function. It reduces a circuit to an identifier, the compromised positions, and the positions adjacent to compromised positions.

Definition 14. *Let $\rho : U \times N^l \times D \times \mathbb{N}_+ \times \mathcal{P}(N) \to \mathbb{N} \times \mathcal{P}(N \times \mathbb{N}_+)$ be:*

$$\rho(u, c, d, j, A) = \begin{cases} (j, \{(r, i) \in N \times \mathbb{N}_+ | c_i = r \wedge (c_{i-1} \in A \vee c_i \in A \vee c_{i+1} \in A)\}) & \text{if } c_i \in A \text{ for some } i \\ (0, \emptyset) & \text{otherwise} \end{cases}$$

In the preceding let c_0 refer to u and c_{l+1} refer to d.

We overload this notation and use $\rho(C)$ to refer to the multiset formed from the circuits of configuration C adjoined with their user and reduced by ρ. That is, $\rho(C) = \{\rho(u, C(u), A) | u \in U\}$. The following lemma shows that ρ captures the indistinguishable features of a configuration according to Theorems 1, 2, and 3.

Lemma 4. *Let C and D be configurations. $C \approx_{D_A} D$ if and only if $\rho(C) = \rho(D)$.*

Proof. First we show that if $\rho(C) = \rho(D)$, then $C \approx_{D_A} D$. Because $\rho(C) = \rho(D)$, there is a permutation π of the users such that $\rho(C(u)) = \rho(D(\pi(u)))$. Let C' be the configuration such that, for every user u, we replace the routers in $C(u)$ that are not compromised or adjacent to compromised routers by the routers in the same circuit position in $D(\pi(u))$, and such that we do the same for an uncompromised destination. C' is indistinguishable from C by Theorem 2. Let C'' be the configurations that results when we permute the users of C' by π. π must act as the identity on all routers that are compromised or have compromised first routers, because otherwise $(u, 0)$ would exist in $\rho(C(u))$ and would not exist in $\rho(D(\pi(u)))$, contradicting their equality. Therefore, C'' is indistinguishable from C' by Theorem 1. Finally, let C''' be the result of changing, for all users such that $\rho(u) = (0, \emptyset)$, the circuit identifiers $C''_{l+2}(u)$ to $D_{l+2}(u)$. C''' is indistinguishable from C'' by Theorem 3, because $\rho(C(u)) = (0, \emptyset)$ implies that u and $C_i(u)$ are uncompromised, $1 \leq i \leq l$.

For all users u, if $C'''_i(u)$, $C'''_{i-1}(u)$, and $C'''_{i+1}(u)$ are uncompromised, $1 \leq i \leq l+1$, then

$$\begin{aligned} C'''_i(u) &= C'_i(\pi^{-1}(u)) && (5.1) \\ &= D_i(\pi(\pi^{-1}(u))) \text{ by construction.} && (5.2) \end{aligned}$$

If $C'''_i(u)$, $C'''_{i-1}(u)$, or $C'''_{i+1}(u)$ is compromised, $1 \leq i \leq l+1$, then

$$\begin{aligned} C'''_i(u) &= C'_i(\pi^{-1}(u)) && (5.3) \\ &= C_i(\pi^{-1}(u)) && (5.4) \\ &= D_i(\pi(\pi^{-1}(u))) \text{ because } \rho(C(u)) = \rho(D(\pi(u))). && (5.5) \end{aligned}$$

If $C'''(u) = (0, \emptyset)$, then the circuit identifier $C'''_{l+2}(u) = D_{l+2}(u)$ by construction. If $C'''(u) \neq (0, \emptyset)$, then $C'''_{l+2}(u) = D_{l+2}(u)$, because

$$\begin{aligned} \rho(C'''(u)) &= \rho(C(\pi^{-1}(u))) && (5.6) \\ &= \rho(D(\pi(\pi^{-1}(u)))), && (5.7) \end{aligned}$$

and the circuit identifier is included in $\rho(c)$ when c contains compromised agents. Therefore $C''' = D$, and thus $C \approx_{D_A} = D$.

Now we show that if $C \approx_{D_a} D$, then $\rho(C) = \rho(D)$. Because $C \approx_{D_A} D$, there must be a chain of indistuinguishable configurations relating C to D such that each adjacent pair in the chain is indistinguishable by Theorems 1, 2, and 3. For any pair of configurations C_1 and C_2 that are indistinguishable by one of these theorems, $\rho(C_1) = \rho(C_2)$:

1. If C_1 and C_2 are related by Theorem 1, a pair of uncompromised users with uncompromised first routers is swapped. ρ is invariant under this operation.

2. If C_1 and C_2 are related by Theorem 2, then a router or destination r_i such that r_i, r_{i-1}, and r_{i+1} are uncompromised is replaced by r'_i. ρ is invariant under this operation.

3. If C_1 and C_2 are related by Theorem 3, the circuit identifier of an uncompromised circuit is modified. ρ is invariant under this operation.

Therefore $\rho(C) = \rho(D)$. □

Now we show that the equivalence relation is in fact the entire indistinguishability relation and that Theorems 1, 2, and 3 characterize which configurations are indistinguishable. The reason for this is that an adversary can easily determine which entries in the compromised routers belong to the same circuits and what positions they hold in those circuits. The adversary links together entries in its routers by using the circuit identifiers that are uniquely associated with each circuit. And because circuits have a fixed length compromised routers can determine their position in the circuit by counting the number of messages received after the circuit entry is made.

Theorem 4. *Configurations C and D are indistinguishable only if $C \approx_{D_A} D$.*

Proof. Suppose that C and D are not in the same equivalence class. Let the adversary run the automata prescribed by the protocol on the agents it controls. Let α be a fair, cryptographic execution of C and β be a fair, cryptographic execution of D.

Partition the adversary actions of α into subsequences that share the same circuit identifier. There is at most one such partition for each circuit. Circuit positions that are created in the same partition belong to the same circuit. In each partition the adversary can determine the absolute location of a circuit position filled by a given compromised agent a by counting the total number of messages it sees after the initial CREATE. A can also determine the agents that precede and succeed a on the circuit and the circuit identifier itself. Therefore A can determine the reduced circuit structure $\rho(C)$ from α.

The adversary can use β in the same way to determine $\rho(D)$. By Lemma 4, $\rho(C) \neq \rho(D)$, and so A can always distinguish between C and D. □

5.3.6 Anonymity

The configurations that provide sender anonymity, receiver anonymity, and relationship anonymity follow easily from Theorems 1, 2, 3, and 4.

Corollary 1. *User u has sender anonymity in configuration C with respect to adversary A if and only if at least one of the following cases is true:*

1. *u and $C_1(u)$ are uncompromised, and there exists another user $v \neq u$ such that v and $C_1(v)$ are uncompromised.*
2. *u and $C_i(u)$ are uncompromised, for all i.*

Proof. Suppose u has sender anonymity in configuration C. Then there must be an indistinguishable configuration D in which u uses a different circuit identifier. By Theorem 4, $C \approx_{D_A} D$, and therefore, by definition, there must be a chain of configurations from C to D such that consecutive pairs are indistinguishable by Theorems 1, 2, and 3. At some point in this chain, the circuit identifier of u must change. If, at this point, the consecutive configurations are indistinguishable by Theorem 1, u must swap circuits with some v, and the conditions for this are exactly those of (1). It is not possible for the consecutive configurations at this point to be indistinguishable by Theorem 2, because it cannot affect the circuit identifier of u. If, at this point, the configurations are indistinguishable by Theorem 3, the conditions of (2) must hold.

Conversely, if (1) holds we can apply Theorem 1 to C to yield an indistinguishable configuration in which u has a different circuit identifier. If (2) holds we can apply Theorem 3 to C to change the circuit identifier of u entirely. □

Corollary 2. *Destination d has receiver anonymity on u's circuit, in configuration C, and with respect to adversary A if and only if at least one of the following cases is true:*

1. *u and $C_l(u)$ are uncompromised, and there exists more than one destination.*
2. *u and $C_i(u)$ are uncompromised, for all i.*

Proof. We use the assumption that all destinations are uncompromised. Suppose d has receiver anonymity in configuration C. Then there must be an indistinguishable configuration D in which d is not the destination on a circuit with the circuit identifier of u in C. By Theorem 4, $C \approx_{D_A} D$, and therefore, by definition, there must be a chain of configurations from C to D such that consecutive pairs are indistinguishable by Theorems 1, 2, and 3. At some point in this chain, d must no longer be the destination on a circuit with the circuit identifier of u in C. Theorem 1 only affects users, and thus cannot apply at this point. If Theorem 2 applies, the conditions of (1) must be satisfied. If Theorem 3 applies, the conditions of (2) must be satisfied.

Conversely, if (1) holds we can apply Theorem 2 to C to yield an indistinguishable configuration in which d is not the destination of a circuit with the circuit identifier of u in C. If (2) holds, we can apply Theorem 3 to C to change the circuit identifier of u entirely. □

Corollary 3. *User u and destination d have relationship anonymity in configuration C with respect to adversary A if and only if at least one of the following cases apply:*

1. *u and $C_l(u)$ are uncompromised, and there exists more than one destination.*
2. *u and $C_1(u)$ are uncompromised. There exists another user $v \neq u$ such that v and $C_1(v)$ are uncompromised. $C_{l+1}(v) \neq d$, or $C_l(v)$ is uncompromised and there exists more than one destination.*

Proof. Suppose u and d have relationship anonymity in configuration C. By Theorem 4, $C \approx_{D_A} D$. Therefore, by definition, there must be a chain of configurations $C = C^0, C^1, \ldots, C^m = D$ such that each consecutive pair, (C^i, C^{i+1}), is indistinguishable by Theorems 1, 2, and 3. At some point i in the chain, d must cease to be the destination of u. Suppose that, at this point, the configurations are indistinguishable by Theorem 1. Then u and $C_1^i(u)$ are uncompromised, and, for some $v \neq u$, v and $C_1^i(u)$ are uncompromised. Also, the destination of v is not d. Then either $C_{l+1}(v) \neq d$, or v and d have relationship anonymity in C. We inductively assume (performing induction on the length of the chain) that Corollary 3 holds. Then if (1) holds for v, (2) is satisfied for u. If (2) holds for v, then, for some w, v and w can be swapped to give v relationship anonymity. w can therefore play the role of v in the conditions for (2). In the base case, swapping u and v happens at $i = 0$, and it must be the case that $C_{l+1}(v) = d$, satisfying (1).

Conversely, if (1) holds, we can apply Theorem 2 to replace d with another destination, yielding an indistinguishable configuration in which the destination of u is not d. Now suppose (2) holds. If $C_{l+1}(v) \neq d$, then we can apply Theorem 1 to swap u and v in C, yielding an indistinguishable configuration in which the destination of u is not d. Otherwise, we can swap u and v by Theorem 1, and then we are able to apply Theorem 2 to replace d as the destination of u. □

5.3.7 Model Changes

We chose the described protocol to balance two goals. The first was to accurately model the Tor protocol . The second was to make it simple enough to be analyzed, and also so that the main ideas of the analysis weren't unnecessarily complicated. Our results are robust to changes of the protocol, however. We can make the protocol simpler by removing circuit identifiers and multiple encryption without weakening the indistinguishability results. In the other direction, we can make it more complicated with a stream cipher and multiple circuits per user without weakening the distinguishability results.

Circuit identifiers can be simulated by an adversary performing a timing attack, and therefore are just a convenience that helps us reason about the anonymity of the protocol. Consider modifying the protocol so that the messages sent by users do not include circuit identifiers. That is, remove the leading number from all messages received or sent in Automaton 1. Call this protocol "identifier-free onion routing." The following theorem shows that an adversary can simulate the use of circuit identifiers.

Theorem 5. *There exists an adversary that can calculate, for an execution of identifier-free onion routing, a sequence of messages he could have received had the users used circuit identifiers.*

Proof. The adversary must be able to link together the different parts of the circuit-building process that he can observe to add circuit identifiers consistently to the messages he receives. To do so, the adversary gives a unique number n_i to each router a_i that he controls, where $n_i < 2l$ and $|n_i - n_j| > 2l$ for all i, j. Then, he runs the protocol normally, except that, after sending a CREATED message from his router a_i, he sends n_i dummy messages up the same link. These messages get forwarded up the circuit. Then, if the adversary observes between n_i and $2l + n_i$ messages coming up a circuit at some other compromised router a_j, he knows that a_i and a_i are on the same circuit.

Then the adversary can pick any set of n numbers as circuit identifiers, assign those to circuits, and for each circuit add its identifier to the beginning of those messages that are sent on that circuit. □

Theorem 5 shows that circuit identifiers do not affect anonymity, because sender anonymity, receiver anonymity, and relationship anonymity are invariant under the actual numbers used as identifiers. The protocol automata only test identifiers for equality, and therefore permuting circuit identifiers doesn't affect which executions are indistinguishable. Thus if the adversary can determine a set of circuit identifiers which could have been used in the protocol, the sender, receiver, and relationship anonymity is the same as if those identifiers or any permutation of them had been used.

Multiple encryption does not appear to be necessary for the distinguishability theorems, and therefore the anonymity results. Consider a single-encryption protocol in which the user only encrypts each message with the key of the last router added to the circuit. Messages aren't encrypted or decrypted as they pass up and down a circuit. The adversary still is not able to determine parts of a circuit that aren't adjacent to a compromised agent. The proof of this under multiple encryption did not use the changing representation of messages going along a circuit, and only relied on the last key of the multiple encryption to hide the content of messages. Single encryption does allow the adversary to easily link entries in his routers by sending messages along the circuit. This power is already available in our model from circuit identifiers, though.

Stream ciphers are used in the Tor protocol and prevent signaling along a circuit using dummy messages. Sending such messages will throw off the counter by some routers on the circuit and the circuit will stop working. We can model a stream cipher by expressing the encryption of the ith message p with key k as $\{p\}_{(k,i)}$, and allowing a different permutation to be applied for every pair (k, i). This can only increase the size of the configuration indistinguishability relation. However, the proof for the distinguishability of configurations only relies on the ability of the adversary to decrypt using his keys, count messages, and recognize the circuit identifier. Therefore it should still hold when the model uses a stream cipher. Also, with a stream cipher the circuit identifier is still not necessary for our results. The adversary can again use the process described above to link entries in compromised routers, since although it involves sending dummy messages, they are sent after the circuit creation is finished and therefore do not interfere with it.

Allowing users to create multiple circuits doesn't weaken the adversary's power to link together its circuit positions and determine their position, but the number of configurations that are consistent with this view does in some cases increase. Let users create an arbitrary number of circuits, where a different key is used between a user and router for every position-and-circuit pair. Then the protocol executes as if each circuit is created by a different user. Therefore the adversary should be able to determine the reduced circuit structure $\rho(C)$ of a configuration C. Moreover, the adversary should not be able to determine any additional router information, which does not depend on the identity of the user. However, the function mapping circuits to users is no longer a permutation, and therefore there should only be more possibilities for the user of a given circuit. This would only improve anonymity.

Chapter 6

Probabilistic Anonymity in Onion Routing

6.1 Introduction

In Chapters 4 and 5, we present a formal I/O automata model of onion routing and proved anonymity guarantees. In this chapter, we construct an abstraction of that model and treat the network simply as a black box to which users connect and through which they communicate with destinations. The abstraction captures the relevant properties of a protocol execution that the adversary can infer from his observations - namely, the observed users, the observed destinations, and the possible connections between the two. We give a map between the models and show that they posses the same anonymity properties. In this way, we show that one can abstract away from much of the design specific to onion routing so that our results may apply both to onion routing and to other low-latency anonymous-communication designs. The executions described in Chapter2 disappear from the analysis done herein.

Our previous analysis in the I/O-automata model was possibilistic, a notion of anonymity that is simply not sensitive enough. It makes no distinction between communication that is equally likely to be from any one of a hundred senders and communication that came from one sender with probability .99 and from each of the other 99 senders with probability .000101. An adversary in the real world is likely to have information about which scenarios are more realistic than others. In particular, users' communication patterns are not totally random. When the adversary can determine with high probability, *e.g.*, the sender of a message, that sender is not anonymous in a meaningful way.

Using this intuition, we add a probability distribution to the I/O-automata model of onion routing given in Chapter4. For any set of actual circuit sources and destinations, there is a larger set that is consistent with the observations made by an adversary. The adversary can then infer conditional probabilities on this larger set using the distribution. This gives the adversary probabilistic information about the facts we want the network to hide, such as the initiator of a communication.

The probability distribution that we use models heterogeneous user behavior. In our onion-routing protocol, each user chooses a circuit to a destination. We make this choice probabilistic and have each user choose a destination according to some probability distribution, allowing this distribution to be different for different users. We assume that the users choose their circuits by selecting the routers on it independently and at random.

After observing the protocol, the adversary can in principle infer some distribution on circuit source and destination. He may not actually know the underlying probability distribution, however. In particular, it doesn't seem likely that the adversary would know how every user selects destinations. In our analysis, we take a worst-case view and assume that the adversary knows the distribution exactly.

This assumption may not be too pessimistic, because over time the adversary might learn a good approximation of user behavior via the long-term intersection attack [14]. In an intersection attack, one watches repeated communication events for patterns of senders and receivers over time. Unless all senders are on and sending all the time (in a way not selectively blockable by an adversary) and/or all receivers receiving all the time, if different senders have different receiving partners, there will be patterns that arise and eventually differentiate the communication partners. It has long been recognized that no system design is secure against a longterm intersection attack.

If the adversary does know the distribution over destinations for every user, may seem as though anonymity has been essentially lost anyway. However, even when the adversary knows how a user generally behaves, the anonymity network may make it hard for him to determine who is responsible for any specific action, and the anonymity of a specific action is what we are interested in.

We analyze relationship anonymity in our probabilistic onion-routing model. Our definition of relationship anonymity in Chapter 5 simply requires that, as far as the adversary can tell, there is more than one possibility for a given user's destination. The probability distribution on executions we add in this chapter yields a conditional distribution on these possible destinations. Therefore we can use a quantitative metric for anonymity, as discussed in Chapter 2. We will use the probability assigned to the correct destination as our metric. In part, this is because it is the simplest metric. Also, any statements about entropy and maximum probability metrics only make loose guarantees about the probability assigned to the actual subject, a quantity that clearly seems important to the individual users.

We look at the value of this anonymity metric for a choice of destination by a user. Fixing a destination by just one user, say u, does not determine what the adversary sees, however. The adversary's observations are also affected by the destinations chosen by the other users and the circuits chosen by everybody. Because those variables are chosen probabilistically under the distribution we added, the anonymity metric will have its own distribution. Several statistics about this distribution might be interesting; in this paper, we look at its expectation.

The distribution of the anonymity metric for a given user and destination depends on the other users' destination distributions. If their distributions are very different, the adversary may have an easy time separating out the actions of the user. If they are similar, the user may more effectively hide in the crowd.

We begin by providing a kind of worst-case guarantee to a user with a given destination distribution by finding the maximum expectation over the possible destination distributions of the other users. Our results show that the worst case is when every other user either always visits the destinations the user is otherwise least likely to visit or always visits his actual destination. Which one is worse depends on how likely he was to visit his destination in the first place. If he is unlikely to visit it, it is worse when everybody else always visits his otherwise least-likely destination, because the adversary can generally infer that he is not responsible for communication to that destination. When he is likely to visit it, the adversary considers him likely to be the culprit whenever the destination is observed, and so observing that destination often causes the adversary to suspect the truth. We give an approximation to the user's anonymity in these worst cases for large user populations that shows that on average it decreases by about the fraction of the network the adversary controls.

We then consider anonymity in a more typical set of user distributions. In the model suggested by Shmatikov and Wang [62], each user selects a destination from a common Zipfian distribution. Because the users are identical, every user hides well among the others. As the user population grows, the anonymity loss in this case tends to the square of the fraction of the network that is compromised.

We expect this to have potential practical applications. For example, designs for shared security-alert repositories to facilitate both forensic analysis for improved security design and quicker responses to widescale attacks have been proposed [40]. A participant in a shared security-alert repository might expect to be known to communicate with it on a regular basis. Assuming reports of intrusions, etc. are adequately sanitized, the concern of the participant should be to hide when it is that updates from that participant arrive at the repository, *i.e.*, which updates are likely to be from that participant as opposed to others.

6.2 Technical Preliminaries

6.2.1 Model

We describe our analysis of onion routing in terms of a black-box model of anonymous communication. We are using a black-box model for two reasons: First, it abstracts away the nonessential details, and second, its generality immediately suggests ways to perform similar analyses of other anonymity networks. It models a round of anonymous communication as a set of inputs owned by users and a set of outputs owned by destinations. The adversary observes the source of some of the inputs and the destination of some of the outputs. This captures the basic capabilities of an adversary in an onion-routing network that controls some of the routers. In this situation, the adversary can determine the source of messages when he controls the first router on the source's circuit and the destination of messages when he controls the last router. In order for the adversary always to be able to recognize when it controls the onion router adjacent to the circuit source, we assume that the initiating client is not located at an onion-routing network node. This is the case for the vast majority of circuits in Tor and in all significant deployments of onion routing and similar systems to date. The black box system can also model a mix network under attack by a global, passive adversary. Such a model was used by Kesdogan et al. [36] in their analysis of an intersection attack.

We add two assumptions to specialize this model to onion routing. First, we assume that every user owns exactly one input and is responsible for exactly one output in a round. Certainly users can communicate with multiple destinations simultaneously in actual onion-routing systems. However, it seems likely that in practice most users have at most some small constant number of active connections at any time, and the smaller this constant is the fewer possibilities there are that are consistent with the adversary's observations. Therefore, this assumption is a conservative one that gives the adversary as much power to break anonymity as the limited number of user circuits can provide. Second, we assume the adversary can link together an input and output from the same user when he observes them both. This is another conservative assumption that is motivated by the existence of timing attacks that an active adversary can use to link traffic that it sees at various points along its path through the network.

Let U be the set of users with $|U| = n$. Let Δ be the set of destinations. A round of communication is defined by the *black-box configuration* that includes the destination chose by the users and the observation made by the adversary.

Definition 15. *A* black-box configuration C *in a black-box system consists of:*

1. $C_D : U \to \Delta$*, which indicates the selection of a destination by each user.*

2. $C_I : U \to \{0, 1\}$*, which indicates the set of users whose inputs are observed.*

3. $C_O : U \to \{0, 1\}$*, which indicates the set of users whose outputs are observed.*

For the remainder of the chapter, the term *configuration* will refer to a black-box configuration. A user's input, output, and destination will be called its *circuit*.

We include the probabilistic behavior of users by adding a probability distribution over configurations. Let each user u select a destination d from a distribution p^u over Δ, where we denote the probability that u chooses d as p^u_d. Every input and output is independently observed with probability b. This reflects the probability that the first or last router of a user's circuit is compromised when the user selects the circuit's routers independently and at random, and the adversary controls a fraction b of the routers. The probability of a configuration C is the joint probability of its events:

$$Pr[C] = \prod_{u \in U} \left(p^u_{C_D(u)}\right) \left(b^{C_I(u)}(1-b)^{1-C_I(u)}\right) \left(b^{C_O(u)}(1-b)^{1-C_O(u)}\right). \tag{6.1}$$

For any configuration, there is a larger set of configurations that are consistent with the inputs and outputs that the adversary sees. We will call two configurations *indistinguishable* if the sets of inputs, outputs, and links between them that the adversary observes are the same.

Definition 16. *Configurations* C *and* $\overline{C}$ *are* indistinguishable *if there exists a permutation* $\pi : U \to U$ *such that for all* $u \in U$*:*

1. $C_I(u) = 1 \wedge C_O(u) = 1 \Rightarrow \overline{C}_I(u) = 1 \wedge \overline{C}_O(u) = 1 \wedge C_D(u) = \overline{C}_D(u)$

2. $C_I(u) = 1 \wedge C_O(u) = 0 \Rightarrow \overline{C}_I(u) = 1 \wedge \overline{C}_O(u) = 0$

3. $C_I(u) = 0 \wedge C_O(u) = 1 \Rightarrow \overline{C}_I(\pi(u)) = 0 \wedge \overline{C}_O(\pi(u)) = 1 \wedge C_D(u) = \overline{C}_D(\pi(u))$

4. $C_I(u) = 0 \wedge C_O(u) = 0 \Rightarrow \overline{C}_I(\pi(u)) = 0 \wedge \overline{C}_O(\pi(u)) = 0$

Thus, two configurations are indistinguishable if they have the same pattern of observed inputs, outputs, and destinations, while allowing the identities of users with unobserved inputs to be permuted. The adversary relation is an equivalence relation, and, in particular, is symmetric, because, if C and $\overline{C}$ are indistinguishable under π, then $\overline{C}$ and C are indistinguishable under π^{-1}. Therefore we use the notation $C \approx \overline{C}$ to indicate that configurations C and $\overline{C}$ are indistinguishable.

6.3 Relationship Anonymity

We analyze the relationship anonymity of users and destinations in our model. As described in Chapter 2, relationship anonymity refers to the adversary's ability to determine if a user and destination communicate with each other.

A user has relationship anonymity in a possibilistic sense if there is an indistinguishable configuration in which the user does not communicate with his destination. For example, under this definition a user with observed output but unobserved input sends that output anonymously if there exists another user with unobserved input.

The probability distribution we have added to configurations allows us to use a probabilistic notion of anonymity and incorporate the degree of certainty that the adversary has about communication between users and destinations. After making observations in the actual configuration, the adversary can infer a conditional probability distribution on configurations. We measure the relationship anonymity of user u and destination d by the posterior probability that u chooses d as his destination. The lower this is, the more anonymous we consider their relationship.

The relationship anonymity of u and d varies with the destination choices of the other users and the observations of the adversary. If, for example, u's output is observed, and the inputs of all other users are observed, then the adversary assign u's destination a probability of 1. Because we want to examine the relationship anonymity of u conditioned only on his destination, we end up with a distribution on the anonymity metric. We look at the expectation of this distribution.

This expectation distribution depends on the destination distributions of all of the users. If the other users behave similar to u, for example, it might be hard to determine which observed destination belongs to u, while if they rarely choose the same destinations as u, it might be easy to figure out u's destination. We consider how this expectation varies with the other users' destination distributions p^v, In particular, we examine the maximum expectation for a given user u over the p^v, for all $v \neq u$, as this provides a lower bound to the user on his anonymity. We continue by examining the expectation in a more likely situation.

6.3.1 Model Equivalence

What makes the black-box model abstraction useful is that relationship anonymity preserved under a natural map from the protocol configurations of Definition 8 to black-box configurations. Let $\mathcal{C}_p$ be the set of protocol configurations and $\mathcal{C}_b$ be the set of black-box configurations. Let $\phi : \mathcal{C}_p \to \mathcal{C}_b$ be the function such that $\phi(C^p) = (C^b_D, C^b_I, C^b_O)$, where for all $u \in U$

$$\begin{aligned}
C^b_D(u) &= C^p_{l+1}(u) \\
C^b_I(u) &= \begin{cases} 1 & \text{if } u \in A \vee C^p_1(u) \in A \\ 0 & \text{otherwise} \end{cases} \\
C^b_O(u) &= \begin{cases} 1 & \text{if } C^p_l(u) \in A \\ 0 & \text{otherwise} \end{cases}
\end{aligned}$$

The black-box model essentially discards all information in a protocol configuration except the user destinations, whether or not the adversary observes a user, and whether or not the adversary observes a destination. By Corollary 3, these properties of a configuration determine relationship anonymity of the configuration. Indeed, user u has possibilistic relationship anonymity in protocol configuration C^p_1 if and only if u has possibilistic relationship anonymity in black-box configuration $\phi(C^p_1)$. In fact, with an appropriate probability distribution over executions of the protocol configurations, ϕ preserves probabilistic anonymity.

6.4 Expected Anonymity

Let the set $\mathcal{C}$ of all configurations be the sample space and X be a random configuration. X is then distributed according to Equation 6.1. Let Y be the posterior probability of the event that u chooses d as a destination, that is, $Y(C) = Pr[X_D(u) = d | X \approx C]$. Y is our metric for the relationship anonymity of u and d.

6.4.1 Calculation and Bounds

Let $\mathbb{N}^\Delta$ represent the set of multisets over Δ. Let $\rho(\Delta^0)$ be the number of permutations of $\Delta^0 \in \mathbb{N}^\Delta$ that only permute elements of the same type:

$$\rho(\Delta^0) = \prod_{\delta \in \Delta} |\{\delta \in \Delta^0\}|!.$$

Let $\Pi(A, B)$ be the set of all injective maps $A \to B$. The following theorem gives an exact expression for the conditional expectation of Y in terms of the underlying parameters U, Δ, p, and b:

Theorem 6.

$$
\begin{aligned}
E[Y|X_D(u) = d] = \quad & b(1-b)p^u_d + b^2 \\
& + \textstyle\sum_{S\subseteq U:u\in S} \sum_{\Delta^0\in\mathbb{N}^\Delta:|\Delta^0|\leq S} b^{n-|S|+|\Delta^0|}(1-b)^{2|S|-|\Delta^0|} \\
& \left(\begin{array}{l} \sum_{T\subseteq S-u:|T|=|\Delta^0|-1} \sum_{\pi\in\Pi(T+u,\Delta^0):\pi(u)=d} p^u_d \prod_{v\in T} p^v_{\pi(v)} \\ + \sum_{T\subseteq S-u:|T|=|\Delta^0|} \sum_{\pi\in\Pi(T,\Delta^0)} p^u_d \prod_{v\in T} p^v_{\pi(v)} \end{array} \right)^2 \\
& [\rho(\Delta^0)]^{-1}(p^u_d)^{-1} \left(\textstyle\sum_{T\subseteq S:|T|=|\Delta^0|} \sum_{\pi\in\Pi(T,\Delta^0)} \prod_{v\in T} p^v_{\pi(v)} \right)^{-1}
\end{aligned}
\tag{6.2}
$$

Proof. At a high level, the conditional expectation of Y can be expressed as

$$
\mathrm{E}[Y|X_D(u) = d] = \sum_{C\in\mathcal{C}} \Pr[X = C|X_D(u) = d]Y(C)
$$

We calculate Y for a configuration C by finding the relative weight of indistinguishable configurations in which u selects d. The adversary observes some subset of the circuits. If we match the users to circuits in some way that sends users with observed inputs to their own circuits, the result is an indistinguishable configuration. Similarly, we can match circuits to destinations in any way that sends circuits on which the output has been observed to their actual destinations in C.

The value of $Y(C)$ is especially simple if u's input has been observed. If the output has not also been observed, then $Y(C) = p^u_d$. If the output has also been observed, then $Y(C) = 1$.

For the case in which u's input has not been observed, we have to take into account the destinations of and observations on the other users. Let $S \subseteq U$ be the set of users s such that $C_I(s) = 0$. Note that $u \in S$. Let Δ^0 be the multiset of the destinations of circuits in C on which the input has not been observed, but the output has.

Let $f_0(S, \Delta^0)$ be the probability that in a random configuration the set of unobserved inputs is S and the set of observed destinations with no corresponding observed input is Δ^0:

$$
f_0(S, \Delta^0) = b^{n-|S|+|\Delta^0|}(1-b)^{2|S|-|\Delta^0|}[\rho(\Delta^0)]^{-1} \sum_{T\subseteq S:|T|=|\Delta^0|} \; \sum_{\pi\in\Pi(T,\Delta^0)} \; \prod_{v\in T} p^v_{\pi(v)}.
$$

Let $f_1(S, \Delta^0)$ be the probability that in a random configuration the set of unobserved inputs is S, the set of observed destinations with no corresponding observed input is Δ^0, the output of u is observed, and the destination of u is d:

$$f_1(S, \Delta^0) = b^{n-|S|+|\Delta^0|}(1-b)^{2|S|-|\Delta^0|}[\rho(\Delta^0)]^{-1} p^u_d \sum_{T \subseteq S-u:|T|=|\Delta^0|-1} \quad \sum_{\pi \in \Pi(T+u,\Delta^0):\pi(u)=d} \quad \prod_{v \in T} p^v_{\pi(v)}.$$

Let $f_2(S, \Delta^0)$ be the probability that in a random configuration the set of unobserved inputs is S, the set of observed destinations with no corresponding observed input is Δ^0, the output of u is unobserved, and the destination of u is d:

$$f_2(S, \Delta^0) = b^{n-|S|+|\Delta^0|}(1-b)^{2|S|-|\Delta^0|}[\rho(\Delta^0)]^{-1} p^u_d \sum_{T \subseteq S-u:|T|=|\Delta^0|} \quad \sum_{\pi \in \Pi(T,\Delta^0)} \quad \prod_{v \in T} p^v_{\pi(v)}.$$

Now we can express the posterior probability $Y(C)$ as

$$Y(C) = \frac{f_1(S, \Delta^0) + f_2(S, \Delta^0)}{f_0(S, \Delta^0)}. \tag{6.3}$$

The expectation of Y is a sum of the above posterior probabilities weighted by their probability. The probability that the input of u has been observed but the output hasn't is $b(1-b)$. The probability that both the input and output of u have been observed is b^2. These cases are represented by the first two terms in Equation 6.2.

When the input of u has not been observed, we have an expression of the posterior in terms of sets S and Δ^0. The numerator $(f_1 + f_2)$ of Equation 6.3 itself sums the weight of every configuration that is consistent with S, Δ^0, and the fact that the destination of u is d. However, we must divide by p^u_d, because we condition on the event $\{X_D(u) = d\}$.

These observations give us the final summation in Equation 6.2. □

The expression for the conditional expectation of Y in Equation 6.2 is not easy to interpret. It would be nice if we could find a simple approximation. The probabilistic analysis in [65] proposes just such a simplification by reducing it to only two cases: *i*) the adversary observes the user's input and output and therefore identifies his destination and *ii*) the adversary doesn't observe these and cannot improve his *a priori* knowledge. The corresponding simplified expression for the expection is

$$E[Y|X_D(u) = d] \approx b^2 + (1-b^2)p^u_d. \tag{6.4}$$

This is a reasonable approximation if the final summation in Equation 6.2 is about $(1-b)p^u_d$. This summation counts the case in which u's input is not observed, and to achieve a good approximation the adversary must experience no significant advantage or disadvantage from comparing the users with unobserved inputs (S) with the discovered destinations (Δ^0).

The quantity $(1-b)p_d^u$ does provide a lower bound on the final summation. It may seem obvious that considering the destinations in Δ^0 can only improve the accuracy of adversary's prior guess about u's destination. However, in some situations the posterior probability for the correct destination may actually be smaller than the prior probability. This may happen, for example, when some user v, $v \neq u$, communicates with a destination e, $e \neq d$, and only u is *a priori* likely to communicate with e. If the adversary observes the communication to e, it may infer that it is likely that u was responsible and therefore didn't choose d.

It is true, however, that in expectation this probability can only increase. Therefore Equation 6.4 provides a lower bound on the expected anonymity.

The proof of this fact relies on the following lemma. Let $\mathcal{E}$ be an event in some finite sample space Ω. Let $\mathcal{A}_1, \ldots, \mathcal{A}_n$ be a set of disjoint events such that $\mathcal{E} \subseteq \bigcup_i \mathcal{A}_i$, and let $\mathcal{A}^j = \bigcup_{i=1}^j \mathcal{A}_i$. Let $\mathcal{E}_i = \mathcal{E} \cap \mathcal{A}_i$. Finally, let $Z(\omega) = \sum_i 1_{\mathcal{E}_i} Pr[\mathcal{E}_i]/Pr[\mathcal{A}_i]$ (where $1_{\mathcal{E}_i}$ is the characteristic function for $\mathcal{E}_i$). $Z(\omega)$ is thus the conditional probability $Pr[\mathcal{E}|\mathcal{A}_i]$, where $\omega \in \mathcal{E}_i$.

Lemma 5. $Pr[\mathcal{E}|\mathcal{A}^n] \leq E[Z|\mathcal{E}]$

Proof.

$$\begin{aligned}
Pr[\mathcal{E}|\mathcal{A}^n] &= \tfrac{Pr[\mathcal{E}]}{Pr[\mathcal{A}^n]} \\
&= \frac{\left(\sum_i \frac{Pr[\mathcal{E}_i]\sqrt{Pr[\mathcal{A}_i]}}{\sqrt{Pr[\mathcal{A}_i]}}\right)^2}{Pr[\mathcal{A}^n]Pr[\mathcal{E}]} && \text{by a simple rewriting} \\
&\leq \frac{\left(\sqrt{\sum_i \frac{(Pr[\mathcal{E}_i])^2}{Pr[\mathcal{A}_i]}}\sqrt{\sum_i Pr[\mathcal{A}_i]}\right)^2}{Pr[\mathcal{A}^n]Pr[\mathcal{E}]} && \text{by the Cauchy-Schwartz inequality} \\
&= \sum_i \tfrac{(Pr[\mathcal{E}_i])^2}{Pr[\mathcal{A}_i]Pr[\mathcal{E}]} \\
&= E[Z|\mathcal{E}]
\end{aligned}$$

□

Theorem 7. $E[Y|X_D(u) = d] \geq b^2 + (1-b^2)p_d^u$

Proof. As described in the proof of Theorem 6

$$E[Y|X_D(u) = d] = b^2 + b(1-b)p_d^u + (1-b)E[Y|X_D(u) = d \wedge X_I(u) = 0]$$

To apply Lemma 5, take the set of configurations $\mathcal{C}$ to be the sample space Ω. Take $\{X_D(u) = d\}$ to be the event $\mathcal{E}$. Take the indistinguishability equivalence relation to be the sets $\mathcal{A}_i$. Finally, take Y to be Z. Then the lemma shows that $E[Y|X_D(u) = d \wedge X_I(u) = 0] \geq p^u_d$. □

To examine the accuracy of our approximation, we look at how large the final summation in Equation 6.2 can get as the users' destination distributions vary. Because this is the only term that varies with the other user distributions, this will also provide a worst-case guarantee on expected anonymity. Our results will show that the worst case can occur when the users other than u act as differently from u as possible by always visiting the destination u is otherwise least likely to visit. Less obviously, we show that the maximum can also occur when the users other than u always visit d. This happens because it makes the adversary observe destination d often, causing him to suspect that u chose d. Our results also show that the worst-case expectation is about $b + (1-b)p^u_d$, which is significantly worse than the simple approximation above.

As the first step in finding the maximum of Equation 6.2 over $(p^v)_{v \neq u}$, we observe that it is obtained when every user $v \neq u$ chooses only one destination d_v, *i.e.* $p^v_{d_v} = 1$ for some $d_v \in \Delta$.

Lemma 6. *A maximum of $E[Y|X_D(u) = d]$ over $(p^v)_{v \neq u}$ must occur when, for all $v \neq u$, there exists some $d_v \in \Delta$ such that $p^v_{d_v} = 1$.*

Proof. Take some user $v \neq u$ and two destinations $e, f \in \Delta$. Assign arbitrary probabilities in p^v to all destinations except for f, and let $\zeta = 1 - \sum_{\delta \neq e,f} p^v_\delta$. Then $p^v_f = \zeta - p^v_e$. Consider $E[Y|X_D(u) = d]$ as a function of p^v_e. The terms t_i of $E[Y|X_D(u) = d]$ that correspond to any fixed S and Δ^0 are of the following general form, where $\alpha_i, \beta_i, \gamma_i, \delta_i, \epsilon_i, \eta_i \geq 0$:

$$t_i = \frac{(\alpha_i p^v_e + \beta_i(\zeta - p^v_e) + \gamma_i)^2}{\delta_i p^v_e + \epsilon_i(\zeta - p^v_e) + \eta_i}.$$

This is a convex function of p^v_e:

$$t_i'' = \frac{2(\gamma_i(\delta_i - \epsilon_i) + \beta_i(\delta_i\zeta + \eta_i) - \alpha_i(\epsilon_i\zeta + \eta_i))^2}{(\epsilon_i(\zeta - p^v_e) + \delta_i p^v_e + \eta_i)^3} \geq 0$$

The leading two terms of $E[Y|X_D(u) = d]$ are constant in p^v, and the sum of convex functions is a convex function, and so $E[Y|X_D(u) = d]$ is convex in p^v_e. Therefore, a maximum of $E[Y|X_D(u) = d]$ must occur when $p^v_e \in \{0, 1\}$. □

The following lemma shows that we can further restrict ourselves to distribution vectors in which, for every user except u, the user either always chooses d or always chooses the destination that u is otherwise least likely to visit.

Lemma 7. *Order the destinations $d = d_1, \ldots, d_{|\Delta|}$ such that $p^u_{d_i} \geq p^u_{d_{i+1}}$ for $i > 1$. Then a maximum of $E[Y|X_D(u) = d]$ must occur when, for all users v, either $p^v_{d_1} = 1$ or $p^v_{d_{|\Delta|}} = 1$.*

Proof. Assume, following Lemma 6, that $(p^v)_{v \neq u}$ is an extreme point of the set of possible distribution vectors.

Equation 6.2 groups configurations first by the set S with unobserved inputs and second by the observed destinations Δ^0. Instead, group configurations first by S and second by the set $T \subseteq S$ with observed outputs. Because every user except u chooses a destination deterministically, Y only depends on the sets S and T. Therefore we can use the notation $Y(S, T)$ without ambiguity.

$$\begin{aligned} E[Y|X_D(u) = d] = \quad & b(1-b)p^u_d + b^2 + \\ & \textstyle\sum_{S:u \in S} \sum_{T:T \subseteq S} b^{n-|S|+|T|}(1-b)^{2|S|-|T|} Y(S,T) \end{aligned} \tag{6.5}$$

Select two destinations $d_i, d_j, 1 < i < j$. We break up the sum in Equation 6.5 and show that, for every piece, the sum can only be increased by changing $(p^v)_v$ so that any user that always chooses d_i always chooses d_j instead.

Fix $S \subseteq U, u \in S$. Let $S_i, S_j \subseteq S$ be such that $p^s_{d_i} = 1$ if and only if $s \in S_i$, and $p^s_{d_j} = 1$ if and only if $s \in S_j$. Fix $T' \subseteq S \backslash S_i \backslash S_j$ and some $l \geq |T'|$. Let s_{d_k} be the number of users in $S - u$ that always visit d_k and t_{d_k} be the number of users in $T' - u$ that always visit d_k. Let $f(S, T')$ be the sum of terms in Equation 6.5

that are indexed by S and some T such that $|T| = l$ and $T \supseteq T'$. Let $m = l - |T'|$.

$$\begin{aligned}
f(S,T') &= b^{n-|S|+l}(1-b)^{2|S|-l} \sum_{k=0}^{m} \binom{s_{d_i}}{k}\binom{s_{d_j}}{m-k} \prod_{e\in\Delta\setminus\{d,d_i,d_j\}} \binom{s_e}{t_e} \\
&\quad p_d^u \left(\binom{s_d}{t_d-1}\binom{s_{d_i}}{k}\binom{s_{d_j}}{m-k} + \binom{s_d}{t_d}\binom{s_{d_i}}{k}\binom{s_{d_j}}{m-k} \right) \\
&\quad \left(\begin{array}{l} \prod_{e\in\Delta\setminus\{d_i,d_j\}} \binom{s_e}{t_e}\binom{s_{d_i}}{k-1}\binom{s_{d_j}}{m-k} p_{d_i}^u + \prod_{e\in\Delta\setminus\{d_i,d_j\}} \binom{s_e}{t_e}\binom{s_{d_i}}{k}\binom{s_{d_j}}{m-k-1} p_{d_j}^u \\ + \sum_{f\in\Delta\setminus\{d_i,d_j\}} \prod_{e\in\Delta\setminus\{f,d_i,d_j\}} \binom{s_e}{t_e}\binom{s_f}{t_f-1}\binom{s_{d_i}}{k}\binom{s_{d_j}}{m-k} p_f^u + \prod_{e\in\Delta\setminus\{d_i,d_j\}} \binom{s_e}{t_e}\binom{s_{d_i}}{k}\binom{s_{d_j}}{m-k} \end{array} \right)^{-1} \\
&= b^{n-|S|+l}(1-b)^{2|S|-l} \prod_{e\in\Delta\setminus\{d_i,d_j\}} \binom{s_e}{t_e} p_d^u \left(\frac{t_d}{s_d+1-t_d} + 1 \right) \\
&\quad \sum_{k=0}^{m} \binom{s_{d_i}}{k}\binom{s_{d_j}}{m-k} \frac{(s_{d_i}+1-k)(s_{d_j}+1-m+k)}{\left(\begin{array}{l} p_{d_i}^u k(s_{d_j}+1-m+k) + p_{d_j}^u (m-k)(s_{d_i}+1-k) \\ +(s_{d_i}+1-k)(s_{d_j}+1-m+k)\left(\sum_{f\in\Delta\setminus\{d_i,d_j\}} p_f^u \frac{t_f}{s_f+1-t_f} + 1\right) \end{array} \right)} \\
&= \alpha \sum_{k=0}^{m} \binom{s_{d_i}}{k}\binom{s_{d_j}}{m-k} \frac{(s_{d_i}+1-k)(s_{d_j}+1-m+k)}{\left(\begin{array}{l} p_{d_i}^u (s_{d_i}+1)(s_{d_j}+1-m+k) + p_{d_j}^u (s_{d_j}+1)(s_{d_i}+1-k) \\ +(s_{d_i}+1-k)(s_{d_j}+1-m+k)\beta \end{array} \right)} \\
\alpha,\beta &\geq 0
\end{aligned}$$

This can be viewed as the weighted convolution of binomial coefficients. Unfortunately, there is no obvious way to simplify the expression any further to find the maximum as we trade off s_{d_i} and s_{d_j}. There is a closed-form sum if the coefficient of the binomial product is a fixed-degree polynomial, however. Looking at the coefficient, we can see that it is concave.

$$\begin{aligned}
c_k &= \tfrac{(s_{d_i}+1-k)(s_{d_j}+1-m+k)}{p_{d_i}^u(s_{d_i}+1)(s_{d_j}+1-m+k)+p_{d_j}^u(s_{d_j}+1)(s_{d_i}+1-k)+(s_{d_i}+1-k)(s_{d_j}+1-m+k)\beta} \\
\tfrac{d^2 c_k}{dk^2} &= -\frac{\left(\begin{array}{l} 2((s_{d_i}+1)(s_{d_j}+1)(2+s_{d_i}+s_{d_j}-m)^2 p_{d_i}^u p_{d_j}^u + \\ b((s_{d_i}+1)(s_{d_j}+1+k-m)^3 p_{d_i}^u + (s_{d_j}+1)(s_{d_i}+1-k)^3 p_{d_j}^u)) \end{array} \right)}{\scriptstyle ((s_{d_j}+1+k-m)(b(s_{d_i}+1-k)+p_{d_i}^u(s_{d_i}+1))+(s_{d_j}+1)(s_{d_i}+1-k)p_{d_j}^u)^3} \leq 0
\end{aligned}$$

We can use this fact to bound the sum above by replacing c_k with a line tangent at some point k_0. Call this approximation $\tilde{f}$. Holding $s_{d_i} + s_{d_j}$ constant, this approximation is in fact equal at $s_{d_i} = 0$ because the sum has only one term. Then if $s_{d_i} = 0$ still maximizes the sum, the theorem is proved.

$$
\begin{aligned}
f(S,T') &\leq \sum_{k=0}^{m} \binom{s_{d_i}}{k}\binom{s_{d_j}}{m-k}(c'_{k_0}(k-k_0)+c_{k_0}) \\
&= \binom{s_{d_i}+s_{d_j}}{m}\left(c_{k_0}+c'_{k_0}\frac{m\cdot s_{d_i}}{s_{d_i}+s_{d_j}}-c'_{k_0}k_0\right) \\
&= \tilde{f}(S,T')
\end{aligned}
$$

The linear approximation will be done around the point $k_0 = m \cdot s_{d_i}/(s_{d_i}+s_{d_j})$. This results in a simple form for the resulting approximation, and also the mass of the product of binomial coefficients concentrates around this point. Set $\nu = s_{d_i} + s_{d_j}$ to examine the tradeoff between s_{d_i} and s_{d_j}.

$$
\begin{aligned}
\tilde{f}(S,T') &= \binom{\nu}{m}(c_{\frac{m\cdot s_{d_i}}{\nu}}) \\
&= \binom{\nu}{m}\frac{((\nu-s_{d_i})(\nu-m)+\nu)((s_{d_i}+1)\nu-m\cdot s_{d_i})}{\left(\begin{array}{l} p^u_{d_i}\nu(s_{d_i}+1)((\nu-s_{d_i})(\nu-m)+\nu)+ \\ p^u_{d_j}\nu(\nu-s_{d_i}+1)(\nu+s_{d_i}(\nu-m))+ \\ \beta((s_{d_i}+1)\nu-m\cdot s_{d_i})((\nu-s_{d_i})(\nu-m)+\nu) \end{array}\right)}
\end{aligned}
$$

Direct calculation can easily show that $\frac{d^2\tilde{f}}{d(s_{d_i})^2} \geq 0$. Then it is straightforward to show that $f(s_{d_i}=0) \geq f(s_{d_j}=0)$.

□

Therefore, in looking for a maximum we can assume that every user except u either always visits d or always visits $d_{|\Delta|}$. We can use the same idea with d and $d_{|\Delta|}$ as was used with d_i and d_j and consider $E[Y|X_D(u)=d]$ as we trade off the number of users visiting them. Doing this shows that a maximum is obtained either when all users but u always visit d or when they always visit $d_{|\Delta|}$. This is the worst-case expected anonymity.

Theorem 8. *A maximum of $E[Y|X_D(u)=d]$ occurs when either $p^v_d = 1$ for all $v \neq u$ or when $p^v_{d_{|\Delta|}} = 1$ for all $v \neq u$.*

Proof. Assume, following Lemma 7, that $(p^v)_{v\neq u}$ is such that $p^v_d = 1$ or $p^v_{d_{|\Delta|}} = 1$ for all $v \neq u$.

The expected posterior probability can be given in the following variation on Equation 6.5:

$$E[Y|X_D(u) = d] = b(1-b)p_d^u + b^2 + \sum_{S:u\in S} \sum_{t=0}^{|S|} b^{n-|S|+t}(1-b)^{2|S|-t} Y(S,t)$$

Here $Y(S,t)$ is the expectation of Y conditional on the event that S is the set of users with unobserved inputs and t is the size of the subset of S with observed outputs.

Take some arbitrary S containing u. Let $s = |S|$, and let $d = |\{s \in S - u : p_d^s = 1\}|$. $Y(S,t)$ can be expressed as follows, where k represents the number of observed outputs of users in S with destination d:

$$Y(S,t) = \sum_{k=0}^{t} \binom{d+1}{k}\binom{s-d-1}{t-k} \left[\frac{p_d^u \binom{d}{k-1}\binom{s-d-1}{t-k} + p_d^u \binom{d}{k}\binom{s-d-1}{t-k}}{p_d^u \binom{d}{k-1}\binom{s-d-1}{t-k} + p^u_{d_{|\Delta|}} \binom{d}{k}\binom{s-d-1}{t-k-1} + \binom{d}{k}\binom{s-d-1}{t-k}} \right]$$

$$= \sum_{k=0}^{t} \binom{d+1}{k}\binom{s-d-1}{t-k} \left[\frac{p_d^u(d+1)(s-d-t+k)}{\begin{array}{l} p_d^u(d+1)(s-d-t+k) + p^u_{d_{|\Delta|}}(d+1-k)(s-d) \\ +(1-p_d^u - p^u_{d_{|\Delta|}})(d+1-k)(s-d-t+k) \end{array}} \right]$$

Because $Y(S,t)$ only depends on s,d, and t, we overload the notation and let $Y(s,t,d) = Y(S,t)$. The makes the dependence on d explicit. We will show that $Y(s,t,d)$ achieves a maximum over d at either $d = 0$ or $d = s-1$, and that the maximizing value of d doesn't depend on s or t. These facts imply the theorem.

As in Theorem 7, we can view the expression for $Y(s,t,d)$ as the weighted convolution of binomial coefficients:

$$Y(s,t,d) = \sum_{k=0}^{t} \binom{d+1}{k}\binom{s-d-1}{t-k} a(d,k) \qquad (6.6)$$

To make the analysis of this sum more tractable, we will approximate $a(d,k)$ with a function $\tilde{a}(d,k)$ that is linear in k. In the following analysis, we are only concerned with values k such that the binomial coefficients $\binom{d+1}{k}$ and $\binom{s-d-1}{t-k}$ are nonzero, that is, that satisfy $\max(0, t-s+d+1) \leq k \leq \min(d+1, t)$.

Observe that $\frac{\partial^2 a}{\partial k^2} \neq 0$. This is clear from a direct calculation of the second derivative. Therefore it is of constant sign. We will use this fact and linearly approximate a with both a tangent line and a line through the endpoints. One of the two approximations will overestimate $a(d,k)$ for all k in range for a fixed d. Therefore one of the lines will overestimate $Y(s,t,d)$ at that value d.

The first approximation overestimates a when it is concave. Let $k_0 = t(d+1)/s$. Also let a' refer to $\frac{\partial a}{\partial k}$.

The linear approximation $\tilde{a}$ is the tangent at k_0:

$$\tilde{a}(d,k) = (k-k_0)a'(d,k_0) + a(d,k_0)$$

Then there is a closed-form for the sum using this approximation:

$$\begin{aligned}\tilde{Y}(s,t,d) &= \sum_{k=0}^{t} \binom{d+1}{k}\binom{s-d-1}{t-k}\tilde{a}(d,k) \\ &= \binom{s}{t}a(d,k_0)\end{aligned}$$

A calculation of $\frac{\partial \tilde{Y}}{\partial d}$ shows that it is nonnegative. Therefore a maximum of $\tilde{Y}$ is found at $d = n-1$.

The second approximation overestimates a when it is convex. The endpoints of the range of k are $k_0 = \max(0, t-s+d+1)$ and $k_1 = \min(d+1, t)$. Let $\tilde{a}$ be the line going through the function at these points:

$$\tilde{a}(d,k) = \frac{(a(d,k_1)-a(d,k_0))(k-k_0)}{k_1-k_0} + a(d,k_0)$$

Substituting $\tilde{a}$ for a in the expression for $Y(s,t,d)$ we get:

$$\tilde{Y}(s,t,d) = \binom{s}{t}a(d,k_0) + \binom{s-1}{t-1}\frac{(d+1)(a(d,k_1)-a(d,k_0))}{k_1-k_0}$$

There are four possible cases for the values of k_1 and k_0.

1. $t \leq s-d-1 \wedge t \leq d+1$

   In this case $k_0 = 0$, $k_1 = t$, and $d \in [t-1, s-1-t]$. Then it can be shown that $\tilde{Y}(s,t,t-1) \geq \tilde{Y}(s,t,d)$. Therefore the maximum of $\tilde{Y}$ is obtained at $d = t-1$.

2. $t \leq s-d-1 \wedge t \geq d+1$

   In this case $k_0 = 0$, $k_1 = d+1$, and $d \in [0, \min(s-t-1, t-1)]$. It can be easily shown that $\frac{\partial \tilde{Y}}{\partial d} \leq 0$. Therefore the maximum of $\tilde{Y}$ is obtained at $d = 0$.

3. $t \geq s-d-1 \wedge t \leq d+1$

   In this case $k_0 = t-s+d+1$, $k_1 = t$, and $d \in [\max(s-t-1, t-1), s-1]$. It can be shown that if $\frac{\partial^2 \tilde{Y}}{\partial d^2} \leq 0$ then $\frac{\partial \tilde{Y}}{\partial d} \geq 0$. Therefore the maximum of $\tilde{Y}$ is obtained at $d = \max(s-t-1, t-1)$ or $d = s-1$.

4. $t \geq s-d-1 \wedge t \geq d+1$

   In this case, $k_0 = t-s+d+1$, $k_1 = d+1$, and $d \in [s-t-1, t-1]$. It is easily shown that $\frac{\partial^2 \tilde{Y}}{\partial d^2} \geq 0$. Therefore the maximum of $\tilde{Y}$ is obtained at $d = s-t-1$ or $d = t-1$.

Thus we have the following set of points as candidate maxima: $\{0, s-t-1, t-1, s-1\}$.

When $t \leq s-t$, the intervals of d with the same endpoints are $[0, t-1]$, $[t-1, s-1-t]$, and $[s-1-t, s-1]$. We just showed that the maximum in these intervals is at 0, $t-1$, and $s-1-t$ or $s-1$, respectively. Therefore $\tilde{Y}(s,t,0) \geq \tilde{Y}(s,t,t-1) \geq \tilde{Y}(s,t,s-1-t)$, and so the maximum over the whole interval is at $d = 0$ or $d = s-1$.

When $t \geq s-t$, the intervals of d with the same endpoints are $[0, s-1-t]$, $[s-1-t, t-1]$, and $[t-1, s-1]$. We showed above that the maximum in these intervals is 0, $s-1-t$ or $t-1$, and $t-1$ or $s-1$, respectively. Therefore $\tilde{Y}(s,t,0) \geq \tilde{Y}(s,t,s-1-t)$, and so the maximum over the whole interval is at one of $\{0, t-1, s-1\}$. Examination of the expressions for $\tilde{Y}$ at these values shows that if $\tilde{Y}(s,t,0) \leq \tilde{Y}(s,t,t-1)$, then $\tilde{Y}(s,t,t-1) \leq \tilde{Y}(s,t,s-1)$. Therefore the maximum is always found at $d = 0$ or $d = s-1$.

The sum in Equation 6.6 for $Y(s,t,d)$ has only one term at $d = 0$ and $d = s-1$, and therefore $\tilde{Y}(s,t,d) = Y(s,t,d)$ at these points. $\tilde{Y}$ is an overestimate of Y, and thus $Y(s,t,d)$ is also a maximum at $d = 0$ or $d = s-1$. □

6.4.2 Asymptotic Anonymity

The exact value of the maximum of $E[Y|X_D(u) = d]$ is not simple to express, but we can give a straightforward approximation for large user populations n. We focus on large n, because anonymity networks, and onion routing in particular, are understood to have the best chance at providing anonymity when they have many users. Furthermore, Tor is currently used by an estimated 200,000 people.

Theorem 9. *When $p^v_{d_{|\Delta|}} = 1$, for all $v \neq u$,*

$$E[Y|X_D(u) = d] = b + b(1-b)p^u_d + (1-b)^2 p^u_d \left(\frac{1-b}{1-(1-p^u_{d_{|\Delta|}})b} + O\left(\sqrt{\frac{\log n}{n}}\right)\right). \tag{6.7}$$

Proof. Let $f(n)$ represent the conditional expectation of the posterior probability in the case that u's input and output are not observed. Observe that in the case that u's output is observed $Y = 1$, because u is the only user that visits d. Thus $E[Y|X_D(u) = d] = b + b(1-b)p^u_d + (1-b)^2 f(n)$.

When u's input and output are unobserved, the value of Y depends on the number of other users with unobserved inputs, s, and the number of those s users with observed outputs, t. We can then express f as:

$$
\begin{aligned}
f(n) &= E[Y|X_D(u) = d \wedge X_I(u) = 0 \wedge X_O(u) = 0] \\
&= \sum_{s=0}^{n-1} (1-b)^s b^{n-1-s} \binom{n-1}{s} \sum_{t=0}^{s} b^t (1-b)^{s-t} \binom{s}{t} \frac{p_d^u \binom{s}{t}}{p_{d_{|\Delta|}}^u \binom{s}{t-1} + \binom{s}{t}} \\
&= \sum_{s=0}^{n-1} (1-b)^s b^{n-1-s} \binom{n-1}{s} \sum_{t=0}^{s} b^t (1-b)^{s-t} \binom{s}{t} \frac{p_d^u (s-t+1)}{p_{d_{|\Delta|}}^u t + s - t + 1}
\end{aligned}
$$

Consider the inside sum. It is equal to the expected value of $g_0(s,T) = \frac{p_d^u(s-T+1)}{p_{d_{|\Delta|}}^u T + s - T + 1}$, where $T \sim Bin(s,b)$. As s gets large, Chernoff bounds on the tails show that they contribute little to the expectation. Let $\varepsilon_0(s)$ be the terms of the sum for which t is more than $\sqrt{s \log s}$ from its expectation, $\mu_0 = bs$:

$$
\begin{aligned}
\varepsilon_0(s) &= \sum_{t:|t-\mu_0|>\sqrt{s\log s}} b^t (1-b)^{s-t} \binom{s}{t} g_0(s,t) \\
&\leq \sum_{t:|t-\mu_0|>\sqrt{s\log s}} b^t (1-b)^{s-t} \binom{s}{t} && \text{because } g_0 \leq 1 \\
&\leq 2e^{-c_0 \log s / b} && \text{for any } c_0 < 1/2 \text{ and large } s \\
& && \text{by Chernoff's inequality} \\
&= s^{-c_1} && \text{for some } c_1 > 1/2
\end{aligned}
$$

Now we look at how much the values of g_0 inside the tail differ from the value of g_0 at its expectation. Let $\varepsilon_1(s,t) = g_0(s,t) - g_0(s,\mu_0)$ be this difference. It can be shown through direct calculation that $\frac{\partial \varepsilon_1}{\partial t} \leq 0$ and $\frac{\partial^2 \varepsilon_1}{\partial t^2} \leq 0$. Therefore, for values of t that are within $\sqrt{s \log s}$ of μ_0,

$$
\begin{aligned}
|\varepsilon_1(s,t)| &\leq \left| \varepsilon_1 \left(s, \mu_0 + \sqrt{s \log s} \right) \right| \\
&= \frac{p_d^u p_{d_{|\Delta|}}^u (1 + 1/s)}{\left(1 - \left(1 - p_{d_{|\Delta|}}^u\right) b + 1/s\right) \left(\sqrt{\frac{s}{\log s}} \left(1 - \left(1 - p_{d_{|\Delta|}}^u\right) b\right) - (1-p) + (s \log s)^{-1/2} \right)} \\
&= O\left(\sqrt{\frac{\log s}{s}} \right)
\end{aligned}
$$

Looking now at the outside sum we have

$$
f(n) = \sum_{s=0}^{n-1} (1-b)^s b^{n-1-s} \binom{n-1}{s} \left[\frac{p_d^u (s(1-b)+1)}{s\left(1 - \left(1 - p_{d_{|\Delta|}}^u\right) b\right) + 1} + O(s^{-c_1}) + O\left(\sqrt{\frac{\log s}{s}} \right) \right].
$$

This is the expectation of a function of a binomially distributed random variable with mean $\mu_1 = (1-b)(n-1)$. Let $\varepsilon_2(n)$ be the parts of this sum that are greater than $k(n-1)$ from μ_1, $k < \min(b, 1-b)$:

$$\begin{aligned}
\varepsilon_2(n) &\leq \sum_{s:|s-\mu_1|>k(n-1)} (1-b)^s b^{n-1-s} \binom{n-1}{s} && \text{because inner sum is at most 1} \\
&\leq 2e^{-c_2 k^2 (n-1)/(1-b)} && \text{for some } c_2 > 0 \text{ by Chernoff's inequality} \\
&= O(e^{-c_3 n}) && c_3 = c_2 k^2/(1-b)
\end{aligned}$$

Let g_1 be the non-vanishing inner term of $f(n)$:

$$g_1(s) = \frac{p_d^u (s(1-b)+1)}{s\left(1 - \left(1 - p_{d_{|\Delta|}}^u\right) b\right) + 1}.$$

As s grows it approaches a limit of

$$g_1^* = \frac{p_d^u (1-b)}{1 - \left(1 - p_{d_{|\Delta|}}^u\right) b}.$$

Let $\varepsilon_3(s) = g_1(s) - g_1^*$ be the difference from this limit. Direct calculation shows that $\frac{d\varepsilon_3}{ds} \leq 0$ and $\frac{d^2\varepsilon_3}{ds^2} \geq 0$. Therefore, for values of s within $k(n-1)$ of μ_1,

$$\begin{aligned}
|\varepsilon_3(s)| &\leq \varepsilon_3(\mu_1 - k(n-1)) \\
&= b p_d^u p_{d_{|\Delta|}}^u / \left[\left(1 - \left(1 - p_{d_{|\Delta|}}^u\right) b\right) \left((n-1)(1-b-k)\left(1 - \left(1 - p_{d_{|\Delta|}}^u\right) b\right) + 1\right)\right] \\
&= O(1/n)
\end{aligned}$$

Now we can show the desired asymptotic expression for the entire sum:

$$\begin{aligned}
f(n) &= O(e^{-c_3 n}) + \sum_{s=\mu_1-k(n-1)}^{\mu_1+k(n-1)} (1-b)^s b^{n-1-s} \binom{n-1}{s} \left[g_1^* + \varepsilon_3(s) + O(s^{-c_1}) + O\left(\sqrt{\frac{\log s}{s}}\right) \right] \\
&= g_1^* + O(e^{-c_3 n}) + O(1/n) + O(n^{-c_1}) + O\left(\sqrt{\frac{\log n}{n}}\right) \\
&= \frac{p_d^u (1-b)}{1 - \left(1 - p_{d_{|\Delta|}}^u\right) b} + O\left(\sqrt{\frac{\log n}{n}}\right)
\end{aligned}$$

□

Theorem 10. *When $p_d^v = 1$, for all $v \neq u$,*

$$E[Y|X_D(u) = d] = b^2 + b(1-b)p_d^u + (1-b)\frac{p_d^u}{1-(1-p_d^u)b} + O\left(\sqrt{\frac{\log n}{n}}\right). \tag{6.8}$$

Proof. The proof of this theorem is similar to that of Theorem 9.

Let $f(n)$ represent the conditional expectation of the posterior probability in the case that u's input is not observed. Thus $E[Y|X_D(u) = d] = b^2 + b(1-b)p_d^u + (1-b)f(n)$.

We can express f as

$$\begin{aligned}
f(n) &= E[Y|X_D(u) = d \wedge X_I(u) = 0] \\
&= \sum_{s=0}^{n-1}(1-b)^s b^{n-1-s}\binom{n-1}{s}\sum_{t=0}^{s+1} b^t(1-b)^{s+1-t}\binom{s+1}{t}\frac{p_d^u\binom{s}{t} + p_d^u\binom{s}{t-1}}{\binom{s}{t} + p_d^u\binom{s}{t-1}} \\
&= \sum_{s=0}^{n-1}(1-b)^s b^{n-1-s}\binom{n-1}{s}\sum_{t=0}^{s+1} b^t(1-b)^{s+1-t}\binom{s+1}{t}\frac{p_d^u(s+1)}{s-(1-p_d^u)t+1}.
\end{aligned}$$

As s gets large, Chernoff bounds on the tails of the inner sum show that they contribute little to the expectation. Let $g_0(s,t) = \frac{p_d^u(s+1)}{s-(1-p_d^u)t+1}$. Let $\varepsilon_0(s)$ be the terms of the inner sum for which t is more than $\sqrt{s \log s}$ from $\mu_0 = b(s+1)$:

$$\begin{aligned}
\varepsilon_0(s) &= \sum_{t:|t-\mu_0|>\sqrt{(s+1)\log(s+1)}} b^t(1-b)^{s+1-t}\binom{s+1}{t} g_0(s,t) \\
&\leq \sum_{t:|t-\mu_0|>\sqrt{(s+1)\log(s+1)}} b^t(1-b)^{s+1-t}\binom{s+1}{t} && \text{because } g_0 \leq 1 \\
&\leq 2e^{-c_0\log(s+1)/b} && \text{for any } c_0 < 1/2 \text{ and large enough } s \\
& && \text{by Chernoff's inequality} \\
&= (s+1)^{-c_1} && c_1 > 1/2
\end{aligned}$$

Let $\varepsilon_1(s,t) = g_0(s,t) - g_0(s,\mu_0)$. It can be shown through direct calculation that $\frac{\partial \varepsilon_1}{\partial t} \geq 0$ and $\frac{\partial^2 \varepsilon_1}{\partial t^2} \geq 0$.

Therefore, for values of t that are within $\sqrt{s\log s}$ of μ_0,

$$\begin{aligned}|\varepsilon_1(s,t)| &\le \varepsilon_1\left(s, \mu_0 + \sqrt{(s+1)\log(s+1)}\right)\\ &= \frac{p_d^u(1-p_d^u)}{(1-(1-p_d^u)b)\left(\sqrt{\frac{s+1}{\log(s+1)}}(1-(1-p_d^u)b)-(1-p)\right)}\\ &= O\left(\sqrt{\frac{\log s}{s}}\right)\end{aligned}$$

Looking now at the outside sum we have

$$f(n) = \sum_{s=0}^{n-1}(1-b)^s b^{n-1-s}\binom{n-1}{s}\left[\frac{p_d^u}{1-(1-p_d^u)b} + O(s^{-c_1}) + O\left(\sqrt{\frac{\log s}{s}}\right)\right].$$

Let $\varepsilon_2(n)$ be the parts of this sum that are greater than $k(n-1)$ from $\mu_1 = (n-1)(1-b)$, $k < \min(b, 1-b)$:

$$\begin{aligned}\varepsilon_2(n) &\le \sum_{s:|s-\mu_1|>k(n-1)}(1-b)^s b^{n-1-s}\binom{n-1}{s} && \text{because inner sum is at most 1}\\ &\le 2e^{-c_2k^2(n-1)/(1-b)} && \text{for some } c_2 > 0 \text{ by Chernoff's inequality}\\ &= O(e^{-c_3 n}) && c_3 = c_2k^2/(1-b)\end{aligned}$$

Now we can show the desired asymptotic expression for the entire sum:

$$\begin{aligned}f(n) &= O(e^{-c_3n}) + \sum_{s=(1-b-k)(n-1)}^{(1-b+k)(n-1)}(1-b)^s b^{n-1-s}\binom{n-1}{s}\left[\frac{p_d^u}{1-(1-p_d^u)b} + O(s^{-c_1}) + O\left(\sqrt{\frac{\log s}{s}}\right)\right]\\ &= \frac{p_d^u}{1-(1-p_d^u)b} + O(e^{-c_3n}) + O(n^{-c_1}) + O\left(\sqrt{\frac{\log n}{n}}\right)\\ &= \frac{p_d^u}{1-(1-p_d^u)b} + O\left(\sqrt{\frac{\log n}{n}}\right)\end{aligned}$$

□

To determine which distribution is the worst case for large n, simply examine the difference between the limits of the expressions in Theorems 9 and 10. It is clear from this that the worst-case distribution is $p_d^v = 1, \forall v \neq u$, only when $p_{d_{|\Delta|}}^u \geq \frac{(1-b)(1-p_d^u)^2}{p_d^u(1+b)-b}$. This happens when $p_d^u \geq 1/2$ and $p_{d_{|\Delta|}}^u$ is near $1-p_d^u$. For $p_{d_{|\Delta|}}^u$ small, which we would expect as it must be less than $1/|\Delta|$, the worst-case distribution is $p_{d_{|\Delta|}}^v = 1, \forall v \neq u$.

In this case the expected assigned probability is about $b + (1-b)p^u_d$. This can be viewed as decreasing the "innocence" of u from $1 - p^u_d$ to $(1-b)(1-p^u_d)$. It is also equal to the lower bound on anonymity in onion routing when the adversary controls a fraction $\sqrt{b}$ of the network.

6.5 Typical Distributions

It is unlikely that users of onion routing will ever find themselves in the worst-case situation. The necessary distributions just do not resemble what we expect user behavior to be like in any realistic use of onion routing. Our worst-case analysis may therefore be overly pessimistic. To get some insight into the anonymity that a typical user of onion routing can expect, we consider a more realistic set of users' destination distributions in which each user selects a destination from a common Zipfian distribution. This model of user behavior is used by Shmatikov and Wang [62] to analyze relationship anonymity in mix networks and is motivated by observations that the popularity of sites on the web follows a Zipfian distribution. Our results show that a user's expected assigned probability is close to $b^2 + (1-b^2)p^u_d$ for large populations, which is the best that can be expected when there is a b^2 probability of total compromise.

Let each user select his destination from a common Zipfian distribution p: $p_{d_i} = 1/(\mu i^s)$, where $s > 0$ and $\mu = \sum_{i=1}^{|\Delta|} 1/i^s$. It turns out that the exact form of the distribution doesn't matter as much as the fact that it is common among users.

Theorem 11. *When $p^v = p^w$, for all $v, w \in U$,*

$$E[Y|X_D(u) = d] = b^2 + (1-b^2)p^u_d + O(1/n)$$

Proof. Let p be the common destination distribution. The expected assigned probability can be expressed as

$$E[Y|X_D(u) = d] = b^2 + b(1-b)p^u_d + (1-b)\sum_{s=1}^{n} b^{n-s}(1-b)^{s-1}\binom{n-1}{s-1}$$
$$\sum_{t=0}^{s}(1-b)^{s-t}b^t\left[\binom{s-1}{t-1}\sum_{\Delta\in D^t:\Delta_1=d}\prod_{i=2}^{t}p_{\Delta_i}\psi(s,\Delta) + \binom{s-1}{t}\sum_{\Delta\in D^t}\prod_{i=1}^{t}p_{\Delta_i}\psi(s,\Delta)\right].$$

Here, s represents the size of the set of users with unobserved inputs, t represents the size of the subset of those s users that also have observed outputs, Δ represents the t observed destinations, and $\psi(s, \Delta)$ is the posterior probability.

Let $\Delta_d = |\{i : \Delta_i = d\}|$. Let $x^{\underline{n}}$ denote the falling power $x(x-1)\dots(x-n+1)$. The posterior probability ψ can be expressed simply as

$$\begin{aligned}\psi(s, \Delta) &= \frac{\Delta_d (s-1)^{\underline{t-1}} + p_d (s-1)^{\underline{t}}}{s^{\underline{t}}} \\ &= (\Delta_d + p_d(s-t))/s\end{aligned}$$

The sum $\sum_{\Delta \in D^t : \Delta_1 = d} \prod_{i=2}^{t} p_{\Delta_i} \psi(s, \Delta)$ calculates the expectation for ψ conditioned on s, t, and the destination of u being observed. The expression for ψ shows that this depends linearly on the expected value of Δ_d. This expectation is simply $1 + p_d(t-1)$, because one destination in this case is always d, and each of the other $t-1$ is d with probability p_d. The sum $\sum_{\Delta \in D^t} \prod_{i=1}^{t} p_{\Delta_i} \psi(s, \Delta)$ similarly depends linearly on the expectation of Δ_d, which in this case is $p_d t$.

With this observation, it is a straightforward calculation to show that the inner sum over t is simply:

$$b \frac{p_d(s-1) + 1}{s} + (1-b) p_d$$

We insert this into the larger sum and simplify:

$$\begin{aligned}E[Y | X_D(u) = d] &= b^2 + b(1-b)p_d^u + (1-b) \sum_{s=1}^{n} b^{n-s} (1-b)^{s-1} \binom{n-1}{s-1} \left[b \frac{p_d(s-1)+1}{s} + (1-b)p_d \right] \\ &= b^2 + b(1-b)p_d^u + (1-b) \left[b \left(p_d + \frac{(1-p_d)(1-(1-b)^{n+1})}{b(n+1)} \right) + (1-b)p_d \right] \\ &= b^2 + (1-b^2)p_d^u + O(1/n)\end{aligned}$$

□

Chapter 7

Improving Onion Routing Through Trust

7.1 Introduction

When designing or analyzing anonymous communication networks, researchers generally assume that all nodes routing traffic are equally trusted. But this typically is incorrect. There is much information available to those selecting routes that can affect trust: information about who runs some components of the infrastructure, what computing platforms are used, how long and how reliably some components have been running, etc. And if routing designs were to begin taking trust into account, then even more extensive and diverse bases for trust might be available.

Onion routing is a type of anonymous communication that creates cryptographic circuits along an unpredictable route through a network of nodes called *onion routers* and passes traffic bidirectionally along those circuits with minimal latency [29, 55, 18]. An adversary observing an entry node and an exit node of an onion-routing network through which one is, e.g., browsing the web can easily link the two ends of the connection and correlate source to destination. This has been an acknowledged feature of the design since its inception [65]. Correlation is easily done with extremely high confidence by *passive timing*, that is, simply by observing the timing pattern of data entering the network and of data exiting the network and matching incoming and outgoing patterns. Correlation can also be done with *active timing*, where the adversary inserts unique patterns in incoming data and observes where they appear among outgoing data. It is this vulnerability of onion routing circuits to hostile pairs of entry and exit nodes that is our focus. There are many documented attacks that have some effect on onion routing—correlation, congestion, intersection, destination fingerprinting, latency, etc. None of the others have the efficiency or certainty that correlation does when an attacker owns so little of the network (i.e., just one entry node and one exit node) and observes so little traffic.

Correlation is, at least in this way, the most significant unaddressed problem for onion routing and one that can likely be improved with trust knowledge. (Correlation could be countered by mixing, padding, or other approaches; however, to date no proposed countermeasure has had both low enough overhead and high enough expectation of success against realistic attackers to be pursued in practice.) This introduces many questions, such as whether using more trusted nodes helps profile or identify clients and what to do about that, how to model diverse trust assumptions, etc. But even ignoring these, it is not obvious how to take advantage of trust as a criterion in route selection. In particular, using trusted nodes more often has the disadvantage of simultaneously providing a small set of nodes for the adversary attempt to monitor. We focus on whether there is a way to use trust to reduce the probability of a circuit compromise by endpoints.

Tor [68] is the current widely-deployed and used public onion-routing network, with an estimated quarter-million concurrent users and a few thousand network nodes. It is thus useful to consider trust issues that arise for this deployed network. For example, a correlating adversary could try to compromise nodes in the network. Because Tor nodes are run by volunteers, however, an even easier attack is to simply set up hostile nodes and use those to attack traffic on the network. We have already noted that correlation attacks are strong and low cost. This shows us that they are also easy to deploy in practice.

One way Tor reduces the threat of linking exit activity to sources is by use of entry guards, a small number of nodes that a single client uses persistently to connect to the Tor network. If a client has chosen guard nodes that are not compromised, it can never be linked by correlation to its activity by a pair of compromised entry-exit nodes. When entry guards were introduced [52], there was a brief discussion of the relative merits of choosing guards randomly versus based on trust or other features of the guard nodes. So far, no one has analyzed the implications of choosing nodes based on trust. Entry guards are currently chosen randomly from the set of Tor nodes (subject to some performance and other criteria). Abusing entry-guard selection criteria can increase the chances of a node being chosen as an entry guard, especially if they are based on reliability, performance, etc. rather than based on any sort of trust. Many of the threats initially observed about this ([52, 3]) are not feasible in the current Tor network. Statistically, however, the percentage of all circuits compromised by hostile entry-exit pairs is not reduced by the use of randomly chosen entry guards, nor is the probability that any given client will have compromised guards; it only affects the distribution of compromised circuits over the client space. If one were able to choose not just guards but whole routes from a more trusted set of nodes, then one's threat of circuit compromise might be reduced. We hope through our analysis to show how best to add this protection to Tor and similar systems.

In this chapter we first set out a simple model that should facilitate reasoning about using trust in routing. We define trust simply to be the probability that an attempt by the adversary to control a node fails. We include a roving adversary that can attempt to compromise a certain number of nodes. Route selection is modeled as a three-stage game in which the user first picks a distribution over paths, then the adversary chooses a set of nodes to attempt to compromise, and finally the user samples a path from his distribution. While we expect this model to bear further fruit, we use it in this paper to show a number of results of both theoretical and practical interest.

We consider various strategies for choosing first and last nodes in the network so as to minimize the maximum probability a correlating adversary has for linking source to destination. We first look at the general case, in which there is an arbitrary number of trust levels. We observe that a straightforward algorithm to calculate an optimal distribution runs in time exponential in the size of the adversary. We consider a natural simplification of looking at distributions on individual nodes rather than pairs of nodes and considering the product distribution as an approximation of the joint distribution on pairs. We find two optimal distributions over single nodes, but we then show that optimal distributions on pairs are arbitrarily better than products of those optimal distributions on single nodes.

In practice, it is unlikely that one can realistically assign many different levels of trust, and so we next consider restricting to the case where there are only two trust levels for nodes in the network. Here we find three distributions and prove that in every case one of them must be optimal. Lastly, we discuss determining in practice when one of the three distributions is optimal based on the values of the system variables: trust values, size of the trusted and untrusted sets, and the size of the adversary.

7.2 Trust model

A user wants to use a network of onion routers for anonymous communication. He trusts some onion routers more than others in the sense that he trusts that they are less likely to attempt to compromise his anonymity. How should he take this trust into account when he selects his paths?

7.2.1 The model

To make this question concrete, we need to make the notions of trust, anonymity, and an adversary precise. We use the network and user models described in Chapter 2 and the onion-routing protocol described in Chapter 4.

We modify the adversary that is trying to compromise the user's anonymity. The adversary selects k routers in R that he will attempt to compromise and use for deanonymization. If a router is not selected, it cannot be used by the adversary in an attack.

When an onion router i is selected, the adversary fails to compromise it with probability t_i. This represents the user's trust in the router. It will be convenient to define $c_i = 1 - t_i$, the probability that the adversary *does* successfully compromise router i when he attempts to do so.

A user selects a path for a circuit from some probability distribution. If the adversary has selected and successfully compromised the first and last nodes on the chosen path, he can correlate timing patterns between the two and the user has no anonymity. To calculate the probability of such an event, we need only look at the user's distribution over entry-and-exit-node pairs.

We would like to find the probability distribution over pairs of routers that minimizes the maximum chance that the adversary can obtain of selecting both members of the pair and successfully compromising them. More precisely, we want to find $p \in \Delta_{n(n-1)/2}$, that is, a probability distribution p over pairs in R, that minimizes

$$c(p) = \max_{K \subseteq R : |K| = k} \sum_{\{r,s\} \in \binom{K}{2}} p(r,s) c_r c_s.$$

For a set S and $j \leq |S|$, we use $\binom{S}{j}$ to represent the collection of all subsets of S of size j. Also, for convenience, we write $p(\{r,s\})$ as $p(r,s)$.

This problem complements our results in Chapter 6. There, we analyze anonymity under uniformly-random path selection. Here, we consider improving anonymity by optimizing our path-selection distribution.

7.2.2 The adversary

The adversary's size represents a limit of his resources. While caution might suggest designing against an adversary that can compromise the entire network as a worst case, the optimal strategy in this case would be to select the two most-trusted routers. This fails our intuition that this solution allows the adversary to concentrate his efforts on these routers. And, especially for large diverse networks, it is typically unrealistic to assume that an adversary has the capacity to attack the entire network. System and protocol designs have been shown to provide a guarantee against various types of failure or compromise as long as no more than some fixed threshold of nodes is compromised at any time, e.g., Byzantine fault-tolerance.

The particular partial-network adversary from which our work derives is the roving adversary of Ostrovsky and Yung [51]. They introduced and were motivated by the concept of proactive security, in which an adversary could compromise arbitrary optimal sets of nodes given his current information. The roving adversary can potentially compromise every node in the network, but it can compromise no more than a fixed maximum number of nodes at any one time. Proactive security is concerned with properties that are resilient to such attacks. This can be useful for secret sharing and other distributed applications. The adversary model was applied to onion routing by Syverson et al. [65].

We alter the basic roving adversary model in two ways. First, to incorporate trust we add the idea that an adversary does not always succeed when attempting to compromise a node. Second, the adversary selects only one set to attack—there is only one move by the adversary. It may be useful to bring multiple rounds back in for future work. Though likely of limited use for individual correlation attacks (given the typically short duration of onion-routing circuits), roving could allow the adversary to learn various communication and trust properties of the network and its users.

The adversary is assumed to have prior knowledge of the distribution that is used to pick a route, and he uses this knowledge to pick the set of nodes that he will attempt to compromise. It is realistic in many settings to assume the adversary has such knowledge. For example, the probability distributions may be set in some software or common system parameters given to a wide group in which there is at least one compromised member. The adversary may also be able to infer trust information from outside knowledge about the user.

7.2.3 Trust

Trust is captured in our model with the probability t_i that the adversary's attempt to compromise a node fails. This notion accommodates several different means by which users in the real world might trust an onion router.

The probability might represent the user's estimate of how likely it is that the operator of a given node is trying to provide, rather than break, anonymity. It might represent the user's faith in the security of a given node against outside attack.

To arrive at such conclusions, the users must rely on some outside knowledge. This might include knowledge of the organizations or individuals who run nodes, both knowledge of their technical competence and the likelihood of themselves harboring ill intent. It also includes knowledge of computing platforms on which a network node is running, geopolitical information about the node, knowledge about the hosting facility where a node might be housed or the service provider(s) for its access to the underlying communications network, etc.

Admittedly, it may not be the case that one can realistically assign specific probabilities to each node in the network separately. It is for this reason that we consider in sections 7.5 and 7.6 restriction to just two trust levels. Even if one cannot be certain of the probability of compromise to assign at one level or another, one may be in a position to know the divergence of those levels. This is particularly the case if one is considering nodes run by, e.g., security or law-enforcement agencies of friendly governments or their contractors vs. the rest of the nodes on the network. Alternatively one can imagine sets of nodes run by reputable human rights groups, NGOs, or human rights agencies of friendly governments.

Unlike many other areas, network performance or reliability reputation are not good bases for trust for anonymous communication. That is because an adversary that is focused on learning as much as possible about communication patterns has incentive to run the highest performing, most reliable nodes in the network. Thus, many of the usual metrics do not apply.

7.2.4 Anonymity

We will consider a user to be anonymous unless the adversary has compromised the first and last routers on his path. This is motivated by the correlation attacks mentioned above. The model does not include some other methods the adversary can use, for example congestion attacks [49, 21], denial-of-service attacks [6], latency [32], or destination fingerprinting [31, 39]. It also does not take into account the total effect of an adversary's actions on a user's anonymity, such as the analysis performed in [25]. The attacks on which we focus are conceptually much simpler than these others, but more importantly, as noted in Section 7.1, none of these other attacks succeeds with as much certainty using as little resources as this one. Note that such entry-exit correlation attacks could also be done by the links from source to the entry onion router on the entry side and links from the exit onion router to the destination on the exit side (or by the destination itself). For example, an autonomous system or internet exchange on these links could participate in a correlation attack [23, 50]. We focus, however, on just the attack as it can be done by network nodes. Besides simplifying analysis, this is reasonable to model as a practical attack given the ease with which nodes can be added to the network.

Using this model, the user's selection of the pair constituting the first and last onion routers on his path is the only relevant factor in his anonymity. The user may make this selection using any probability distribution p over pairs of routers.

7.2.5 Objective function

We set as our objective function to find the distribution on pairs of routers that minimizes the probability of circuit compromise over all possible sets that the adversary could choose:

$$\min_{p \in \Delta_{n(n-1)/2}} \max_{K \subseteq R : |K| = k} \sum_{\{r,s\} \in \binom{K}{2}} p(r,s) c_r c_s .$$

This provides a worst-case guarantee, and if the user has a distribution with a low worst-case value, he is guaranteed anonymity with high probability regardless of the adversary's actions. As a worst-case criterion, however, it may direct the user to protect against adversarial actions that are unlikely. Indeed, while the adversary's goal is to find the subset $K \subseteq R$ that maximizes his chance of compromise, it is easy to see that this problem in general is equivalent to the **NP**-hard problem CLIQUE. Therefore the adversary may fail in many cases to actually select the worst-case set.

7.3 Strategies for the general case

Given arbitrary trust values $t_1, \ldots, t_n$, we would like to find a polynomial-time algorithm that takes as input the trust values and outputs an optimal or near-optimal distribution p^*.

7.3.1 Exact algorithm

There is a straightforward formulation of this problem as a linear program. Let the set of variables be p_{ij}, $i, j \in R$. The following constraints ensure that p is a probability distribution:

$$\begin{aligned} &\textstyle\sum_{\{r,s\} \in \binom{R}{2}} p_{rs} = 1 \\ &\quad 0 \le p_{rs} \le 1 \qquad \text{for all } \{r,s\} \in \binom{R}{2}. \end{aligned}$$

We want to find the distribution that satisfies the minimax criterion

$$\min_p \max_{K \in \binom{R}{k}} \sum_{\{r,s\} \in \binom{K}{2}} c_r c_s p(r,s).$$

For any fixed K, the sum

$$c(p,K) = \sum_{\{r,s\}\in\binom{K}{2}} p(r,s)c_r c_s$$

is linear in p. Therefore the minimax criterion minimizes the maximum of linear functions. We can thus transform it into a simple minimization problem by adding a slack variable t and some linear constraints. We force t to be greater than the maximum of our linear functions:

$$t - c(p,K) \geq 0 \quad \text{for all } K \in \binom{R}{k}$$

Then the objective function is simply $\min t$. Unfortunately, this linear program is of exponential size $\big(O(n^k)\big)$ because of the constraints for each subset.

7.3.2 Choosing a simple distribution

A straightforward simplification is to consider restricting the output to be a distribution in which the first and last routers are chosen independently and identically at random and then minimizing the probability that they are individually compromised.

Let p_R be a distribution on R. We consider the distribution p_R^* that minimizes the probability that an adversary chooses and successfully compromises a single router:

$$\begin{aligned} c(p_R) &= \max_{K\in\binom{R}{k}} \sum_{r\in K} p_R(r)c_r \\ p_R^* &= \operatorname*{argmin}_{p_R} c(p_R) \end{aligned}$$

The following theorem states that it is always optimal either to put all the probability on the most trusted router or to set the probabilities such that the values $c_i p_R(r_i)$ are equal for all $r_i \in R$.

Theorem 12. *Let $c_\mu = \min_j c_j$. Let p_R^1 put all the probability on the most trusted router:*

$$p_R^1(r) = \begin{cases} 1 & \textit{if } r = r_\mu \\ 0 & \textit{otherwise} \end{cases}$$

Let p_R^2 set probability inversely proportional to c_i:

$$p_R^2(r_i) = \alpha / c_i$$

where $\alpha = \left(\sum_i 1/c_i\right)^{-1}$.

Then

$$c(p_R^*) = \begin{cases} c(p_R^1) & \textit{if } c_\mu \leq k\alpha \\ c(p_R^2) & \textit{otherwise} \end{cases}$$

Proof. Suppose p_R is an optimal distribution. Sort the routers so that $c_1 p_R(r_1) \geq c_2 p_R(r_2) \geq \ldots \geq c_n p_R(r_n)$. The set K that maximizes $\sum_{r \in K} c_r p_R(r)$ is then $\{r_1, r_2, \ldots, r_k\}$, and the value of p_R is $c(p_R) = \sum_{i=1}^{k} c_i p_R(r_i)$.

Let l be the largest index such that $c_l p_R(r_l) = c_k p_R(r_k)$.

If $l < n$, we could decrease $c_i p_R(r_i), k \leq i \leq l$ by moving $\epsilon c_k / c_i$ probability from r_i to r_{l+1}. This decreases $c_i r_i$ by $c_k \epsilon$ and increases $c_{l+1} p_R(r_{l+1})$ by $\epsilon c_{l+1} c_k / c_i$. For small enough ϵ we maintain that if $i < j$ then $c_i p_R(r_i) \geq c_j p_R(r_j)$, and therefore we reduce the value $c(p_R)$. Therefore p_R cannot be optimal, contradicting our assumption.

Thus it must be that $l = n$. Let m be the smallest index such that $c_m p_R(r_m) = c_k p_R(r_k)$. Assume that p_R is an optimal distribution that has the smallest m possible.

If $m = 1$, we are in the case that $c_i p_R(r_i) = c_j p_R(r_j)$ for $1 \leq i, j \leq n$. This is the distribution p_R^2.

Suppose $m > 1$. If $p_R(r_m) = 0$, then $c(p_R) = \sum_{i=1}^{m-1} c_i p_R(r_i)$. Let $c_\mu = \min_i c_i$. Because all of the probability is contained in a set that the adversary can completely select, we do not increase $c(p_R)$ by moving all the probability to r_μ:

$$\begin{aligned} c(p_R) &= \sum_{i=1}^{m-1} c_i p_R(r_i) \\ &\geq \sum_{i=1}^{m-1} c_\mu p_R(r_i) \\ &= c_\mu . \end{aligned}$$

c_μ is equal to $c(p_R^1)$.

Now consider the case that $p_R(r_m) > 0$. Recall that $c_i p_R(r_i) = c_j p_R(r_j)$ for all pairs r_i,r_j, in the set $S = \{r_i, m \leq i \leq n\}$. Consider moving probability between r_{m-1} and S in a way that maintains the equality of $c_i p_R(r_i)$ for $r_i \in S$. This can be achieved by setting the probability of r_{m-1} to

$$p'_R(r_{m-1}, t) = p_R(r_{m-1}) + t$$

and the probability of $r_i \in S$ to

$$p'_R(r_i, t) = p_R(r_i) - \frac{t}{c_i(\sum_{r_j \in S} 1/c_j)}.$$

For small enough values of t, this preserves the property that if $i > j$ then $c_i p'_R(r_i, t) \leq c_j p'_R(r_j, t)$. Therefore $c(p'_R) = \sum_{i=1}^{k} c_i p'_R(r_i, t)$. The fact that p'_R is linear in t makes $c(p'_R)$ also linear in t for small enough values of t.

If $D_t c(p'_R)|_{t=0} \geq 0$, then for $t < 0$ large enough $c(p'_R)$ doesn't increase. This corresponds to moving probability from r_{m-1} to S, and the smallest t that maintains the ordering by $c_i p'_R(r_i)$ results in $c_{m-1} p'_R(r_{m-1}) = c_m p'_R(r_m)$. This contradicts the assumption about the minimality of the index m.

If $D_t c(p'_R)|_{t=0} \leq 0$, then for $t > 0$ small enough $c(p'_R)$ doesn't increase. This corresponds to moving probability from S to r_{m-1}. In fact, no positive value of t increases $c(p'_R)$. This is because setting $t > 0$ decreases the probability of all r_i, $i > k$, and only increases the probability of $r_{m-1}, m \leq k$, and thus preserves the fact that $c(p'_R) = \sum_{i=1}^{k} c_i p'_R(r_i, t)$. Therefore we can increase t until $c_i p'_R(r_i) = 0$ for all $r_i \in S$. This puts us in the case where $p'_R(r_m) = 0$, which we have already shown implies that $c(p'_R) \geq c(p^1_R)$.

Thus we have shown that either p^1_R or p^2_R is an optimal distribution. $c(p^1_R) = c_1$ and $c(p^2_R) = k\alpha$. Therefore, if $c_1 \leq k\alpha$, $c(p^*_R) = c(p^1_R)$, and otherwise $c(p^*_R) = c(p^2_R)$. □

We might hope that the product distributions $p^1_R \times p^1_R$ and $p^2_R \times p^2_R$ over $R \times R$ are good approximations to an optimal distribution p^*. However, this is not the case, and we can find inputs such that $c(p^i_R)/c(p^*)$, $i \in \{1, 2\}$, is arbitrarily high. In fact, we can show this for slightly improved distributions p^1 and p^2 over $\binom{R}{2}$.

Notice that $p^i_R \times p^i_R$, $i \in \{1, 2\}$, puts positive probability on the user choosing the same router twice. The problem as formulated in Section 7.2 allows distributions only over distinct pairs in $\binom{R}{2}$. This doesn't affect the optimum, however. There is always an optimal distribution that puts zero probability on $(r, r) \in R \times R$. Let p be a distribution on $R \times R$. Then let

$$p'(r, s) = \begin{cases} 0 & \text{if } r = s \\ p(r, s) + q_{rs} & \text{otherwise} \end{cases}$$

where for all $r \in R$, $\sum_{s \neq r} q_{rs} = p(r, r)$.

Lemma 8. $c(p') \leq c(p)$ □

Now assume that $c_1 \leq c_2 \leq \ldots \leq c_n$ and consider two distributions over $\binom{R}{2}$:

$$p^1(r,s) = \begin{cases} 1 & \text{if } r = c_1 \wedge s = c_2 \\ 0 & \text{otherwise} \end{cases}$$

and

$$p^2(r,s) = \frac{\alpha}{c_r c_s}$$

where $\alpha = \left(\sum_{\{r,s\}\in\binom{R}{2}} 1/(c_r c_s)\right)^{-1}$. By Lemma 8 $c(p^1) \leq c(p_R^1)$ and $c(p^2) \leq c(p_R^2)$.

Now let $\mathcal{I}_n = (c_1, \ldots, c_n, k)$ be a problem instance that, as n grows, satisfies

1. $c_1 = O(1/n)$.

2. $c_2 > c$ for some constant $c \in (0,1)$.

3. $k = o(n)$

4. $k = \omega(1)$

For large enough n, $\mathcal{I}_n$ has an optimal value that is arbitrarily smaller than the values achieved by p^1 and p^2. Let $c(\mathcal{I}_n, p)$ be the value of $\mathcal{I}_n$ under distribution p.

Theorem 13.

$$\begin{aligned} c(\mathcal{I}_n, p^1)/c(\mathcal{I}_n, p^*) &= \Omega\left(\frac{n}{k}\right) && (7.1) \\ c(\mathcal{I}_n, p^2)/c(\mathcal{I}_n, p^*) &= \Omega(k) && (7.2) \end{aligned}$$

Proof. The following distribution achieves the ratios in Eqs. 7.1 and 7.2. Let

$$p^3(r,s) = \begin{cases} \frac{\alpha}{c_r c_s} & \text{if } r = r_1 \\ 0 & \text{otherwise} \end{cases}$$

where $\alpha = \left(\sum_{i>1} 1/(c_1 c_i)\right)^{-1}$. This distribution puts weight on all distinct pairs that include r_1. It represents a middle approach between putting all the probability on the lightest pair, as p^1 does, and spreading the probability over all pairs, as p^2 does. The optimal distribution for each $\mathcal{I}_n$ only has higher ratios with p^1 and p^2 than p^3 does.

The ratio between p^1 and p^3 is

$$\begin{aligned}
\frac{c(\mathcal{I}_n, p^1)}{c(\mathcal{I}_n, p^3)} &= \frac{c_1 c_2}{(k-1)/\left(\sum_{i=2}^n 1/(c_1 c_i)\right)} \\
&\geq (1 + c_2(n-2)/c_n)/(k-1) \\
&= \Omega\left(\frac{n}{k}\right).
\end{aligned}$$

The ratio between p^2 and p^3 is

$$\begin{aligned}
\frac{c(\mathcal{I}_n, p^2)}{c(\mathcal{I}_n, p^3)} &= \frac{\binom{k}{2}\left(\sum_{i \neq j} 1/(c_i c_j)\right)^{-1}}{(k-1)/\left(\sum_{i=2}^n 1/(c_1 c_i)\right)} && (7.3) \\
&= \frac{k}{2}\left(1 + c_1 \frac{\sum_{2 \leq i < j \leq n} 1/(c_i c_j)}{\sum_{i=2}^n 1/c_i}\right)^{-1} && (7.4) \\
&\geq \frac{k}{2}\left(1 + \frac{c_1}{2}\left(\sum_{i=2}^n 1/c_i - 1\right)\right)^{-1} && (7.5) \\
&= \Omega(k). && (7.6)
\end{aligned}$$

In Eq. 7.5, $\sum_{i=2}^n 1/c_i$ is bounded by n because $c_i > c$, $i > 1$. The last line then follows because $c_1 = O(1/n)$. □

Intuitively, the reason p^1 does arbitrarily worse than p^3 is that it doesn't take advantage of an adversary of size $o(n)$ by putting probability on $\Omega(n)$ pairs, while p^2 does arbitrarily worse than p^3 because it puts probability on pairs $\{r_i, r_j\}$, $i, j > 1$, that have $\Omega(n)$ times higher probability of being successfully compromised than pairs including r_1.

7.4 When pairing off, trust is everything

Allowing arbitrary trust values may be unnecessarily general. Users are unlikely to have precise knowledge of the probability of compromise for each onion router in the network. Instead, they seem more likely to have a few classes of trust into which they can partition the routers, or to have detailed knowledge about only a small number of routers. This fact may help us deal with the apparent computational intractability of the general problem. Also, the potentially complicated optima that result from arbitrary trust values may not satisfy other criteria for path-selection strategies that our problem formulation does not include. For example, we may want the number of possible optimal strategies to be small so users share their behavior with many others, or we may want the strategies to be robust to small changes in trust values.

Therefore, we now consider the case that there are only two trust values. We refer to the nodes with higher trust as the *trusted* set, and nodes with lower trust as the *untrusted* set. This case is simple yet results in non-obvious conclusions, and also still provides practical advice to users.

In Section 7.5 we show that, when there are only two trust values, there are three strategies that are potentially optimal. But first we give here a lemma that allows us to consider only distributions that treat the routers within a trust set identically. Note that this lemma holds for general trust values.

Lemma 9. *Let U be a set of routers with identical trust values c, where $|U| = m$. Let V be the rest of the routers, where $|V| = n$. Then the set of routers is $R = U \cup V$. There exists an optimal distribution p in which the following hold:*

1. *For all $\{u, v\}, \{w, x\} \in \binom{U}{2}$, $p(u, v) = p(w, x)$.*
2. *For all $v \in V$, $u, w \in U$, $p(v, u) = p(v, w)$.*

Proof. Consider some distribution over pairs $p : \binom{R}{2} \to [0, 1]$, $\sum_{\{r,s\} \in \binom{R}{2}} p(r, s) = 1$. Consider any subset $S \subseteq V$. Let X_S be a subset chosen randomly from all subsets X of size k such that $X \cap V = S$. Let $j = k - |S|$ be the size of $X_S \cap U$. Let $c(p, K)$ be the probability of compromise under p, given that set K is chosen by the adversary. That is,

$$c(p, K) = \sum_{\{r,s\} \in \binom{K}{2}} p(r, s) c_r c_s$$

We can calculate the expected probability of compromise of X_S as follows:

$\mathbb{E}\left[c(p, X_S)\right]$

$$= \left\{ \binom{m}{j}^{-1} \sum_{T \subseteq \binom{U}{j}} \left[\begin{array}{l} \sum_{\{t,u\} \in \binom{T}{2}} p(t,u) c^2 + \\ \sum_{u \in T, v \in S} p(u,v) c \cdot c_v + \\ \sum_{\{v,w\} \in \binom{S}{2}} p(v,w) c_v c_w \end{array} \right] \right. \tag{7.7}$$

$$= \left\{ \begin{array}{l} \binom{m}{j}^{-1} \binom{m-2}{j-2} c^2 \sum_{\{t,u\} \in \binom{U}{2}} p(t,u) + \\ \binom{m}{j}^{-1} \binom{m-1}{j-1} c \sum_{v \in S, u \in U} p(v,u) c_v + \\ \sum_{\{v,w\} \in \binom{S}{2}} p(v,w) c_v c_w \end{array} \right. \tag{7.8}$$

$$= \left\{ \begin{array}{l} \frac{j(j-1)c^2}{m(m-1)} \sum_{\{t,u\} \in \binom{U}{2}} p(t,u) + \\ \frac{j \cdot c}{m} \sum_{v \in S, u \in U} p(v,u) c_v + \\ \sum_{\{v,w\} \in \binom{S}{2}} p(v,w) c_v c_w \end{array} \right. \tag{7.9}$$

There must be some set $T \subseteq U$ of size j such that $c(p, S \cup T)$ is at least the expectation expressed in Eq. 7.9. If we modify p to treat all nodes in U the same, and thus satisfy the conditions in the statement of the lemma, every such T achieves the value in Eq. 7.9. Let p' be this modified distribution:

$$p'(r,s) = \begin{cases} \sum_{\{t,u\} \in \binom{U}{2}} p(t,u) / \binom{m}{2} & \text{if } \{r,s\} \in \binom{U}{2} \\ \sum_{u \in U} p(r,u)/m & \text{if } r \in V, s \in U \\ \sum_{u \in U} p(s,u)/m & \text{if } r \in U, s \in V \\ p(r,s) & \text{if } \{r,s\} \in \binom{V}{2} \end{cases}$$

The probability of compromise for any value $S \cup T$ of X_S is

$$c(p', S \cup T) = \binom{j}{2}\binom{m}{2}^{-1} \sum_{\{t,u\} \in \binom{U}{2}} p(t,u)c^2 + \frac{j}{m} \sum_{v \in S} \sum_{u \in U} p(v,u)c_v c + \sum_{\{v,w\} \in \binom{S}{2}} p(v,w)c_v c_w. \tag{7.10}$$

Equations 7.9 and 7.10 are equal, and therefore $\max_{T:|T|=j} c(p', S \cup T) \leq \max_{T:|T|=j} c(p, S \cup T)$. Because this holds for all $S \subseteq V$, $\max_{K:|K|=k} c(p', K) \leq \max_{K:|K|=k} c(p, K)$. □

7.5 Choosing pairs to avoid compromise

Now we analyze optimal distributions for selecting pairs when there are two trust values in the network, c_1 and c_2, with $c_1 \leq c_2$. We show that, in this case, one of the following strategies is always optimal: (*i*) choose a pair of trusted routers uniformly at random, (*ii*) choose pairs such that $p(r,s)c_r c_s$ is equal for all $\{r,s\} \in \binom{R}{2}$, or (*iii*) choose only fully-trusted or fully-untrusted pairs such that the adversary has no advantage in attacking either trusted or untrusted routers. Distribution (*i*), corresponds to distribution p^2, described in Section 7.3.2, with the difference that (*i*) spreads probability to all the most-trusted routers and not just two. Distribution (*ii*) corresponds to distribution p^1 of Section 7.3.2. Distribution (*iii*) shows that non-obvious distributions can exist even when the trust values are very restricted.

Let U be the trusted set, with trust value c_1, $|U| = m$. Let V be the untrusted set, with trust value c_2, $|V| = n$.

Theorem 14. *Let $v_0 = \max(k - m, 0)$ and $v_1 = \max(k - n, 0)$. Then let $g_0 = \frac{v_0(v_0-1)}{n(n-1)}$ and $g_1 = \frac{v_1(v_1-1)}{m(m-1)}$. One of the following is an optimal distribution:*

$$p(r,s) = \begin{cases} \frac{(c_2)^2}{\binom{m}{2}(c_2)^2+(mn)(c_1c_2)+\binom{n}{2}(c_1)^2} \\ \qquad\qquad \textit{if } \{r,s\} \in \binom{U}{2} \\ \frac{(c_1c_2)}{\binom{m}{2}(c_2)^2+(mn)(c_1c_2)+\binom{n}{2}(c_1)^2} \\ \quad \textit{if } (r,s) \in U \times V \cup V \times U \\ \frac{(c_1)^2}{\binom{m}{2}(c_2)^2+(mn)(c_1c_2)+\binom{n}{2}(c_1)^2} \\ \qquad\qquad \textit{if } \{r,s\} \in \binom{V}{2} \end{cases} \tag{7.11}$$

$$p(r,s) = \begin{cases} \binom{m}{2}^{-1} & \textit{if } \{r,s\} \in \binom{U}{2} \\ 0 & \textit{otherwise} \end{cases} \tag{7.12}$$

$$p(r,s) = \begin{cases} \binom{m}{2}^{-1} \frac{c_2^2(1-g_0)}{c_1^2(1-g_1)+c_2^2(1-g_0)} \\ \qquad\qquad \textit{if } \{r,s\} \in \binom{U}{2} \\ \binom{n}{2}^{-1} \frac{c_1^2(1-g_1)}{c_1^2(1-g_1)+c_2^2(1-g_0)} \\ \qquad\qquad \textit{if } \{r,s\} \in \binom{V}{2} \\ 0 \\ \textit{if } (r,s) \in U \times V \cup V \times U \end{cases} \tag{7.13}$$

Proof. Let p be some distribution on $\binom{R}{2}$. By Lemma 9, we can assume that $p(t,u) = p(x,y)$, if $t,u,x,y \in U$. Similarly, $p(v,w) = p(x,y)$, if $v,w,x,y \in V$. Again using Lemma 9, $p(u,v) = p(u,y) = p(x,y)$, if $u,x \in U$ and $v,y \in V$. This shows that all pairs intersecting both U and V have equal probability.

If $k >= n+m$, the adversary can try to compromise all routers. Thus the best strategy is to only choose pairs from the trusted set U, as described in Eq. 7.12. From now on, assume that $k < n+m$.

Let $K_j \subseteq R$ be of size k and have an intersection with U of size j. The value of j alone determines the probability of compromise for K_j, because it determines the number of pairs in $\binom{U}{2}$, $U \times V$, and $\binom{V}{2}$. As we have just shown, the exact pairs included do not matter because their probability is determined by their class. Let $p_1 = \sum_{\{t,u\}\in\binom{U}{2}} p(t,u)$, $p_2 = \sum_{(u,v)\in U\times V} p(u,v)$, and $p_3 = \sum_{\{v,w\}\in\binom{V}{2}} p(v,w)$. Then we can say that

$$\begin{aligned} c(p,K_j) = &\binom{j}{2}(c_1)^2\frac{p_1}{\binom{m}{2}} + j(k-j)c_1c_2\frac{p_2}{mn} + \\ &\binom{k-j}{2}(c_2)^2\frac{p_3}{\binom{n}{2}} \end{aligned} \tag{7.14}$$

To narrow the set of possible optimal assignments of p_1, p_2, and p_3, we will first consider the effect of varying p_2. The quantity we want to minimize is the maximum value of Eq. 7.14. Equation 7.14 is a quadratic function of j. Assume that the second derivative is non-zero. If it is zero it is easy to show that the distribution p is the distribution described in Eq. 7.11. Otherwise, we will show that we can improve the maximum by changing p_2. We can find the local extremum by taking the derivative of Eq. 7.14 and setting it to zero. Solving for j gives

$$j^* = \frac{\begin{gathered} n(n-1)p_1c_1^2 - k(m-1)(n-1)p_2c_1c_2 + \\ (2k-1)m(m-1)p_3c_2^2 \end{gathered}}{\begin{gathered} 2(n(n-1)p_1c_1^2 - (m-1)(n-1)p_2c_1c_2 + \\ m(m-1)p_3c_2^2) \end{gathered}}. \tag{7.15}$$

Unfortunately, j^* must be integral to represent a worst-case subset, and therefore we cannot just substitute the expression in Eq. 7.15 into Eq. 7.14 and solve for the optimal value of p_2. There may in fact be two values of j that are maxima, and varying p_2 could possibly increase the value at one while decreasing the value at other. Therefore, while varying p_2, we simultaneously vary p_1 and p_3 to maintain the local extremum of Eq. 7.14 at j^*. Then both possible maxima are changed in the same way.

By observing that $p_3 = 1 - p_1 - p_2$ in Eq. 7.15 we can see that p_1 and p_2 are linearly related. Solve this for p_1 and call the expression p_1'. Now let $j' \in \mathbb{N}$, $0 \le j' \le k$, be any value that maximizes $c(p, K_{j'})$. j' is either an endpoint of $[0, k]$ or a closest integer to a local maximum. Substitute p_1' for p_1 in $c(p, K_{j'})$, and the result is a linear function of p_2. Therefore either increasing or decreasing p_2 does not increase $c(p, K_{j'})$. Suppose we move p_2 in the direction that decreases $c(p, K_{j'})$. Because we vary p_1' (and p_3) with p_2 in such a way as to maintain the extremum of the parabola at the same value j^*, j' is maintained as a maximum of $c(p, K_j)$ as long as the second derivative of $c(p, K_{j'})$ remains non-zero.

The process of changing p_2 stops when (*i*) the second derivative of $c(p, K_{j'})$ becomes zero, (*ii*) p_2 reaches zero, (*iii*) p_3 reaches zero, or (*iv*) p_1 reaches zero.

Case (i): In this case, all sets have the same value. This is only satisfied when the distribution is that of

Eq. 7.11.

Case (ii): In this case, all probability is in pairs of two trusted or two untrusted nodes. Therefore the maximizing value of j must be when it is as small as possible or as large as possible, i.e., at $\max(0, k-n)$ or $\max(k, m)$. If the former case is strictly larger, we can reduce it by decreasing p_3 and increasing p_1. If the latter case is strictly larger, we can do the reverse. Therefore the value in these two cases must be equal. To find the probabilities p_1 and p_3 that satisfy this, let $p_3 = 1 - p_1$, $v_0 = \max(k-m, 0)$, and $v_1 = \max(k-n, 0)$. Then setting them equal and solving for p_1 yields the condition

$$p_1 = \frac{c_2^2\left(1 - \frac{v_0(v_0-1)}{n(n-1)}\right)}{c_1^2\left(1 - \frac{v_1(v_1-1)}{m(m-1)}\right) + c_2^2\left(1 - \frac{v_0(v_0-1)}{n(n-1)}\right)}. \tag{7.16}$$

Equation 7.16 then gives us the probability for each pair in $\binom{U}{2}$ and $\binom{V}{2}$, and this is the same as the distribution in Eq. 7.13.

Case (iii): In this case, $p_3 = 0$. Then if $p_2 = 0$ also, we put all probability in the trusted nodes, which is the distribution described in Eq. 7.12.

Now suppose that $p_2 > 0$. We will consider moving probability between p_1, p_2, and p_3 to show that this case isn't possible. Let $p_2 = 1 - p_1$ in Eq. 7.14 and call this $c_3(p, K_j)$. Then use this to consider trading off p_1 and p_2 to find the optimal assignment. As p_1 varies, the change in the value of the set K_j is

$$D_{p_1} c_3(p, K_j) = \frac{jc_1}{m}\left[\frac{(j-1)c_1}{m-1} - \frac{(k-j)c_2}{n}\right]. \tag{7.17}$$

Next, let $p_2 = 1 - p_1 - p_3$ in Eq. 7.14 and call this $c_4(p, K_j)$. Moving p_2 to p_3 results in a change of

$$D_{p_3} c_4(p, K_j) = \frac{(k-j)c_2}{n}\left[\frac{(k-j-1)c_2}{n-1} - \frac{jc_1}{m}\right]. \tag{7.18}$$

Let $j^* \in \text{argmax}_j c(p, K_j)$ be the largest integer that is a maximum of $c(p, K_j)$.

We observe that $D_j^2 c(p, K_j) \leq 0$. If not, we would have $j^* = k$. Then Eq. 7.17 shows that decreasing p_1 would decrease the value at j^*, and p_1 is non-zero so we could do this because, at $p_1 = 0$, $c(p, K_j)$ is largest at $j^* = \lceil k/2 \rceil \neq k$. Such a decrease would contradict the optimality of j^*.

Now, because $D_j^2 c(p, K_j) \leq 0$, there may be some $\hat{j} \in \text{argmax}_j c(p, K_j)$ such that $\hat{j} < j^*$. There are four cases to consider here: (1) $D_{p_1} c_3(p, K_{j^*})$, $D_{p_1} c_3(p, K_{\hat{j}}) \leq 0$, (2) $D_{p_1} c_3(p, K_{j^*})$, $D_{p_1} c_3(p, K_{\hat{j}}) \geq 0$, (3) $D_{p_1} c_3(p, K_{j^*}) \geq 0$ and $D_{p_1} c_3(p, K_{\hat{j}}) \leq 0$, and (4) $D_{p_1} c_3(p, K_{j^*}) \leq 0$ and $D_{p_1} c_3(p, K_{\hat{j}}) \geq 0$.

In case (1), we could decrease c at j^* and $\hat{j}$ by moving probability from p_2 to p_1. This would contradict the optimality of p.

For case (2), we use the fact that

$$0 \leq a < b \Rightarrow \frac{a-1}{b-1} < \frac{a}{b}. \tag{7.19}$$

Inequality 7.19 implies that if $D_{p_1}c_3(p, K_j) \geq 0$, then $D_{p_3}c_4(p, K_j) \leq 0$. Therefore we could decrease c at j^* and $\hat{j}$ by moving probability from p_2 to p_3, contradicting the optimality of p.

For case (3), we show that we can still decrease both j^* and $\hat{j}$ while maintaining their equality, and hence maximality, by moving some probability from p_2 to p_1 and p_3. Moving probability from p_2 to p_1 increases the value at j^* and decreases the value at $\hat{j}$. This implies, by Inequality 7.19, that moving probability from p_2 to p_3 decreases the value at j^*. Furthermore, can assume that it increases it at $\hat{j}$ because otherwise we could decrease both j^* and $\hat{j}$ by moving probability directly from p_2 to p_3.

For j^* and $\hat{j}$ to be integral maxima of Eq. 7.14, it must be that $j^* - 1 = \hat{j}$. Also, solving $D_{p_1}c_3 = D_{p_3}c_4$ for j, we find that at this point, $D_{p_1}c_3 \leq 0$ and $D_{p_3}c_4 \leq 0$. Therefore, j^* is at most one more than this point. We can observe by calculation that within this range the ratio $|D_{p_1}c_3/D_{p_3}c_4|$ is less than one. Similarly, $\hat{j}$ is at most one less than this point, and within this range the ratio $|D_{p_1}c_3/D_{p_3}c_4|$ is greater than one.

This shows that we can move probability from p_2 to p_1 and p_3 at rates that decrease the value at both j^* and $\hat{j}$. Because they were maximum, we have lowered the value of the worst-case subset K_{j^*}, contradicting the optimality of p.

Case (4) is not possible because $D_j^2[D_{p_1}c_2] \geq 0$ and $D_{p_1}c_3(p, K_0) = 0$.

Case (iv): In this case, if $p_2 > 0$, the case is symmetric to the case of $p_1, p_2 > 0$ and we can apply the same argument. Therefore assume that $p_2 = 0$, which implies that $p_3 = 1$. It must be that $m < n$ because otherwise we could set $p_1 = 1$ and $p_3 = 0$ and improve the worst case. But now consider moving some probability from p_3 to p_1. Let $p_1 = 1 - p_3$ in Eq. 7.14 and call this c_3. The change in the worst-case case subset, K_n, is

$$D_{p_3}c_3(p, K_n) = c_2^2 - \frac{(k-n)(k-n-1)c_1^2}{m(m-1)}.$$

This must be greater than zero because $c_2 \geq c_1$ and $k - n < m$. Therefore decreasing p_3 decreases $c(p, K_n)$, contradicting the optimality of p. □

7.6 Choosing a distribution

We have shown that there are three possibilities for an optimal strategy in choosing nodes that will minimize the best chances a fixed size adversary has to compromise both endpoints of an onion-routing circuit when a trusted set is available. To choose a distribution, a user can simply calculate the probability of compromise in each case and use the distribution with the smallest result. The optimal distribution depends on all the variables in the system: the trust values, the size of the trusted set, the size of the untrusted set, and the size of the adversary.

In the first distribution, described in Eq. 7.11, the user chooses pairs $\{i, j\}$ to make $p(i, j)c_i c_j$ equal for all i, j. This is a random choice of pairs weighted by the trust in the pair. The probability of compromise under this strategy is

$$C_1 = \frac{k(k-1)c_1^2 c_2^2}{m(m-1)c_2^2 + 2mnc_1c_2 + n(n-1)c_1^2}. \tag{7.20}$$

This strategy is optimal when the network is large compared to the adversary, and so it benefits the user to spread out his distribution, even to less-trusted routers. It is also optimal when the trust values are close.

In the second distribution, described in Eq 7.12, the user randomly selects pairs from within the trusted set. This can only be optimal if the size k of the adversary is larger than the size m of the trusted set. Otherwise, the user could decrease the probability of compromise by putting some of the pair-selection distribution on pairs outside the trusted set. Doing so would not change the adversary's worst-case subset, which is entirely in the trusted set, but it would decrease the probability that those nodes are chose by the user. The probability of compromise, assuming $k > m$, is simply

$$C_2 = c_1^2. \tag{7.21}$$

We can compare this to Eq. 7.20 and observe that c_1 can always be made small enough to make this value less than the value of the first strategy. These equations also show that choosing only trusted nodes will be optimal when k is large relative to the network. When $k = m + n$, this case is always optimal.

The third distribution, given in Eq. 7.13, is perhaps the least obvious one, and arises as a result of the fact that users choose their distribution over pairs, while the adversary attacks individual routers. Let $v_0 = \max(k-m, 0)$ and $v_1 = \max(k-n, 0)$. Then let $g_0 = v_0(v_0-1)/(n(n-1))$ and $g_1 = v_1(v_1-1)/(m(m-1))$.

In general, the probability of compromise under this distribution is

$$C_3 = \begin{cases} \frac{c_1^2c_2^2(1-g_0)}{c_1^2(1-g_1)+c_2^2(1-g_0)} + \\ \frac{v_0(v_0-1)c_1^2c_2^2(1-g_1)}{n(n-1)(c_1^2(1-g_1)+c_2^2(1-g_0))} \end{cases} \tag{7.22}$$

$$= \begin{cases} \frac{v_1(v_1-1)c_1^2c_2^2(1-g_0)}{m(m-1)(c_1^2(1-g_1)+c_2^2(1-g_0))} + \\ \frac{c_1^2c_2^2(1-g_1)}{c_1^2(1-g_1)+c_2^2(1-g_0)} \end{cases} \tag{7.23}$$

To make some sense of this, it is helpful to consider some special cases. When $n > k, m < k$, the probability of compromise is

$$C_3 = \frac{k(k-1)c_1^2c_2^2}{n(n-1)(c_1^2 + c_2^2(1-g_0))}$$

We can see that there is some large m such that C_3 is less than C_2 and C_1. What happens in this case is that there are large number of routers, and the user wants to spread his probability among them. However, because $k > n$, spreading the probability to all cross-pairs (one trusted and one untrusted router) means that an adversary selecting as many untrusted routers as possible gains $(k-n)n/(mn) = (k-n)/m$ of the probability on such pairs. On the other hand, when spreading to trusted pairs $(k-n)(k-n-1)/(m(m-1))$ of the shifted probability is captured by the adversary. The latter shrinks quadratically with m while the former shrinks only linearly. At some point it will be beneficial to spread probability to trusted pairs but not to cross-pairs. The case when $m > k, n < k$ is similar. This distribution is never optimal when $m > k$ and $n > k$, because the worst-case sets are contained within U and V, and so spreading probability to the cross-pairs some small amount will always decrease the probability of compromise.

Chapter 8

A Protocol for Highly-Anonymous Low-Latency Communication

8.1 Introduction

Anonymous communication protocols are designed primarily to allow *users* to communicate with *destinations* anonymously. They face, however, the challenge of optimizing over several competing criteria: anonymity, latency, and bandwidth. Mix networks such as Mixminion [13] trade off latency for good anonymity and low message overhead. Protocols such as Dining Cryptographers [10] and Pipenet [11] can provide low latency and good anonymity at the cost of sending a large number of extra messages. Protocols such as onion routing [29] and Crowds [56] do not add much latency or message overhead but provide weak anonymity. High latency and reduced bandwidth are unacceptable for many popular Internet applications, and onion routing, despite its weak anonymity, has become a successful protocol for anonymous communication on the Internet. To design a useful protocol, we focus on designing a protocol that provides better anonymity than onion routing while maintaining desirable latency and bandwidth.

Low-latency protocols in general have been vulnerable to several attacks based on the timing of events in the system. Typically, the user in these protocols chooses a set of *routers* to mediate between the user and the destination, forwarding data between the two and obscuring their relationship. The essential problem is that timing patterns in these data are conserved between the source and destination. Therefore an adversary only needs to observe the incoming *stream* of data (the consecutive messages exchanged during a communication session), from the user and the outgoing stream of data to the destination to use patterns to link the two. In a *passive* timing attack, an adversary relies on timing patterns that are generated by the user. Because the user creates these patterns, he can prevent this attack by adding dummy packets and delays into the stream to make his traffic look similar to the traffic of other users [65]. However, the adversary can defeat this by performing an *active* attack, in which he inserts timing patterns into the traffic as it passes through routers under his control.

As a result of this sort of active attack, existing low-latency anonymity protocols do not provide anonymity when the adversary controls the routers that the user communicates with directly and the routers that the destination communicates with directly. Suppose the adversary controls a fraction b of the network. In onion routing, users select routers uniformly at random, and the adversary therefore compromises anonymity with probability b^2. In Crowds, the adversary need only control the first router [60], which then occurs with probability b.

These probabilities are fixed and cannot be improved by trading off performance elsewhere, and they can be quite insufficient. Consider Tor [18], the popular open-source implementation of onion routing. The dedicated Tor network consists, as of March 2009, of around 1500 routers [69] provided by volunteers. An adversary that runs 30 routers, just 2% of the network, has over a 50% chance of compromising 1 in 1750 *circuits* (the persistent connections through the network that users set up to route traffic). McCoy et al. [44],observed 7571 unique clients while running a router for one day. At this usage rate, our adversary, with 30 routers, would expect to compromise over 3 users per day. Also, currently in Tor users choose new circuits every 10 minutes. Therefore, any given user has a 50% chance of being compromised at least once in about 2 weeks of continuous use. Moreover, as observed by Bauer et al. [3], Tor weights router selection by bandwidth to provide good performance. We can expect any system of heterogeneous resources to do something similar for satisfactory performance. Thus an adversary need only control 2% of the bandwidth to achieve these success rates. McCoy et al. observed in [44] that the top 2% of routers transported about 50% of the traffic. In such a situation, supplying 2% of the bandwidth in Tor can be achieved by supplying just the two routers with the highest bandwidth. Thus, the system provides almost no anonymity against a wide variety of realistic opponents, such as governments, ISPs, and criminals willing to purchase the use of botnets.

We consider the very weak anonymity provided by low-latency protocols against an active adversary to be a fundamental and critical challenge in anonymous communication. In this chapter, we present a low-latency protocol that provides arbitrarily good anonymity against an adversary that can observe and create timing patterns. The protocol makes black-box use of a padding scheme to prevent passive timing attacks. Several padding schemes that defeat passive timing attacks have been proposed [65, 63, 71], and furthermore we believe that there is still potential for substantial improvement. The protocol provides two-way stream communication.

The key ingredients of our solution are timestamping packets with their intended send time, sending multiple copies of the same packet over redundant paths, and padding a stream inside the network. These ideas have appeared before in the literature (cf. [37, 19] for timing techniques, [66, 35] for redundancy, and [63] for in-stream padding). The essential aspects of the protocol include:

1. A mesh topology from the user to the destination that balances limiting view of the stream to a small number of routers while providing redundancy against malicious delays

2. A predefined padding scheme from the destination to the user

3. A method to select free routes in the mesh topology

We evaluate the anonymity provided by our protocol in a network model that incorporates timing and an active adversary. We take this approach for three reasons. First, we wish to make security claims against all adversaries that have certain general capabilities, and therefore the only way to fully demonstrate that the claims hold is to prove them. Second, we wish to make precise what adversarial capabilities our scheme protects against. Third, the open problem we investigated had a wide space of possible outcomes. Could we achieve arbitrarily good security with reasonable performance? Was using free routes possible once you leave the simple path topology? As a result, our aim was to show good *asymptotic* peformance in our design criteria. Our scheme may be suboptimal in several ways, and, had we undertaken certain tests, the results for our specific design choices may have appeared quite poor. The theoretical results, however, suggest that the approach is indeed viable and that a promising next step is to optimize within the framework of the scheme we describe.

However, because we are concerned with eventual practicality, we do measure a component of the system over which we have no control and which could have made our protocal unusable - delay variations in the host network. Specifically, we measured the likely latency costs of running our protocol on the existing Tor [18] network. The Tor network consists of routers of widely different capacities located around the world and therefore is likely to exhibit a high variability in performance. This provides a strenuous, real-world scenario in which to evaluate our protocol's performance. We examine this by taking measurements of the inter-host network latencies and the per-host processing delays of the Tor routers over the course of a month.

Our results are therefore a mix of the theoretical and experimental:

1. We show that the user can be made anonymous with arbitrarily high probability as long as b is less than $1/2$. The user is anonymous within the set of users with identical traffic patterns as produced by the input padding scheme.

2. We prove that our mesh topology in the forward direction is optimal for anonymity in a limit sense.

3. The latency of our protocol is proportional to the length of the mesh and path. We show that the probability of compromise decreases to zero polynomially in that length. This compares well with onion routing, which adds latency proportional to its path length.

4. The bandwidth used is $2w + (l-1)w^2 + 1$, where l is the mesh/path length, and $w = \Theta(\log l)$. This compares well asymptotically to the $l + 2$ copies that are sent in an onion-routing path of the same total length.

5. For most of our measurements, we observe that added packet delay would need to be less than a factor of two to achieve desirable reliability.

The results suggest that our approach indeed has the potential to mitigate active timing attacks. However, the case is not fully made. A truly suitable padding scheme, with low overhead and large *anonymity sets* (groups of users within which the adversary cannot distinguish), does not currently exist. There still exists lots of design space within which to vary the construction of our scheme, such as changing the mesh/path length or using persistant circuits and private keys. Moreover, most of the delays measured appear to be caused by congestion at the router. Tor was designed for optimal throughput rather than predictable and reliable latency, and therefore we expect that a usable network could be built with even better performance than our measurements suggest.

8.2 Model

We will express and analyze our anonymity protocol in a model of network and adversary behavior. A particular advantage of this approach is the ability to make convincing guarantees of security when we cannot predict the tactics that an adversary will use.

8.2.1 Network

Let the network consist of a set of onion routers R, a user population U, and a set of destinations D. The network is completely connected, in that every host can send a message directly to every other host. Each event in the network occurs at some global time. We assume that each user and router has a local clock that accurately measures time, and that these clocks have run some sort of synchronization protocol [46]. Let δ_{sync} be the largest difference between two synchronized clocks in the network.

There is some probabilistic network delay $d_{net}(r, s)$ between every pair (r, s) of routers. This models the unpredictable delay between routers due to factors such as congestion and route instability. There is also some probabilistic processing delay $d_{proc}(r)$ at every router r. This reflects changes in delay at a router due to time sharing with other processes. The delay of a message is the sum of delays drawn independently from these two distributions. We also let the delay of one message be independent of the delays of the other messages. We assume the distributions of both sources of delay is known to the system. In practice, this may be achievable via network measurements.

We assume that all hosts (in particular, all destinations) respond to a simple connection protocol. One host h_1 begins the connection by sending to another host h_2 the pair $< n, M >$, where $n \in N^+$ is a number and M is a message. Any responses M' from h_2 are sent back as $< n, M' >$. Once established connections in the anonymity protocol cannot be closed by anyone to prevent distinctions of one from another by open and close times. All connections thus stay open for a fixed amount of time and then close automatically. Application connections running over these can of course be closed by the ultimate source and destination; although this will not close the anonymity circuit, and padding messages will continue to be sent until connection timeout.

8.2.2 Users

User communication drives the operation of the anonymity network. We view the communication of a user as a sequence of connections. Each connection is to one destination, and it includes messages to and from the destination.

8.2.3 Adversary

The adversary controls some subset $A \subseteq R$ of the routers, where $b = |A|/|R|$. It seems plausible that an adversary can run routers that are at least as fast as the other routers on the network, and that it may dedicate them to running the protocol. Therefore, in contrast to non-adversarial routers, we pessimistically assume that the adversary has complete control over the processing delay $d_{proc}(a)$ of routers $a \in A$.

8.2.4 Padding scheme

The padding scheme $\mathcal{P}$ is a black box that initially takes as input a connection start time and outputs the timing of the return traffic. Then every time step it takes the presence of data from the user and returns whether or not a packet should be sent. Note that this requires the scheme to determine in advance the length of the connection. Basic constant-rate padding clearly satisfies these requirements, although it typically causes high added latency and/or message overhead. The padding scheme of Shmatikov and Wang [63] could be adapted to these requirements by fixing the return scheme (perhaps by sampling in advance from the delay distribution). It is not hard to conceive of novel padding schemes that might satisfy these requirements, although getting a good mix of anonymity and performance does not seem easy.

Let S_u be the set of users that start connections at the same time as user u and have the same traffic pattern. Our proposed protocol relies on the effectiveness of the padding scheme. At best, it makes u indistinguishable within the set S_u supplied by $\mathcal{P}$.

8.3 Problem

The problem in this model is to design an anonymity protocol that supports the low-latency, two-way, stream communication that has made Tor [18] popular. We understand stream communication to mean that users communicate in sessions of arbitrary length and volume, which makes embedding timing signatures possible. In order to allow communication with hosts that are ignorant of the protocol, we require that only one host communicates with the final destination, and that the communication is only the original messages generated by the user.

We evaluate our protocol by three criteria: anonymity, latency, and the amount of data transferred.

Anonymity can be a subtle concept to define [53]. First, in an anonymity network there are several actions that one may wish to perform anonymously. We evaluate our protocol according to its *relationship anonymity*, that is, the extent to which it prevents an adversary from determining which user-destination pairs are communicating. Second, there are several possible metrics [59, 16, 70] to measure the amount of anonymity that the network provides. We simply use the probability that the adversary assigns to the correct destination of a user connection.

To be useful, it is important for an anonymity protocol to have low latency. Therefore we consider the amount of time it takes for a message to reach the destination from a user. It is also important to keep the bandwidth requirements of the protocol low, and so we consider the total amount of data that has to be transferred during a user connection.

We are interested in both providing good asymptotic behavior of the protocol and potentially usable performance estimates. Asymptotic behavior is interesting theoretically, is a likely prerequisite for usable performance, and provides a good starting point from which to attempt to optimize the protocol for real-world application.

8.4 A Time-stamping Solution

The padding scheme gives us sets of users that have traffic streams with identical timing patterns. However, the model we have described gives the adversary the ability to modify these patterns as the traffic travels through its routers towards the destination. To prevent this, we try to enforce the desired timing pattern on packets sent by including the times that the routers should forward them. Any honest node that receives the packet will obey the instructions, removing any delays inserted by the adversary. For traffic sent from the user to the destination we can trust the user to correctly encode the padding-scheme times. Traffic sent from the destination to the user must by our requirements pass initially through a single router. Because it may be compromised, we cannot trust it to use a proper padding scheme. However, this traffic is destined for an anonymity protocol participant, the user; therefore, unlike traffic from the user, we can destroy inserted timing patterns by re-padding it all the way to the user. Observe that re-padding does not work for traffic from the user, because the final router sees which packets are real and which are padding.

8.4.1 From the user

First, consider what we could do if propagation and processing delays were deterministic. The user could send through a path in the network a layered data structure that we call an *onion* that, for each packet, includes in the ith layer the time that the onion should arrive at the ith router. Then each router on the path could unwrap the onion to make sure that the initial timing sequence was being preserved and, if so, forward the onion.

Unfortunately, in real networks, delays are somewhat unpredictable. For example, an onion might be delayed by congestion in the underlying network. However, if the distribution of delays is known, we know how long we need to wait at a router for onions to arrive with any fixed probability. We will set that probability to balance decreasing added latency with decreasing the chance of a successful timing attack. Then we add in this buffer time to the send time.

Another problem is that the adversary could drop onions entirely in a pattern the propagates down the path. Our approach to this problem is to send multiple copies of an onion down redundant, intersecting paths. A router on a path needs only one copy of the onion to arrive in time from any incoming path in order to forward it by its send time.

This approach has limits, because each redundant router adds another chance for the adversary to observe an entire path from source to destination. For example, suppose that we simply send onions over k paths of length l that intersect at a final router, where every router is chosen uniformly at random. Let b be the fraction of routers that are controlled by the adversary. The probability that at least one path is entirely composed of compromised routers is $b(1-(1-b^{l-1})^k)$. This quickly goes to b as k increases. We use a layered-mesh topology to balance the ability of the adversary to passively observe a path with his ability to actively perform a timing attack.

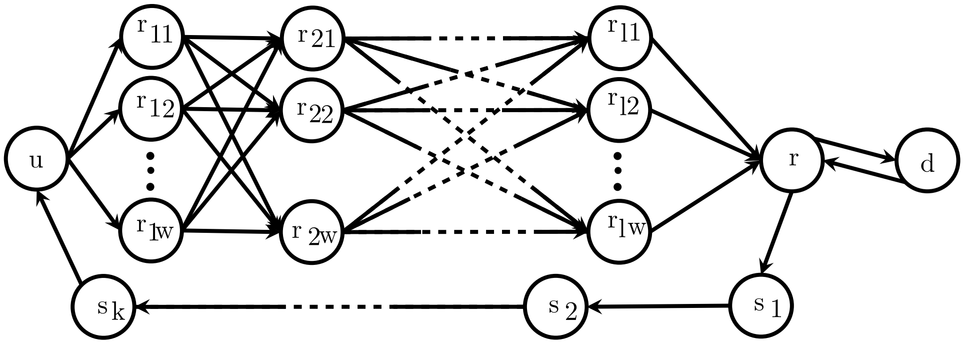

Figure 8.1: Layered-mesh topology

Topology

The layered-mesh topology we propose is pictured in Figure 8.1. For some length l and width w, user u sends a copy of each onion to the w members r_{1j} of the first layer. Then, in layer i, each router r_{ij} sends one copy of every onion it receives to each router $r_{(i+1)k}$ of the next layer. Finally, the routers r_{lj} in last layer send a copy of each onion received to a single router r, that finishes decrypting them and sends the data on to the destination. We call this structure the *layered mesh*.

Timestamps

As described, the user sets timestamps to instruct routers to maintain a specific timing pattern. A user may have different latencies to the different routers in the first layer. If the time that one of these routers is instructed to forward the packet only depended on the network and processing delays of that router, the first-layer routers could send copies of the same packet at different times. This provides information to the next layer about the identity of the user. Similarly, the adversary could use these different times to link other layers in the mesh. Therefore, we set the send time for each layer to be the send time of the previous layer plus the longest delay at our chosen reliability level p. We must also add some extra delay to allow for clock skews.

Let $d^*(r, s)$ be the amount of delay we need to ensure that a packet from r is processed at s with success probability p:

$$p = Pr[d_{net}(r, s) + d_{proc}(s) \leq d^*(r, s)].$$

At time t, let user u be instructed by $\mathcal{P}$ to send a message. The user chooses the same send time for all routers in the same layer. The send time for routers r_{1j} in the first layer is

$$t_1 = t + \max_j d^*(u, r_{1j}).$$

The send time for routers r_{ij} in the ith layer is

$$t_i = t_{i-1} + \max_{j,k} d^*(r_{(i-1)j}, r_{i,k}) + \delta_{sync}.$$

If a router receives its first copy of an onion after the send time has passed, it immediately forwards the onion to the routers in the next layer. At worst, the delay is the result of attempts by the adversary to delay certain packets. Sending the packet later or not at all in that case would only make it easier for the adversary to observe its lateness later in the mesh. Forwarding it immediately might even allow the onion to eventually get back on schedule. At best, the delay is just a result of network delays and forwarding has no effect on anonymity.

Onions

Let M be the message to be sent. We will encrypt the message with a public key shared by all members of a layer. Given that the layers are set up in advance and known to all, such a key can be generated by a trusted third party or by electing a leader to do it. Let $\{M\}_{r_i}$ denote the encryption of M with the public key of layer i. Let $n_{r_i}, n_{s_1} \in \mathbb{N}$ be random integers. Then the onion that u sends to the routers in layer 1 is

$$\{n_{r_1}, t_1, \{n_{r_2}, t_2, \cdots \{n_r, d, n_{s_1}, k_r, M\}_r \cdots\}_{r_2}\}_{r_1}$$

For each layer i, a user generates the random number $n_{r_i} \in \mathbb{N}$ as an onion identifier. The routers keep track of the onion identifiers they have seen. When they receive an onion, they decrypt it and examine the identifier. Routers only forward an onion if its identifier n_i has not been seen before. n_{s_1} is the identifier that r should use with s_1 when sending back any reply, and k_r is a private key that will let r encrypt the return message for u.

The onion encoding and forwarding scheme should hide routing information, prevent forgery, prohibit replay attacks, and hide message content. For clarity of presentation, we have described a simple scheme that achieves this. We observe, however, that several improvements to the protocol could be made. For example, the protocol could do a more explicit stream open and close to reduce the lists of identifiers that routers have to maintain. Also, symmetric keys could be exchanged to speed up onion processing. Another improvement that we could incorporate is forward secrecy. There are numerous cryptographic details that must be carefully set out (*e.g.* as in [8]) for our protocol to have a cryptographically secure and efficient implementation. These are not our focus here.

8.4.2 To the user

Traffic returning to the user from the destination must first pass through the one router that is selected to communicate directly with the destination. This router may be compromised, and it may try to insert timing patterns in the return traffic. We manage this by giving the intermediate routers the pattern of the return traffic. They enforce it by fitting the return onions into the pattern, adding dummy packets when necessary. We note again that this doesn't work for the traffic from the user because any added delays translate into delays in the underlying data, and this can be viewed by the final router. We choose a simple path of length k for the return traffic (Fig. 8.1), because there is no anonymity advantage to adding redundancy here. We call this structure the *return path*.

To communicate the desired traffic pattern to the return path, we take advantage of the one-way communication protocol already developed. The user takes the return traffic pattern that is given by the padding scheme $\mathcal{P}$ and sends it via the layered mesh to every router in the return path. At the same time, the user generates the random numbers $n_{s_i} \in \mathbb{N}^+, 1 \leq i < k$, and sends two random numbers $n_{s_i}, n_{s_{i+1}}$ and a key k_{s_i} to each router s_i. These will be the incoming and outgoing identifiers for the onion. The user also sends n_k and u to s_k. Let M be the message to be sent back from d to u. The return onion sent to s_i is

$$O_i =< n_{s_i}, \{\cdots\{\{M\}_{k_r}\}_{k_{s_1}}\cdots\}_{k_{s_{i-1}}} > .$$

After r receives M from d, it will take n_{s_1} and k_r out of the inbound onion from the user and send O_1 to s_1. When s_i receives O_i it looks for a matching n_{s_i} in the pairs of number it has received and then forms O_{i+1} to send when the padding scheme instructs it to.

8.4.3 Choosing the routes

A simple method to select the layered mesh and return path would be for it to be chosen by the network and shared among all users. This is analogous to cascades in mix networks [4]. A disadvantage of cascades is that number of users that can be confused with one another, *i.e.*, the size of the anonymity set, is at most the number of users that can simultaneously be handled by a router.

Therefore, we allow users to select their own routes, known as free routes. This can provide a much bigger anonymity set. A relevant comparison of the approaches in the context of mix networks is given in [19]. This procedure must be designed carefully because users are no longer guaranteed that anybody else chooses the same routers they do. We let the set of nodes be divided into groups of size w that form the layers described above. These layers are further stratified by the position they can serve in a mesh, *e.g.*, no second-layer group could serve in any other position in a route chosen by anyone else.

When the user first selects a layer that is completely uncompromised, his stream becomes indistinguishable from the other users that select that layer. His stream can also be confused with routes that have consecutive uncompromised layers covering the same mesh position because the adversary cannot tell what was selected in that position. This continues until the end of the mesh or until the user selects a completely compromised layer (after which the adversary can drop packets to create patterns that are detectable at the end). As the user population grows, the expected number of users sharing a completely uncompromised layer reaches the capacity of those routers. Therefore the expected size of the anonymity set grows with n and also depends on b.

Though outside of the threat model we consider in this paper, one can consider an attacker that attempts to compromise specific nodes over an extended period in an attempt to increase the chances of owning an entire layer and an entire path from a client to that layer. Against such an attacker one would want to have a dynamic network structure in which the positions that layers can be in change periodically as does the composition of each layer group. These may or may not change at the same rate. There are numerous tradeoffs to be made in deciding how best to do this. It is beyond the scope of this paper to do more than note the existence of such responses to a long-term strategically roaming attacker.

8.5 Anonymity

The purpose of our protocol is to provide better anonymity than onion routing at reasonable cost in latency and bandwidth. A major drawback to onion routing is that the probability of compromise is b^2 and cannot be improved, *e.g.* by choosing a longer path. We will show how in fact our scheme can provide arbitrarily good probability by increasing the length and width of the layered mesh.

First, the design of the protocol onions essentially limits the adversary to traffic analysis. For traffic from the user, the use of encryption and unique identifiers forces onions to be passed through the required layers and limits them to being forwarded by an honest router at most once. It also hides the source and destination from all but the first and last routers, and it makes messages leaving one layer unlinkable to messages leaving another layer. For traffic to the user, the source and destination are not included in the packets, and encryption prevents messages leaving one router from being linked with messages leaving another router.

For the adversary to break a user's anonymity, then, he will need to either observe traffic on an entire path between source to destination or link traffic at different steps on that path. The latter depends on the adversary's ability to introduce delays in the packet stream. To evaluate this possibility, we will make the simplifying assumption that the network and processing delay between two routers r, s never falls above the time allowed for it $d^*(r, s)$. In our model, such failures can happen with probability $1 - p$, where p can be set arbitrarily close to 1. Certainly, if the adversary can deanonymize a user even under this assumption, he can do so with possible link failures. When such failures do occur, they open up the chance for an adversary to successfully delay packets and insert a timing pattern. However, late packets are typically irrelevant because the same packet will have been forwarded by another router in the previous layer. Also, late packets that are the first of their type to arrive are immediately forwarded, thus benign delays are somewhat self-correcting, and we estimate that they do not open up a large opportunity for an active timing attack. However, if we wish to make a conservative estimation, we can expect that for any given a fraction p of the packet copies will fail to arrive in time. We can simply add p to the fraction of routers controlled by the adversary to obtain an effective fraction $b^* = b + p$ of compromised routers that can be used to determine the probability that the packet is successfully delayed or dropped from the stream. Of course, p changes with every packet, but for connections with few packets from the user, such as web connections, this approximation may still yield good anonymity guarantees.

Assuming no link failures, then, the anonymity of a user only depends on which routers the adversary controls. Because all users use the same routers, the adversary can either deanonymize all users in the anonymity set S_u, or he can not deanonymize any of them. Because the routers in the topology are selected randomly, there is some probability that the adversary can deanonymize the cascade users. Let $\mathcal{C}$ be the event that the adversary can compromise anonymity. We can quantify the probability of $\mathcal{C}$.

Theorem 15.

$$Pr[\mathcal{C}] = b^{k+1} + b(1-b^k)\cdot \left[b^w \frac{1-(1-(1-b)^w - b^w)^l}{b^w + (1-b)^w} + (1-(1-b)^w - b^w)^l\right] \tag{8.1}$$

Proof. To compromise anonymity, the adversary must control the router r that communicates with the destination. Therefore assume that $r \in A$. This happens with probability b.

Then he can link the destination via the return path only when he controls every router on it. This happens with probability b^{k+1}.

The adversary can deanonymize the user via the mesh in two ways. First, the adversary can control an entire path from the first layer in the mesh to r, but not any entire layer. This happens with probability $(1-(1-b)^w - b^w)^l$. Second, if the adversary controls a path to a completely controlled layer i, he can then use it to selectively delay packets from one user. He can identify the user to target because he controls a path from that user to layer i. This happens with probability $b^w(1-(1-b)^w - b^w)^{i-1}$. If neither of these is the case, then it must be that there is an honest path in the mesh leading to a completely honest layer. This layer receives all packets from all users via the honest path, and beyond it the adversary cannot distinguish between users. Therefore deanonymization is not possible.

Adding all of the probabilities, we get

$$Pr[\mathcal{C}] = b^{k+1} + b(1-b^k)\cdot \left[b^w \sum_{i=0}^{l-1} (1-(1-b)^w - b^w)^i + (1-(1-b)^w - b^w)^l \right]$$

Simplifying the partial geometric sum gives the theorem. □

Theorem 15 shows that, when less than half of the routers are compromised, we can make the probability that a user is anonymous arbitrarily high by setting w, l, and k large enough.

Corollary 4.

$$\lim_{w,l,k\to\infty} Pr[\mathcal{C}] = \begin{cases} 0 & b \le 1/2 \\ 1/4 & b = 1/2 \\ b & b > 1/2 \end{cases}$$

Proof. As $k \to \infty$, the first term in Eq. 15 goes to zero. As $k, l \to \infty$, the second term goes to $b^{w+1}/(b^w + (1-b)^w)$. When $b < 1/2$, this goes to zero, when $b = 1/2$, this goes to $1/4$, and when $b > 1/2$, this goes to b. □

Corollary 4 shows that anonymity can be made arbitrarily good when $b < 1/2$, but that it is worse than onion routing when $b \ge 1/2$. Therefore assume from now on that $b < 1/2$. We would like to determine how big the layered mesh and return path grow as we increase our desired level of anonymity. First, we will consider how wide the mesh must be for a given depth to achieve optimal anonymity. This affects the number of messages that need to be sent. Also, it will allow us to evaluate anonymity as a function of the lengths k and l, quantities which we would like to keep small to provide good latency. Luckily, it turns out that the optimal width w^* grows slowly as a function of l.

Theorem 16. $w^* = O(\log l)$

Proof. Let $\mathcal{A} = \mathcal{C}^c$ be the event that users are anonymous. From Thm. 15, we can determine that

$$Pr[\mathcal{A}] = 1 - b + b(1-b)^w \frac{1-(1-b^w-(1-b)^w)^l}{b^w + (1-b)^w}. \tag{8.2}$$

Let x be the third term in Eq. 8.2. It is the only term that varies with w. Because $b < 1/2$, we can replace b with $1-b$ at point in x to obtain a simpler upper bound:

$$x \leq b(1-(1-(1-b)^w)^l). \tag{8.3}$$

Similarly, we can replace $1-b$ with b to obtain a lower bound:

$$x \geq b(1-(1-(1-b)^w)^l)/2. \tag{8.4}$$

Equation 8.3 is decreasing in w. Equation 8.4 is increasing for small w, achieves a maximum at $w = \log(1/2)/\log p$, and is decreasing for larger w. At its maximum, Eq. 8.4 has value $b/2$. Therefore $Pr[A]$ must be maximized at a smaller w than that at which the upper bound in Eq. 8.3 reaches $b/2$. We can further approximate to find this w:

$$\begin{aligned} x &\leq b(1-(1-(1-b)^w)^l) && (8.5)\\ &\leq bl(1-b)^w && (8.6) \end{aligned}$$

Expression 8.6 is $b/2$ when w is $\frac{\log(2l/b)}{\log(1/(1-b))}$. □

Network users are particularly sensitive to latency, and each additional step in the layered mesh and return path represents a potentially global hop on our network. To keep the lengths l and k of the mesh and return path small, then, we would like for $Pr[\mathcal{C}]$ to converge quickly to its limit. As suggested by Thm. 16, let $w = \log l$, and consider the convergence of $Pr[\mathcal{C}]$. It is clear that the first term shrinks exponentially with k. The convergence time of the second term is polynomial, although it does improve as b gets smaller.

Theorem 17. *Let* $c_1 = \log b$ *and* $c_2 = \log(1-b)$. *Then*

$$Pr[\mathcal{C}] = \Theta(l^{c_1-c_2}).$$

Proof. Let us fix k and only consider how $Pr[\mathcal{C}]$ changes with increasing l. Let x be the part of Eq. 15 that varies with l and substitute in $w = \log l$:

$$x = bl^{c_1}\frac{1-(1-l^{c_1}-l^{c_2})^l}{l^{c_1}+l^{c_2}} + b(1-l^{c-1}-l^{c_2})^l.$$

The first term in x is less than l^{-c_2}. We can give an upper bound for the second term by using the inequality $1 - y \leq e^{-y}$:

$$\begin{aligned} b(1 - l^{c-1} - l^{c_2})^l &\leq b(1 - l^{c_2})^l \\ &\leq be^{-l^{1+c_2}} \end{aligned}$$

$1 + c_2 > 0$ because $b < 1/2$. Therefore $x \leq b(l^{c_1 - c_2}) + e^{-l^{1+c_2}} = O(l^{c_1 - c_2})$.

We can give a lower bound for the first term in x by using the fact that $1 - y \geq e^{-2y}$ for $y \leq 1/2$:

$$\begin{aligned} bl^{c_1} \frac{1 - (1 - l^{c_1} - l^{c_2})^l}{l^{c_1} + l^{c_2}} &\geq bl^{c_1} \frac{1 - e^{-2(l^{c_1} + l^{c_2})l}}{2l^{c_2}} \\ &\geq \frac{b}{2}(1 - e^{-2l^{1+c_2}})l^{c_1 - c_2} \end{aligned}$$

Therefore $x = \Omega(l^{c_1 - c_2})$. □

Our analysis shows how our scheme can provide arbitrarily good probability for $b < 1/2$. Is it possible to improve this to include values of b greater than $1/2$? Also, the convergence as the topology grows is not as fast as we could hope. The return path converges exponentially to perfect anonymity, but the layered mesh does not, and so far we have not justified the specific choice of a mesh. Is it possible that we could design a topology that has improved convergence time?

First, we observe that some other plausible topologies do not perform as well. For example, suppose the user sends onions to k routers, each of which forwards it to the same next router, and then from then on there is a path to the destination. The probability that anonymity is compromised in this situation is $b(b(1 - (1 - b)^k + (1 - b)b^k))$. While for small values of k and b this is a small improvement over the simple path in onion routing, as k grows the anonymity goes to zero.

As another example, consider using a binary tree. The user sends to the leaves of the tree, each child forwards the onion to its parent, and the parent forwards one copy of the onions it receives. Set p so that the probability of a sent packet arriving late is arbitrarily small. There are three events at a router r that are relevant to packet loss propagating up the tree:

$\mathcal{C}_r$	r is compromised.
$\mathcal{D}_r$	Packets from all users can be stopped from coming through r.
$\mathcal{E}_r$	Packets from a known user can be stopped from coming through r.

Let $c_1(r)$ and $c_2(r)$ be the children of r. These events at r can be described in terms of in the same events at its children.

$$\begin{aligned} \mathcal{D}_r &= \mathcal{C}_r \vee (\mathcal{D}_{c_1(r)} \wedge \mathcal{D}_{c_2(r)}) \\ \mathcal{E}_r &= \mathcal{C}_r \wedge (\mathcal{E}_{c_1(r)} \vee \mathcal{E}_{c_2(r)}) \vee \\ & \quad (\mathcal{E}_{c_1(r)} \wedge \mathcal{D}_{c_2(r)}) \vee (\mathcal{D}_{c_1(r)} \wedge \mathcal{E}_{c_2(r)}) \end{aligned}$$

We use these descriptions to give recurrence relations for the probabilities of the events. Let r be at distance i from a leaf. Let D_i be the probability of $\mathcal{D}_r$.

$$\begin{aligned} D_0 &= b \\ D_i &= b + (1-b)D_{i-1}^2 \end{aligned}$$

Let E_i be the probability of $\mathcal{E}_r$.

$$\begin{aligned} E_0 &= b \\ E_i &= bE_{i-1}(2 - E_{i-1}) + (1-b)E_{i-1}(2D_{i-1} - E_{i-1}) \\ &= E_{i-1}(2b + 2(1-b)D_{i-1} - E_{i-1}) \end{aligned}$$

The probability that the adversary can block all packets going through a router is always at least b, and increasing its height in the tree increases the probability up to a limit.

Theorem 18.

$$\lim_{i \to \infty} D_i = \begin{cases} \frac{b}{1-b} & b \leq \frac{1}{2} \\ 1 & b \geq \frac{1}{2} \end{cases}$$

Proof. Let D^* be the fixpoint of D_i.

$$\begin{aligned} D^* &= b + (1-b)(D^*)^2 \\ D^* &= \frac{1 - |1-2b|}{2(1-b)} \end{aligned}$$

It is clear from the definition that D_i is strictly increasing for $b > 0$. Therefore D^* is achieved in the limit. □

We can use this result to determine the limit of E_i.

Theorem 19.

$$\lim_{i \to \infty} E_i = \begin{cases} 0 & b \leq \frac{1}{4} \\ 4b-1 & \frac{1}{4} \leq b \leq \frac{1}{2} \\ 1 & b \geq \frac{1}{2} \end{cases}$$

Proof. Replace D_i with D^* in the definition of E_i, and call it $E_i^{'}$. We can compute the fixpoints for $E_i^{'}$:

$$E^{'*} = E^{'*}(2b + 2(1-b)D^* - E^{'*})$$

$$E^{'*} = 2b - |1-2b| \vee 0$$

We will treat these as the fixpoints for E_i, although more careful arguments need to be made to show that we can treat D_i as its limit in this way.

Observe that when $b \leq 1/4$, E_i is strictly decreasing.

$$2b + 2(1-b)D_i \leq 2b + 2(1-b)D^* \leq 1$$

Therefore, in the limit, E_i reaches the higher fixpoint for $E^{'*}$, which in this case is $E^* = 0$.

Now consider the case that $1/4 \leq b \leq 1/2$. When $E_i > 2b + 2(1-b)D^* - 1$, E_i decreases with i. When $E_i < 2b + 2(1-b)D_{i-1} - 1$, E_i is increasing. Careful limiting arguments show that as $D_i \to D^*$, E_i also approaches a limit. Because E_i increases when it is small, this limit must be the fixpoint $2b - |1-2b| = 4b-1$.

Finally, let $b \geq 1/2$. E_i is multiplied each step by $2b + 2(1-b)F_i - E_i$, which, in this case, approaches $2 - E_i$. Therefore, for large i, E_i is increasing, unless E_i is very close to 1. However, even then E_i cannot decrease too much. Choosing a convergent sequence of F_i yields a sequence of E_i that approaches 1. □

For adversary to link a user with his destination, the final router r_f must be compromised. Also, a signal must propagate either up the send tree or down the return path. The probability of at least one of these compromises occuring is at most $b(E_h + 1 - b^l)$. Therefore, by Theorem 19, we can push it arbitrarily low when $b \leq 1/4$ by increasing h and l.

As Theorem 19 shows, anonymity in the binary tree is not as good as it is in the layered mesh. The following theorem shows that the layered mesh is optimal in the limit among all topologies.

Theorem 20. *Let $c(b)$ be the probability of anonymity compromise when the fraction of adversarial routers is b. Then, if $b < 1/2$, $c(b) < \epsilon$ implies that $c(1-b) > 1 - b - \frac{1-b}{b}\epsilon$.*

Proof. It is easy to show that we can assume the following: the topology is acyclic, there are two users, u and v, and the adversary's strategy is to forward only v's packet when he is sure it belongs to v and otherwise to block all packets.

Assume that the router that communicates with the destination is adversarial. Then, for a given topology, let α be a Boolean function on the remaining router positions that indicates if that position is controlled by the adversary or not. If anonymity is not compromised under α, there must be a set of honest routers C such that C forms a cut between the users and destination and every router in C sends packets from u and v. To see this consider the induced subgraph of routers that send the packet from v but not from u. It cannot connect u and the last router or anonymity would be compromised. Therefore, there must be a set of routers that are outgoing neighbors to this subgraph. These routers and the honest first-hop routers constitute the desired cut.

Every router in C must have a path of honest routers from the user. Otherwise there would be a set of adversarial routers that cut a router in C from the users, and no packets from the users would reach that router. Now consider the assignment $\overline{\alpha} = 1 - \alpha$. If α provides anonymity, then in $\overline{\alpha}$ there must be a cut consisting of adversarial routers such that each router has a path of adversarial routers from the users. Thus anonymity is compromised in $\overline{\alpha}$.

Let $c'(b)$ be the probability of anonymity compromise given that the final router is compromised. The probability of assignment α with a fraction b of compromised routers is equal to the probability of assignment $\overline{\alpha}$ with a fraction $1 - b$ of compromised routers. Because the complement of an assignment providing anonymity is an assignment the compromises anonymity, we have

$$1 - c'(b) \leq c'(1-b). \tag{8.7}$$

$c(b) = bc'(b)$, and, therefore, if $c(b) < \epsilon$, then $c'(b) < \epsilon/b$. Combining this with Inequality 8.7, we get that $c'(1-b) > 1-\epsilon/b$, which implies that $c(1-b) > 1-b-\frac{1-b}{b}\epsilon$.. □

Theorem 20 implies that if the probability of compromise for a topology goes to zero for $b \in [0, \beta]$, then it must go to b for $b \in [1-\beta, 1]$. This is achieved by the layered-mesh topology for the largest range of b possible, $[0, 1/2)$.

8.6 Latency Measurements

Our protocol requires routers to hold messages until a specified send time. Latency between routers varies, and therefore this send time must be set high enough to guarantee that with sufficiently high probability that it occurs after the packet actually arrives. The amount of time that packets spend being delayed before the send time depends on the variability of latency. Because network performance is critical to realizing effective anonymous communication [17], and we wish to evaluate the latency in our protocol.

In order to do so, we have measured latency in the Tor [18] network. Performance data in the Tor network gives us reasonable estimates for the performance of our protocol for several reasons. First, the essential router operation in both protocols is decrypting and forwarding packets. Second, the network is globally distributed and therefore includes a wide variety of network and host conditions. Third, the volunteer-run network is the most successful example of anonymous communication in practice, and it seems likely that any future system for anonymous communication would include such a component.

We can therefore estimate the added delays in our protocol by simply using the network-delay and processing-delay distributions we observe in the Tor network. In addition, we are able to examine the data more closely and understand the underlying factors affecting these distributions, such as Internet route congestion, bandwidth at Tor routers, and process load at Tor routers.

8.6.1 Methodology

During each experiment, we measured the latency of Tor routers in three ways:

1. For a given router, we first opened five TCP connections in a row and measured the time between the TCP SYN request and the SYNACK response. This provides an estimate of the network delay. We refer to this delay as the *round-trip time* (RTT).

2. We then created a circuit from our test host to the router and opened a TCP stream over that circuit from the router back to the test host. We measured the time between sending the stream-open request

to the router and receiving the TCP connection from the router. This time includes host-processing delay from decrypting the packet and responding to the command. We refer to this delay as the *connection delay.*

3. Finally, we sent five 400-byte segments of data up the established TCP stream. The segments were sent every 500ms in separate Tor cells (Tor's uniform-size packets). We measured the time between sending a given segment and receiving all 400 bytes. This time includes host processing delay from decrypting the packet and forwarding the data. We refer to this delay as the *packet delay.*

We took measurements in the Tor network from February 22, 2009, to March 21, 2009. Every hour, we downloaded the latest router directories and used them to identify routers to test during that hour. Following the Tor-specification instruction for choosing routers, we only measured routers with the modifiers "Exit", "Fast", "Running", "Stable", and "Valid". The routers also had to be non-hibernating and could not have exit policies that disallowed connections to our host IP address and port. For every router we tested during the hour, we started an experiment after an exponentially-distributed random delay. After each experiment finished, we started another one after another exponentially-distributed random delay. There was an average number of five experiments per router per hour.

The server port we used was 80. The Tor client was a custom client written in Java. Packet traces were recorded using `tcpdump`. The timestamps on these were used to determine the timing of events during the measurement process.

8.6.2 Results

Over the course of the month, 745 routers appeared that met our selection criteria. The average number of experiments per router was 399, the median was 89, the minimum was 1, and the maximum was 1776. These numbers reflect the uptime of Tor routers.

We estimate network delay by looking at the round-trip time of a TCP SYN/SYNACK pair. Round-trip times for all experiments are shown in Figure 8.2. The mean of these times is 161ms and the median is 110ms. Similar to the connection delays, we see a peak bin centered at 100ms. These data are consistent with the measurements of Mukherjee [48], which, are shown to be well-modeled by a gamma distribution. In particular, these delays are unimodal.

A histogram of all the connection delays measured is shown in Figure 8.3. The top 5% of delays have been removed to show the rest in greater detail. It shows that nearly all delays are less than 1s. Also, we

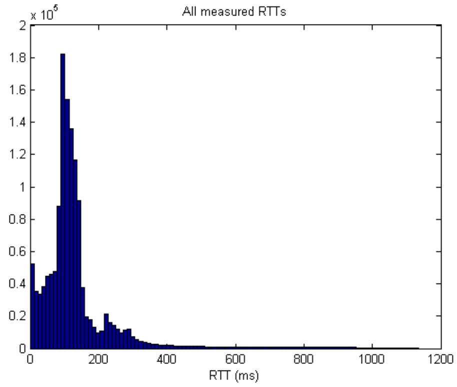

Figure 8.2: Round-trip times (top 1% removed)

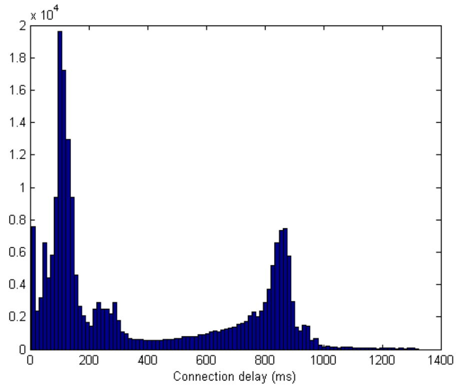

Figure 8.3: Connection delays (top 5% removed)

can see that the distribution is bimodal, with peaks at about 120ms and 860ms.

This pattern is present when considering the connection delays per router. We group the delays up to 1s into bins of size 20ms. Then we examine the top three most-populated bins for evidence of bimodality. We find 100 routers for which there exist, among these three bins, both one centered below 300ms and one centered above 700ms. The connection delays over time for a one such router - *b0b3r* (193.221.122.229) - is shown in Figure 8.4. The clear timing stratification suggests a cause other than varying Internet-route congestion or host-resource contention. Furthermore, the RTTs are unimodal, and so the cause is likely to be in the Tor routers. We believe that this is due to read/write rate-limiting that Tor servers manage by periodically filling global token buckets.

We can combine the RTT and connection delay measurements to estimate the distribution of host processing delays. For each router, we estimate the RTT and connection-delay distributions by partitioning delay times from 0 to 1000ms into 50 bins and treating the normalized frequency counts as probabilities.

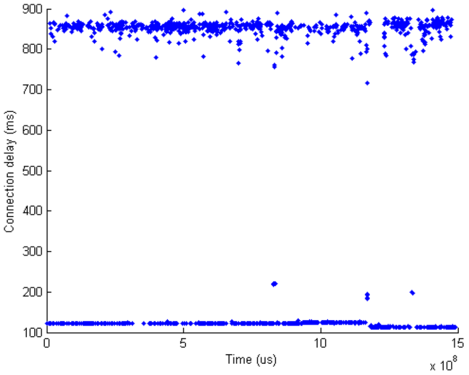

Figure 8.4: Connection delays for router *b0b3r*

The connection delay is the sum of the RTT and the processing delay, and therefore, if we assume independence of the RTT and processing-delay distributions, we can derive the processing delay distribution for each router. Considering the distribution over all routers, there is almost a 40% probability of having a processing delay of nearly zero. Thus processing delays are not due to the inherent cost of computation but to limited resources at the routers.

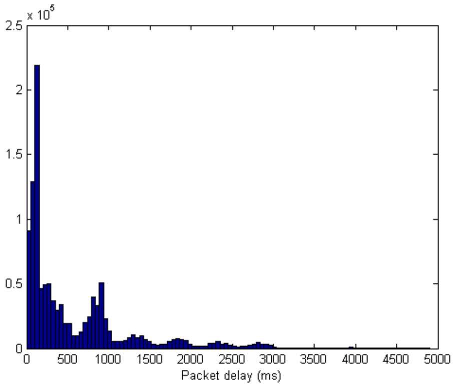

Figure 8.5: Packet delays (top 1% removed)

The delays of all 400-byte packets that were successfully forwarded is shown in Figure 8.5. We again see times that cluster around certain levels. Here, there are six obvious levels, approximately centered at 125ms, 860ms, 1350ms, 1840ms, 2330ms, and 2820ms. To examine any such clustering per router, we group the packet delays for each router by 20ms intervals from 0-5000ms and look at the largest six such groups. There are 136 routers with two groups of these six within 100ms of the observed cluster peaks. There are 14 routers with three such groups. If we examine the delay time series of individual routers we see that again, like with connection delays, the different levels are interleaved. Thus phenomenon is undoubtedly due to the same mechanism underlying the similar pattern in connection delays.

From the delay measurements we can estimate the added delay that would have resulted from our protocol. The protocol chooses a time t that must elapse from the time the packet is originally sent before the packet is forwarded. It is set such that with some success probability p the packet arrives at the forwarding router in less than t time. In our data, we examine the tradeoff that varying t sets up between the fraction p of packets that arrive in time and the forwarding delay that gets added to them.

We divide the delay measurements by router and into periods of 6 hours. This period length is long enough to have 30 an expected experiments per router but is still short to adjust for the changes in average delay that occurs in many routers over time. Within each period, to achieve success probability p we set the send time t to be the smallest value that is larger than at least a fraction p of delays. For those delays smaller than t, using the send time t adds some delay. We look at the relative increase, that is, the total new delay divided by the original delay.

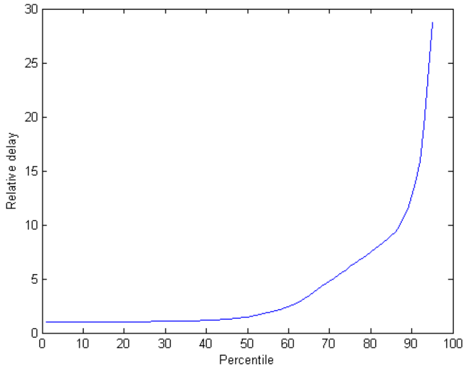

Figure 8.6: Relative connection delays at p=0.95

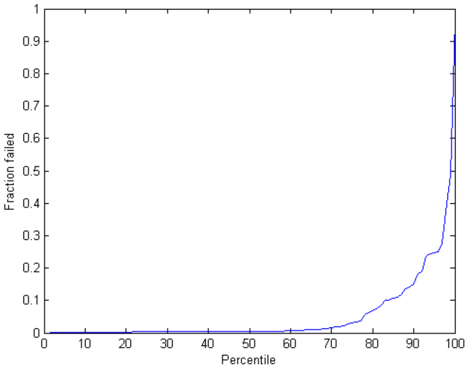

Figure 8.7: Stream-open failures

The distribution over all successfully-opened streams of relative connection delays to achieve a success probability of 0.95 is shown in Figure 8.6. At the 50th percentile, the relative connection delay is less than 1.48. We also must consider the failure rate of opened streams. We could simply consider failures to be connection with infinite delay, but failures often occur for reasons that have nothing to do with congestion. Therefore it is useful to keep them separate. These failures occur after a Tor circuit has been successfully created with the destination router. It thus seems unlikely that most of these failures are due to resource constraints. The distribution of connection failures over all periods is shown in Figure 8.7. At the 50th percentile, the failure rate is less than 0.005.

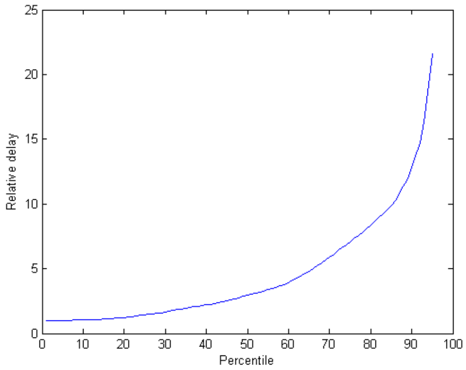

Figure 8.8: Relative packet delays at p=0.95

The data for relative packet delays appears in Figure 8.8. At the 50th percentile, the relative packet delay is just over 2.95. We also considered the distribution over periods of failure rates for forwarding packets, which turned out to be clearly much better than the rates for connection failures. The rate stays below 0.005 until the 99th percentile. The reason for this is that packet sends are not even attempted when the connection fails. It shows that once a router successfully opens a connection, it is quite likely to successfully forward all the data that it receives.

Chapter 9

Conclusion and Future Work

There are several interesting and important questions on trust to examine in future work: What happens when a network is shared between entities that do not share trust levels placed on the nodes? What is the impact of trust on profiling in this case? What is the effect of learning if we add time to the model and allow the adversary to rove rather than conducting a one-off attack?

Though our motivation in our trust model is onion routing, our analysis applies to any network where it would be beneficial to reduce the chance of circuit-endpoint threats by choosing circuits with less vulnerable endpoints. It clearly generalizes to other low-latency anonymity designs, such as Crowds [56]. It also applies beyond networks for anonymity to other concerns. For example, network endpoints may be able to collaborate to cover up checksum or other errors that might flag data-integrity attacks. And, capturing internet traffic for any kind of analysis (cryptanalysis, textual analysis, traffic analysis, etc.) may be easier to do or harder to detect or both if pairs of nodes are collaborating for route capture. Alternatively they might collaborate for unfair resource sharing. Similar observations apply to ad-hoc and peer-to-peer networks and to sensor networks, for which vulnerability of cheap, low-power, and physically accessible nodes is a known concern. Going further, our results are not restricted in applicability to path endpoints. In any setting in which sets of principals can collaborate so that a successfully compromised pair can conduct an attack our results are potentially applicable. Examining larger numbers of nodes being attacked than just pairs is one possible generalization of this work that should apply in many settings.

The protocol described in Chapter 8 would benefit from some developments that have to potential to make it truly useful and effective.

Foremost among these is to design and evaluate a usable padding scheme. We would like it to provide large anonymity sets and low overhead. It should also allow the return padding scheme to be predetermined. It could be possible to weaken the total predictability of return traffic by allowing the user to send updates of the return padding scheme to the return path. One would have to be careful to maintain a large set of other users providing the same updates, however.

Also, we have avoided for now optimizing the efficiency of processing at the routers. Onion routing, in particular, has developed good techniques to make this aspect of the protocol fast. For example, we could use persistant circuits to speed up detecting duplicate packets. We could also switch to private keys to speed up decryption.

Our analysis could be improved in some areas as well. First, we have considered benign link delays to be failures to evaluate their effect on anonymity. However, the protocol calls for such late arrivals to be immediately forwarded. The result of such forwarding seems likely to, at some point, lead to the packet catching up to one of its send times. Understanding this process better could improve the expected anonymity of the protocol. Also, we have observed that most of the delays measured appear to be due to congestion at the router. Tor is not optimized for latency, and therefore we should seek to understand the underlying resource issues in such network to accurately determine the added latencies of our protocol.

Bibliography

[1] Martín Abadi. On SDSIs linked local name spaces. *Journal of Computer Security*, 6(1/2):2–22, 1998.

[2] Adam Back, Ulf Möller, and Anton Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems. In *Proceedings of Information Hiding Workshop (IH 2001)*, pages 245–257, 2001.

[3] Kevin Bauer, Damon McCoy, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Low-resource routing attacks against tor. In Ting Yu, editor, *WPES'07: Proceedings of the Workshop on Privacy in the Electronic Society*, pages 11–20. ACM Press, October 2007.

[4] Oliver Berthold, Andreas Pfitzmann, and Ronny Standtke. The disadvantages of free MIX routes and how to overcome them. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 30–45. Springer-Verlag, LNCS 2009, July 2000.

[5] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In *In Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 164–173. IEEE Computer Society Press, 1996.

[6] Nikita Borisov, George Danezis, Prateek Mittal, and Parisa Tabriz. Denial of service or denial of security? How attacks on reliability can compromise anonymity. In Sabrina De Capitani di Vimercati, Paul Syverson, and David Evans, editors, *CCS'07: Proceedings of the 14th ACM Conference on Computer and Communications Security*, pages 92–102, October 2007.

[7] Philippe Boucher, Adam Shostack, and Ian Goldberg. Freedom systems 2.0 architecture. White paper, Zero Knowledge Systems, Inc., 2000.

[8] Jan Camenisch and Anna Lysyanskaya. A formal treatment of onion routing. In *Proceedings of CRYPTO 2005*, pages 169–187, 2005.

[9] Ajay Chander, John C. Mitchell, and Drew Dean. A state-transition model of trust management and access control. In *Proceedings of the 14th IEEE Computer Security Foundations Workshop, CSFW '01*, pages 27–43. IEEE Computer Society, 2001.

[10] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1:65–75, 1988.

[11] Wei Dai. Pipenet 1.1. Post to Cypherpunks mailing list, November 1998.

[12] George Danezis. Statistical disclosure attacks: Traffic confirmation in open environments. In *Proceedings of Security and Privacy in the Age of Uncertainty, (SEC2003)*, pages 421–426, 2003.

[13] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, pages 2–15, 2003.

[14] George Danezis and Andrei Serjantov. Statistical disclosure or intersection attacks on anonymity systems. In *Proceedings of 6th Information Hiding Workshop (IH 2004)*, pages 293–308, 2004.

[15] Sabrina De Capitani di Vimercati, Sushil Jajodia, Stefano Paraboschi, and Pierangela Samarati. Trust management services in relational databases. In *ASIACCS '07: Proceedings of the 2nd ACM symposium on Information, computer and communications security*, pages 149–160, New York, NY, USA, 2007. ACM.

[16] Claudia Díaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards measuring anonymity. In *Proceedings of Privacy Enhancing Technologies Workshop (PET 2002)*, pages 54–68, 2002.

[17] Roger Dingledine and Nick Mathewson. Anonymity loves company: Usability and the network effect. In Ross Anderson, editor, *Proceedings of the Fifth Workshop on the Economics of Information Security (WEIS 2006)*, Cambridge, UK, June 2006.

[18] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, pages 303–319. USENIX Association, August 2004.

[19] Roger Dingledine, Vitaly Shmatikov, and Paul Syverson. Synchronous batching: From cascades to free routes. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)*, volume 3424 of *LNCS*, pages 186–206, May 2004.

[20] Roger Dingledine and Paul Syverson. Reliable MIX Cascade Networks through Reputation. In Matt Blaze, editor, *Proceedings of Financial Cryptography (FC '02)*. Springer-Verlag, LNCS 2357, March 2002.

[21] Nathan S. Evans, Christian Grothoff, and Roger Dingledine. A practical congestion attack on Tor using long paths, 2009. Manuscript.

[22] Ronald Fagin and Joseph Y. Halpern. I'm OK if you're OK: On the notion of trusting communication. *Journal of Philosophical Logic*, 17(4):329–354, November 1988.

[23] Nick Feamster and Roger Dingledine. Location diversity in anonymity networks. In Sabrina De Capitani di Vimercati and Paul Syverson, editors, *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2004)*, pages 66–76, 2004.

[24] Joan Feigenbaum, Aaron Johnson, and Paul Syverson. A model of onion routing with provable anonymity. In *Proceedings of Financial Cryptography and Data Security (FC 2007)*, pages 57–71, 2007.

[25] Joan Feigenbaum, Aaron Johnson, and Paul Syverson. Probabilistic analysis of onion routing in a black-box model [extended abstract]. In Ting Yu, editor, *WPES'07: Proceedings of the Workshop on Privacy in the Electronic Society*, pages 1–10. ACM Press, October 2007.

[26] Xinwen Fu, Bryan Graham, Riccardo Bettati, and Wei Zhao. Active traffic analysis attacks and countermeasures. In *Proceedings of the 2003 International Conference on Computer Networks and Mobile Computing*, pages 31–39, 2003.

[27] Xinwen Fu, Bryan Graham, Riccardo Bettati, and Wei Zhao. Analytical and empirical analysis of coutermeasures to traffic analysis attacks. In *Proceedings of the 2003 International Conference on Parallel Processing*, pages 483–492, 2003.

[28] Ian Goldberg. On the security of the Tor authentication protocol. In George Danezis and Philippe Golle, editors, *Proceedings of the Sixth Workshop on Privacy Enhancing Technologies (PET 2006)*, pages 316–331, Cambridge, UK, June 2006. Springer.

[29] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Hiding routing information. In *Information Hiding: First International Workshop, Proceedings*, pages 137–150, 1996.

[30] Joseph Y. Halpern and Kevin R. O'Neill. Anonymity and information hiding in multiagent systems. *Journal of Computer Security*, 13(3):483–514, 2005.

[31] Andrew Hintz. Fingerprinting websites using traffic analysis. In Roger Dingledine and Paul Syverson, editors, *Privacy Enhancing Technologies: Second International Workshop, PET 2002*, pages 171–178, San Francisco, CA, USA, April 2002. Springer-Verlag, LNCS 2482.

[32] Nicholas Hopper, Eugene Y. Vasserman, and Eric Chan-Tin. How much anonymity does network latency leak? In Sabrina De Capitani di Vimercati, Paul Syverson, and David Evans, editors, *CCS'07: Proceedings of the 14th ACM Conference on Computer and Communications Security*, pages 82–91. ACM Press, 2007.

[33] Dominic Hughes and Vitaly Shmatikov. Information hiding, anonymity and privacy: A modular approach. *Journal of Computer Security*, 12(1):3–36, 2004.

[34] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography from anonymity. In *FOCS '06: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 239–248, Washington, DC, USA, 2006. IEEE Computer Society.

[35] Jan Iwanik, Marek Klonowski, and Miroslaw Kutylowski. Duo–onions and hydra–onions – failure and adversary resistant onion protocols. In *Proceedings of the IFIP TC-6 TC-11 Conference on Communications and Multimedia Security*, pages 1–15, September 2004.

[36] Dogan Kesdogan, Dakshi Agrawal, and Stefan Penz. Limits of anonymity in open environments. In *Proceedings of Information Hiding Workshop (IH 2002)*, 2002.

[37] Dogan Kesdogan, Jan Egner, and Roland Büschkes. Stop-and-go MIXes: Providing probabilistic anonymity in an open system. In *Proceedings of Information Hiding Workshop (IH 1998)*, 1998.

[38] Brian N. Levine, Michael K. Reiter, Chenxi Wang, and Matthew K. Wright. Timing attacks in low-latency mix-based systems (extended abstract). In *Proceedings of Financial Cryptography (FC '04)*, pages 251–265, 2004.

[39] Marc Liberatore and Brian Neil Levine. Inferring the source of encrypted HTTP connections. In Rebecca N. Wright, Sabrina De Capitani di Vimercati, and Vitaly Shmatikov, editors, *CCS'06: Proceedings of the 13th ACM Conference on Computer and Communications Security*, pages 255–263. ACM Press, 2006.

[40] Patrick Lincoln, Phillip Porras, and Vitaly Shmatikov. Privacy-preserving sharing and correlation of security alerts. In *Proceedings of the 13th USENIX Security Symposium*, pages 239–254, 2004.

[41] Nancy A. Lynch. *Distributed Algorithms.* Morgan Kaufmann Publishers Inc., 1996.

[42] Nick Mathewson and Roger Dingledine. Practical traffic analysis: Extending and resisting statistical disclosure. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)*, pages 17–34, 2004.

[43] Sjouke Mauw, Jan Verschuren, and Erik de Vink. A formalization of anonymity and onion routing. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS 2004)*, pages 109–124, 2004.

[44] Damon McCoy, Kevin Bauer, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Shining light in dark places: Understanding the Tor network. In Nikita Borisov and Ian Goldberg, editors, *Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008)*, pages 63–76, Leuven, Belgium, July 2008. Springer.

[45] Jon McLachlan and Nicholas Hopper. Don't clog the queue: Circuit clogging and mitigation in P2P anonymity schemes. In *Proceedings of Financial Cryptography (FC '08)*, January 2008.

[46] David Mills. Network time protocol (version 3) specification, implementation. RFC 1305, Internet Engineering Task Force, March 1992.

[47] Prateek Mittal and Nikita Borisov. Information leaks in structured peer-to-peer anonymous communication systems. In *Proceedings of the 15th ACM conference on Computer and communications security (CCS 2008)*, pages 267–278, October 2008.

[48] Amarnath Mukherjee. On the dynamics and significance of low frequency components of internet load. *Internetworking: Research and Experience*, 5:163–205, 1992.

[49] Steven J. Murdoch and George Danezis. Low-cost traffic analysis of Tor. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, pages 183–195. IEEE CS, May 2005.

[50] Steven J. Murdoch and Piotr Zieliński. Sampled traffic analysis by internet-exchange-level adversaries. In Nikita Borisov and Philippe Golle, editors, *Proceedings of the Seventh Workshop on Privacy Enhancing Technologies (PET 2007)*, Ottawa, Canada, June 2007. Springer-Verlag, LNCS 4776.

[51] Rafail Ostrovsky and Moti Yung. How to withstand mobile virus attacks. In *Proceedings of the Tenth ACM Symposium on Principles of Distributed Computing (PODC '91)*, pages 51–59. ACM Press, 1991.

[52] Lasse Øverlier and Paul Syverson. Locating hidden servers. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, pages 100–114. IEEE CS, May 2006.

[53] Andreas Pfitzmann and Marit Hansen. Anonymity, unobservability, and pseudonymity: A consolidated proposal for terminology. Draft, July 2000.

[54] Jean-François Raymond. Traffic analysis: Protocols, attacks, design issues, and open problems. In Hannes Federrath, editor, *Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability*, pages 96–114. Springer-Verlag, LNCS 2009, July 2000.

[55] 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.

[56] Michael Reiter and Aviel Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[57] Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618 – 644, 2007.

[58] Steve Schneider and Abraham Sidiropoulos. CSP and anonymity. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS 1996)*, pages 198–218, 1996.

[59] Andrei Serjantov and George Danezis. Towards an information theoretic metric for anonymity. In *Proceedings of Privacy Enhancing Technologies Workshop (PET 2002)*, pages 41–53, 2002.

[60] Vitaly Shmatikov. Probabilistic model checking of an anonymity system. *Journal of Computer Security*, 12(3-4):355–377, 2004.

[61] Vitaly Shmatikov and Carolyn Talcott. Reputation-based trust management. *Journal of Computer Security*, 13(1):167–190, 2005.

[62] Vitaly Shmatikov and Ming-Hsui Wang. Measuring relationship anonymity in mix networks. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2006)*, October 2006.

[63] Vitaly Shmatikov and Ming-Hsui Wang. Timing analysis in low-latency mix networks: Attacks and defenses. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS 2006)*, pages 18–33, 2006.

[64] G. J. Simmons and Catherine Meadows. The role of trust in information integrity protocols. *Journal of Computer Security*, 3(1):71–84, 1994/1995.

[65] Paul Syverson, Gene Tsudik, Michael Reed, and Carl Landwehr. Towards an analysis of onion routing security. In Hannes Federrath, editor, *Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability*, pages 96–114. Springer-Verlag, LNCS 2009, July 2000.

[66] Paul F. Syverson. Onion routing for resistance to traffic analysis. In *Proceedings of the 3rd DARPA Information Survivability Conference and Exposition (DISCEX-III)*, volume 2, pages 108–110, 2003.

[67] Paul F. Syverson and Stuart G. Stubblebine. Group principals and the formalization of anonymity. In *Proceedings of the World Congress on Formal Methods (FM'99), Vol. I*, pages 814–833, 1999.

[68] The Tor project home page. `https://www.torproject.org/` .

[69] Torstatus - tor network status. http://torstatus.kgprog.com/, April 2008.

[70] Gergely Tóth, Zoltán Hornák, and Ferenc Vajda. Measuring anonymity revisited. In *Proceedings of the Ninth Nordic Workshop on Secure IT Systems*, pages 85–90, 2004.

[71] Mehul Motani Wei Wang and Vikram Srinivasan. Dependent link padding algorithms for low latency anonymity systems. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS 2008)*, pages 323–332, October 2008.

[72] Michael K. Wright, Micah Adler, Brian N. Levine, and Clay Shields. The predecessor attack: An analysis of a threat to anonymous communications systems. *ACM Trans. Inf. Syst. Secur.*, 7(4):489–522, 2004.

[73] Wei Yu, Xinwen Fu, Steve Graham, Dong Xuan, and Wei Zhao. Dsss-based flow marking technique for invisible traceback. In *Proceedings of IEEE Symposium on Security and Privacy*, 2007.

[74] Ye Zhu, Xinwen Fu, Bryan Graham, Riccardo Bettati, and Wei Zhao. On flow correlation attacks and countermeasures in mix networks. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)*, pages 207–225, 2004.