

© 2007 by Parisa M. Tabriz. All rights reserved.

BYZANTINE ATTACKS ON ANONYMITY SYSTEMS

BY

PARISA M. TABRIZ

B.S., University of Illinois, Urbana-Champaign 2005

THESIS

Submitted in partial fulfillment of the requirements  
for the degree of Master of Science in Computer Science  
in the Graduate College of the  
University of Illinois at Urbana-Champaign, 2007

Urbana, Illinois

## ABSTRACT

Anonymous communications systems are analyzed against a range of attacker models to measure the level of privacy they provide users. In this thesis, we survey prior models and define our own attacker, a participating Byzantine adversary, which among other malicious tactics, is capable of a class of path compromise attacks. We demonstrate how the Byzantine participant can execute two specific forms of path compromise attacks on deployed anonymity systems. The first attack challenges the assumptions of the MorphMix collusion detection mechanism, which results in compromising significantly more paths than the original analysis suggested was possible. The second attack demonstrates how selectively breaking connections and forcing continual path reformation in Mix-Net architectures can result in an adversary compromising half of the paths in a system, revealing a fundamental limit on the security of these systems. Finally, we generalize our analysis of this adversary and path compromise attacks to propose countermeasure strategies and challenges.

To the adversary, whether he be globally passive, locally active, or a Byzantine participant.

# Acknowledgments

There are many people that helped make this work possible, or at least much more enjoyable.

I would like to thank...

...my family, for their full and continued support, despite not understanding this research.

...my advisor, Nikita Borisov, for his patient guidance and invaluable insight.

...my co-author, George Danezis, for his collaboration and flexibility in meeting with me from halfway around the world.

...Dawn, for sending me chocolate and caffeine to inspire creative thinking.

...Peter, for offering to read thesis drafts despite being busier than I.

...my friends Anthony, Bill, Chris, Frank, Leo, Matt, Mike, Steve, Yisong and everyone else I've foolishly forgotten to mention.

# Table of Contents

List of Tables . . . . .	viii
List of Figures . . . . .	ix
<b>1 Introduction . . . . .</b>	<b>1</b>
<b>2 Background . . . . .</b>	<b>4</b>
<b>3 The Byzantine Participant . . . . .</b>	<b>6</b>
3.1 Privacy as Unlinkability . . . . .	6
3.2 Common Threat Models . . . . .	7
3.3 A Participating Attacker . . . . .	9
3.4 Path Compromise Attacks . . . . .	10
<b>4 Attacking the Collusion Detection Mechanism of MorphMix . . . . .</b>	<b>12</b>
4.1 MorphMix . . . . .	12
4.1.1 MorphMix and Anonymous Tunnels . . . . .	12
4.1.2 Collusion Detection Mechanism . . . . .	14
4.2 Attacking the Collusion Detection Mechanism . . . . .	16
4.2.1 Attacker Model . . . . .	16
4.2.2 Attacker Goal . . . . .	17
4.2.3 Attack Description . . . . .	17
4.3 Simulation . . . . .	19
4.3.1 MorphMix Settings . . . . .	19
4.3.2 Attacker Settings . . . . .	21
4.3.3 Attack Execution . . . . .	22
4.3.4 Optimized Execution for Smaller Adversaries . . . . .	23
4.4 Attack Countermeasures . . . . .	26
<b>5 Selective Denial of Service Attacks . . . . .</b>	<b>31</b>
5.1 Denial of Service against Conventional Anonymity Systems . . . . .	31
5.1.1 Tor . . . . .	31
5.1.2 Vanilla Mix Networks . . . . .	34
5.2 Denial of Service against Systems Featuring Reliability . . . . .	40
5.2.1 Cashmere . . . . .	40
5.2.2 Hydra-Onions . . . . .	44
5.3 Attacks . . . . .	48

5.3.1	Replay attacks against Hydra-Onions and Cashmere . . . . .	48
5.3.2	Bridging Honest Hydra-Onion Groups . . . . .	51
5.3.3	Reducing the Anonymity Sets in Cashmere . . . . .	53
<b>6</b>	<b>Defenses . . . . .</b>	<b>55</b>
6.1	Path Compromise and Reputation . . . . .	55
6.1.1	Path Compromise Detection and Prevention . . . . .	56
6.2	BFT and Anonymous Communications . . . . .	57
<b>7</b>	<b>Conclusion . . . . .</b>	<b>60</b>
	<b>References . . . . .</b>	<b>62</b>

# List of Tables

3.1	Adversary classifications in anonymous networking systems, their influence on the network (where $C$ is the proportion of dishonest users in the network and $0 < C < 1$ ), and their behavior. . . . .	7
3.2	Adversary classifications and systems that have been analyzed with that threat model. . . . .	8
4.1	Node distribution according to Overnet traffic traces. For example, 72% of Overnet is composed of users coming from unique subnets, 14% of Overnet is composed of users coming from subnets with two active users, etc. . . . .	20
4.2	Tunnel construction for range of attackers. . . . .	23
5.1	Variables used in reliability and security analysis . . . . .	35

# List of Figures

4.1	Successfully constructed tunnels from colluding adversaries with (a) $C = 10\%$ executing an uninterrupted attack and (b) $C = 15\%$ executing an uninterrupted attack. . . . .	24
4.2	Successfully constructed tunnels from colluding adversaries with (a) $C = 5\%$ executing an optimized attack and (b) $C = 10\%$ executing an optimized attack	25
4.3	Correlation distribution and correlation limit of (a) random colluding selections and (b) intelligent colluding selections. . . . .	27
5.1	Reliability and security analysis of Tor under the selective DoS attack, with $f = 0.99$ . . . . .	33
5.2	The effect of different fractions of honest nodes $t$ on the security of the routes, and the replication factor $w$ , for a target reliability of 95%. Experimental results: 5000 samples per point, $l = 5$ (Mixminion), $f = 0.90$ . . . . .	39
5.3	Security of Vanilla mixes for different choices of $l$ under the DoS strategy. . .	39
5.4	The security of Cashmere under different fractions of honest nodes with an expected reliability of 100%. Experimental results: 5000 samples per point, $l = 5$ , and $f = .90$ . . . . .	43
5.5	Replication factor $w$ that achieves reliability of 95% for Hydra-Onions under different fraction of honest nodes $t$ , and the corresponding security. Analytical results. $l = 5$ , $f = 0.90$ . . . . .	46
5.6	Comparing the security of Vanilla mixes, Cashmere, and Hydra-Onions under DOS. . . . .	47
5.7	(a) Hydra-Onion communication patterns and (b) Illustration of the Hydra-Onion attack. . . . .	51
5.8	Illustration of the Hydra-Onion attack. . . . .	51

# 1 Introduction

From the comfort of an Internet accessible computer, one is able to participate in online instant messaging and chat rooms, research and self-diagnose ailments, download digital music, and even apply for a home mortgage! With our daily leisure and work habits continually migrating toward online activities, the need for secure Internet communication is apparent. Cryptographers have tackled the issues of communication secrecy and authenticity, yet there is no complete solution to protecting the identity of parties communicating on the Internet, which can often be as revealing.

Anonymous networking has made strong progress in solving this aspect of security. The area is rich with proposed networking mechanisms and systems that enhance the privacy of users, yet it's almost equally full of attacks that disrupt it. Work that takes the stance of an adversary to privacy only helps to focus our attention toward design weaknesses, ultimately strengthening the technologies that result in widespread deployment.

Whereas many systems have been evaluated with the attacker commanding resources external to the anonymous system, we focus on an adversary acting as a participant in the system that both behaves and misbehaves according to his attack strategy. Due to the dynamic of many deployed anonymous networking systems, a participating adversary has a low barrier to entry and minimal resource costs, and thus, is a highly realistic threat to security.

Most anonymous networking systems relay traffic from some connection initiator through a series of routers, or mixes, before reaching a destination. Each mix attempts to mask the relation between its input and output connections, and by using more than one mix, we can still provide anonymity in the face of dishonest mixes. In this work, we define the path

compromise attack and further describe a strategy where a participating attacker directly compromises each mix in a relay path to disturb security. If an adversary controls enough of the relay path, he is able to trace messages through the network and deanonymize the connection. We present two specific instances of this attack and evaluate their impact on existing systems.

The first attack is on MorphMix, a peer-to-peer anonymous networking system for low-latency communications. MorphMix users incrementally construct anonymous communication tunnels based on recommendations from other users in the system; this approach allows the system to scale to millions of users; however, by allowing unknown peers to aid in tunnel construction, MorphMix is vulnerable to colluding attackers that only offer other attacking nodes in their recommendations. MorphMix employs a collusion detection mechanism to identify this type of misbehavior, yet a participating adversary that constructs specific recommendations is able to successfully defeat this mechanism and compromise a significant portion of communication tunnels.

The second attack we contribute is a selective Denial-of-Service (DOS) attack against all anonymous systems that are designed using the Mix-Net architecture. Whereas DOS attacks were previously thought to only impact system reliability, we show that a participating adversary that selectively breaks connections he cannot compromise has an adverse effect on the security of the systems too. We evaluate this attack against both high and low-latency communication systems as well as anonymous networking systems that are focused specifically on improving reliability.

Finally, we present possible defenses to path compromise attacks by means of reputation management and Byzantine fault tolerance techniques. We explore past attempts at using reputation management and describe some of the difficulties in applying these techniques to anonymous communications.

In Chapter 2, we review existing anonymous networking systems that we later examine in our own work. We define our Byzantine participating attacker model and the path com-

promise attack in Chapter 3 and present two attacks in Chapters 4 and 5 that demonstrate the impact of this attack. In Chapter 6, we present possible defenses to path compromise attacks and Byzantine participants in general, and in Chapter 7, we conclude.

# 2 Background

Over 20 years ago, David Chaum introduced the *mix* as a communication proxy to hide the correspondence between messages coming into and going out of a system [6]. In his original Mix-Net proposal, each message is sent through a sequence of mixes that are chosen independently at random from all available mixes. This sequence of mixes is known as the *relay path*. Messages are encrypted in layers with the public keys of the mixes that route them through the network. Each mix decrypts a layer of the message using its private key, performs some batching strategy, and then forwards it onward until it reaches its eventual destination. This design has been extensively used to build anonymous systems ranging from remailers [10, 21, 18] to low-latency communication systems for anonymous Internet access [2, 12, 15, 29].

Most mix network designs use a relatively small and fixed set of mix servers for forwarding all traffic, usually on the order of several dozen. Mixminion [10], a high-latency mix system for anonymous email, uses such a design with each mix performing proper message batching, reordering, and delaying before forwarding messages to subsequent mixes. Many low-latency anonymous communication systems, such as the Tor network [12], are similarly designed as *static mix networks*. While especially useful for daily tasks such as anonymous web browsing and instant messaging, Tor and systems like it are insecure against many traffic analysis attacks [22]. The current deployment of the Tor network has been pushing the limits of the fixed size network somewhat, with several hundred mix servers in operation, but the network cannot grow much larger without major changes to its design and implementation. This imposes a limit on how much traffic these networks can handle and therefore the size of the user population.

MorphMix [29] represents an alternative, peer-to-peer design for anonymous networks. Each MorphMix user runs a node that both generates anonymous traffic of its own and acts as a mix server, forwarding anonymous traffic for others. MorphMix overcomes some of the drawbacks of fixed size mix networks by providing a scalable approach to anonymous networking that is more resistant to traffic analysis attacks, but introduces a new challenge of detecting collusion during anonymous tunnel construction. In Chapter 4, we test the assumptions of this detection mechanism.

Constructing relay paths through mixes in any of the described systems is an expensive process. In the case of low-latency systems, users often construct many paths which they use repeatedly before they are torn down and reconstructed from scratch. Keeping backup paths ready to use when one is torn down cleanly or broken due to node failure helps amortize this cost. Other mechanisms have been designed to help improve reliability in anonymous networking systems as well. Cashmere [37] is an anonymous routing overlay that uses mix relay groups over single node mixes to improve system reliability. Hydra-Onions [19] improve reliability by sending multiple copies of a message through a tunnel as well as forwarding message duplicates to an additional mix at each step. We find that though both systems do improve reliability, they do so at the cost of reduced security, especially in the face of selective Denial-of-Service attack that is presented in Chapter 5.

# 3 The Byzantine Participant

In this Chapter, we review the various threat models used in system analysis and explain why we focus on the Byzantine participant. We also define the path compromise attack and a specific path compromise strategy that we further evaluate in Chapters 4 and 5.

## 3.1 Privacy as Unlinkability

The level of privacy in anonymous communications is often measured by evaluating how well the system performs in the presence of an adversary trying to disrupt security. In [34], Syverson et al. list the basic security properties of senders, receivers, and connections that anonymous communication systems aim to protect. The authors define *source-destination linking* as the broad anonymity goal in these systems. Source-destination linking is the property that a particular source is communicating with a particular destination. This property implies that an attacker has also compromised both sender and receiver anonymity. As an example of an attack on source-destination linking, consider a closed network of both users and services that users are allowed to access. An observer on this network can determine which users are active and which services are being accessed at all points in time by monitoring which entities are exhibiting network activity. This form of traffic analysis is useful in linking users and the services they are accessing. For example, if only one user is active at a certain period of time and only one service is producing activity, the observer can link a connection between a user and the service even if all of the data on the connection is encrypted. It is this property of linkability that we focus on for the remainder of this work.

In many practical scenarios, users would desire source-destination *unlinkability* since just

Adversary	% of Network Affected	Behavior
Global passive adversary	100%	Observes traffic
Local passive adversary	$C$	Inserts, delays, modifies traffic
Global active adversary	100%	Observes traffic
Local active adversary	$C$	Inserts, delays, modifies traffic
Participating adversary	$C$	Participates in network by running mixes

Table 3.1: Adversary classifications in anonymous networking systems, their influence on the network (where  $C$  is the proportion of dishonest users in the network and  $0 < C < 1$ ), and their behavior.

knowing the identity of both the initiator and recipient of some communication would be enough to compromise user privacy. For example, a person’s safety might be in jeopardy if it is observed that he repeatedly accesses a web site hosting controversial or censored material, even if the contents of what he is accessing are kept private. Alternatively, a person may wish to keep private his communication with an online rehabilitation service since just this connection would be enough for outsiders to make assumptions about his health. In this work, we consider the adversary that is interested in performing source-destination linking in anonymous communication systems.

## 3.2 Common Threat Models

The adversary model that has been adopted for these systems has spanned a range of attackers with varying capabilities. The first four rows in Table 3.1 identify the most commonly used adversary models in anonymous communication system design, the size of their influence on the system, and a brief description of their behavior. Table 3.2 lists the systems we describe in this work and the threat models their designers used in their own security analysis.

In [25], Raymond presents a high-level survey of both active and passive traffic analysis techniques that can be applied to all Mix-Net architectures. The passive adversaries are able to log any traffic they observe on the network, and only differ in the scope of their network view. The global passive adversary is the most common model assumed for analysis

Adversary	Systems
Global passive adversary	Babel, DC-Nets, Mix-Nets, Mixmaster, Mixminion, Tarzan
Local passive adversary	Crowds, Mix-Nets
Global active adversary	Mix-Nets
Local active adversary	Hydra-Onions, Mixminion, Tor
Participating HBC adversary	Cashmere, Crowds
Participating Byzantine adversary	Morphmix, Tarzan, Tor

Table 3.2: Adversary classifications and systems that have been analyzed with that threat model.

of theoretical and high-latency anonymous communication systems such as Babel [18], DC-Nets [7], and Mixminion [10]. Due to traffic analysis attacks, all practical low-latency systems are insecure against the global passive adversary, and thus assume a weaker than global attacker with respect to network coverage, but one that is not merely passive.

The active adversaries can additionally disrupt the system by inserting or delaying messages on any communication link under their control. In [33], Serjantov et al. provide a mathematical analysis of message trickle attacks (or message dropping attacks), message flooding attacks, and attacks using a combination of these techniques. Though [33] models a global active adversary in their evaluation of theoretical mix systems, the more limited active adversary has been adopted in most practical low-latency systems.

Additionally, an active adversary can mark messages as they go through a relay in an anonymous network and try to locate the tagged message later to trace the connection. Mixmaster and Babel are both vulnerable to this form of tagging. Mixminion, however, specifically addresses this threat by using hashes to verify the integrity of message headers at every relay. They additionally have payload verification via cryptographic checksums at a crossover point in the relay path. Outside of tagging, many other active attacks have been developed and analyzed against actual system implementations, including attacks on the cryptographic primitives of Mix-Nets [24] and flooding techniques that exploit the bandwidth limitations of low-latency mixes [22].

### 3.3 A Participating Attacker

The final adversary in Table 3.1 is what we call a *participating adversary*. This adversary contributes to the network by controlling mixes like all other users, but does so with malicious intent. Since the dynamic of most deployed anonymous communication systems consists of volunteers located around the world running their own mixes with no central authority managing users, the only requirements to participate, beyond willingness to do so, are access to a computer and reliable Internet connection. It is both to the benefit and detriment of the system that the resource costs to join are so small, as it encourages all types of users to contribute to the network.

The participating adversary is one that has either compromised or is independently running mixes in a system. We can further define this attacker into two subcategories. The first is an “honest, but curious” (HBC) participant that behaves according to the anonymity protocols of the system, but tries to learn as much as he can about the source and destination of all messages he observes. An HBC participant has been used to analyze the security of Cashmere [37] and Crowds [26]. The impact of this adversary is stronger than that of the local passive adversary since the participating adversary observes the batching of the mixes he runs, and is thus able to correlate the inputs and outputs of those mixes. This threat is the underlying reason we use mix chaining instead of just a single mix for anonymous communication. The rationale is that as long as there is one honest mix selected as part of a relay path, the initiator and recipient will remain unlinkable.

The second classification is a participating adversary that both behaves and misbehaves according to the system protocols, depending on his current strategy. This adversary is allowed to lie about his capabilities, lie about his knowledge of the network, and disrupt communications. Additionally, he can behave honestly. This makes him especially dangerous because it is hard, if not impossible, to isolate him from normal users. We can model this adversary’s arbitrary behavior as a Byzantine adversary. The Byzantine failure model [20]

has been increasingly used to evaluate the security of distributed systems [16]. This is because in the worst case, an adversary that exhibits random behavior can be considered as acting malicious, and therefore if a system tolerates Byzantine faults, it can also be regarded as secure from malicious adversaries.

Due to the loose organization of membership in anonymous communication systems and low resource cost to participate, we consider the participating adversary an especially realistic threat. Additionally, since the participating adversary is directly controlling mixes, the resources required to act as a Byzantine participant are no more than those needed to act as an HBC participant, so we focus our attention on this more dangerous case of a Byzantine attacker.

### 3.4 Path Compromise Attacks

A *path compromise attack* is any strategy that takes advantage of a relay path formation protocol to degrade the security of an anonymous communication system. Before a message can be sent through the system, a mix relay path needs to be determined to appropriately encrypt the message. Paths are either chosen from a pre-existing set of valid relay paths, also known as *mix cascades*, or they are constructed independently by the user. In [3], Berthold et al. compare and contrast the security of using mix cascades to using mix relay paths that are freely constructed. While mix cascades are less flexible, they are resilient to some message blocking attacks and global traffic analysis that deployed Mix-Nets are unprotected against. Alternatively, it has been shown that freely constructed relay paths can provide better anonymity and message reliability when using synchronous message batching designs [13].

In the case of free path construction, a path can either be created by the user selecting a series of mixes to relay traffic through or can be constructed iteratively by the recommendations of other mixes in the system. The path formation protocol itself can leak information about relay paths in the system. In [35, 36], Wright et al. demonstrate the theoretical and

practical impact of a passive logging attack that can identify the initiator of a connection based on path reformation over long-term connections. An HBC participating adversary can count, for each user, which user sends a message as part of an identifiable stream. Over time, as the relay path between initiator and destination is continually reconstructed, the user with the highest log count is deemed the initiator of the connection. The attack simply takes advantage of the fact that all users will be present in a relay path with equal probability except for the initiator, who will be present more often.

Path compromise attacks have also been used to degrade the anonymity of anonymous services. An attack presented in [23] demonstrates how attackers in Tor can uncover the location of hidden services based on the properties of the relay path between a rendezvous point and the hidden service.<sup>1</sup>

Instead of exploiting information gathered from the protocol itself, an adversary can simply try to influence how the relay path that a user sends its messages over is built. If we consider a system that constructs paths of length  $l$  from selecting mixes uniform randomly from all available mixes in the system,  $C$  of which are malicious, we would expect to have a completely compromised path with probability  $C^l$ . If the adversary has compromised all of the mixes in the relay path, he is able to link all messages sent by anyone through those mixes to a destination, trivially linking connections. However, system design and implementation rarely enforce strict random selection of mixes in relay paths, so a participating adversary can try to bias this construction and increase his odds of full path compromise. In the following sections, we describe two specific examples of this biasing path compromise attack.

---

<sup>1</sup>Since its publication, the developers have introduced entry guard nodes that diminish the success of this attack and are based on fixed entry mixes, a concept first proposed in [36].

# 4 Attacking the Collusion Detection Mechanism of MorphMix

In this Chapter, we present a path compromise attack on the MorphMix anonymous networking system. MorphMix nodes incrementally construct anonymous communication tunnels based on recommendations from other nodes in the system; this P2P approach allows it to scale to millions of users. However, by allowing unknown peers to aid in tunnel construction, MorphMix is vulnerable to colluding attackers that only offer other attacking nodes in their recommendations. To avoid building corrupt tunnels, MorphMix employs a collusion detection mechanism to identify this type of misbehavior. In the following sections, we review the MorphMix design and challenge the assumptions of the collusion detection mechanism to demonstrate that colluding adversaries can compromise a significant fraction of all anonymous tunnels, and in some cases, a majority of all tunnels built.

## 4.1 MorphMix

### 4.1.1 MorphMix and Anonymous Tunnels

MorphMix is a circuit-based mix network consisting of many MorphMix clients, or *nodes*, that act as both connection initiators and routers for the network. Each MorphMix node maintains a limited number of *virtual links* via TCP connections to *neighbor* nodes within the system. One unique feature of MorphMix is that the route a node uses for its connection, an *anonymous tunnel*, is constructed iteratively by other participating nodes in the system. We briefly describe MorphMix’s anonymous tunnel construction, and refer the reader to [27] for a more detailed look at the MorphMix system and tunnel construction protocol.

An anonymous tunnel consists of the node establishing the connection, the *initiator*, zero or more *intermediate* nodes, and the *final* node of the anonymous tunnel. Similar to other onion routing mix networks like Tor, MorphMix uses fixed size messages and layered encryption across each link of the anonymous tunnel to prevent against traffic analysis attacks and protect message content, respectively.

When an initiator node,  $a$ , wants to create an anonymous connection, it first establishes a shared key with one of its neighboring nodes, say  $b$ , which will be used to encrypt messages sent across that link of the tunnel. If  $a$  decides to extend the tunnel, it asks node  $b$  to recommend a *selection* of nodes from  $b$ 's neighbors to use as a next hop in the anonymous tunnel. Node  $a$  then chooses from the offered selection a node, say  $c$ , to append to  $b$  in the tunnel. Node  $a$  establishes a symmetric key with  $c$  via  $b$  that it will use for encryption across the next link in the tunnel. To prevent against  $b$  performing a man-in-the-middle attack between  $a$  and  $c$ ,  $a$  selects a *witness* node from the nodes it already knows<sup>1</sup> to establish the symmetric key between  $a$  and  $c$ . There are other attacks that can be considered if the witness is in collusion with  $b$ , but to simplify our presentation, we ignore this case and assume the witness is always an honest node. Once a tunnel to  $c$  is established,  $a$  can ask  $c$  for a selection of nodes to extend the tunnel further; this process continues until  $a$  has finished appending nodes to the tunnel.

By using the last hop to discover options for the next hop, MorphMix nodes only need to maintain state information about their local neighbors. This allows MorphMix to scale independently of the number of nodes in the system. However, an immediate threat is introduced when a malicious node is appended because it helps determine the next hop in the tunnel. To prevent a malicious node from offering selections biased with other malicious nodes, MorphMix employs a *collusion detection mechanism* (CDM) to identify this behavior and prohibit this form of attack. MorphMix assumes that a tunnel is compromised, or *mali-*

---

<sup>1</sup>MorphMix uses a peer discovery mechanism to learn about other nodes in the network to use for witness and neighbor selections.

*cious*, if the first intermediate and final node are both controlled by an attacker. Otherwise, it considers the tunnel *fair*.

### 4.1.2 Collusion Detection Mechanism

Similar to other anonymous systems, we assume the primary goal of an attacker in MorphMix is to link communications between initiators and recipients of a connection. While low-latency systems are generally more susceptible to traffic analysis, MorphMix is vulnerable to a more immediate attack when colluding nodes try to append other colluding nodes during tunnel construction. MorphMix detects this behavior by performing collusion detection on each offered selection.

If attackers own a whole range of IP addresses, it would be easy for them to operate many MorphMix nodes. To limit this threat, MorphMix distinguishes individual nodes from each other by their 16-bit IP address prefix. We refer to this prefix as the node's /16 subnet. It is much more costly and difficult for attackers to own nodes in many unique /16 subnets than it is for them to own many nodes in one or more /16 subnets. The CDM is built on the following two assumptions: honest selections will be comprised of nodes selected randomly from many different /16 subnets and malicious selections will be comprised of mostly or all colluding nodes coming from a limited portion of /16 subnets in MorphMix.

Each MorphMix node maintains a fixed size *extended selection list*,  $L_{ES}$ , of entries consisting of the concatenation of the selection and the 16-bit IP prefix of the node that sent that selection; we refer to this entry as an *extended selection* and each 16-bit IP prefix in the entry as the node's subnet. When a tunnel initiator receives a new selection, it compares it to each entry in its  $L_{ES}$  and calculates the proportion of node subnets that have been seen multiple times to those that have been seen only once. The computed correlation is expected to be larger for a colluding selection than it is for an honest selection because colluding nodes will limit their selections to only other nodes in their colluding set and honest nodes will

offer selections consisting of neighbors that have been chosen more or less randomly from all nodes in the system. The algorithm, described in [29], is repeated below:

#### Correlation Algorithm

1. Build a set  $ES_N$  consisting of the 16-bit IP address prefixes of the nodes in the new extended selection.
2. Define a result set  $ES_R$  which is empty at first.
3. Compare each extended selection  $ES_L$  in the extended selections list  $L_{ES}$  with  $ES_N$ . If  $ES_N$  and  $ES_L$  have at least one element in common, add the elements of  $ES_L$  to  $ES_R$ .
4. Count each occurrence of elements that appear more than once in  $ES_R$  and store the result in  $m$ .
5. Count the number of elements that appear only once in  $ES_R$  and store the result in  $u$ .
6. Compute the correlation  $c$  which is defined as  $c = \frac{m}{u}$  if  $u > 0$ , or  $\infty$  otherwise.

Each MorphMix node remembers the correlations it has computed over recent extended selections and represents these in a *correlation distribution*. Every time a node receives a new extended selection, it computes a correlation and updates this distribution according to an exponential weighted moving average. The results in [27] show that these distributions can often be characterized as having two peaks, one formed by the aggregate contribution of honest nodes and one formed by the aggregate contribution of malicious nodes (see Figure 4.3a). From this distribution, MorphMix determines a threshold point between the two peaks, the *correlation limit*, which has the property that correlations greater than this limit are malicious with high probability and correlations less than this limit are honest with high probability.

During tunnel construction, the initiator calculates the correlation of every extended selection it receives and compares this to its correlation limit. If any extended selection during the setup is detected as malicious, the tunnel is torn down and not used. Otherwise, the tunnel is considered fair and used for anonymous connections.

There are two important assumptions to highlight from the collusion detection mechanism that form the intuition behind our attack:

1. An extended selection from a colluding node will overlap with many other colluding entries stored in the  $L_{ES}$ , resulting in a large  $c$ .
2. The  $c$  of a malicious extended selection will, in general, be higher than the correlation limit determined for that node.

By limiting the number of selections that overlap in a victim's  $L_{ES}$ , a colluding adversary can keep  $c$  low such that it frequently falls beneath the correlation limit and is not detected during tunnel construction.

## 4.2 Attacking the Collusion Detection Mechanism

In this section, we define the adversary necessary to perform this attack, describe the attacker's goal, and present a general description of the attack.

### 4.2.1 Attacker Model

Anyone with access to the Internet and a MorphMix client can actively participate in MorphMix. Because attackers can so easily join, contribute, and exit the system, we assume an active, internal adversary. Specifically, we assume that there is some subset,  $n_c$ , of all MorphMix nodes,  $n$ , that is comprised of colluding nodes from unique subnets that are participating in MorphMix. In reality, the number of colluding nodes may be larger than  $n_c$ , but because the CDM does not differentiate between two nodes from the same subnet, we

only consider the number of colluding nodes that can be represented by unique subnets. We assume the colluding set will conspire to choose how they offer selections to a victim node, but otherwise, behave honestly.

We specify that  $n_c$  will be comprised of nodes representing a percentage,  $C$ , of the unique subnets in MorphMix, where  $C$  can realistically range from 0% to 40%. This range represents different attackers present in the system, from a small group of attackers to an organization of moderate resources to an even larger network of compromised zombie machines. This range similarly follows the assumptions made by the MorphMix authors. Consequently, we analyze the success of our attack using a colluding set ranging in size from 5% to 40% of the unique subnets in MorphMix.

### 4.2.2 Attacker Goal

We assume that the goal of attackers is to link a connection initiator with some outgoing stream. Attackers can achieve this goal by owning the first intermediate and final node in an anonymous tunnel. This will happen with probability  $C^2$  during normal MorphMix behavior. Our attackers, however, accomplish linkability by owning every node in the tunnel. We aim to show that by using intelligent selections, colluding attackers can expect to compromise every node in  $C$  anonymous tunnels built by some victim node.

### 4.2.3 Attack Description

Our attack is based on this simple intuition: Because each node's CDM is based on only the local knowledge stored in its  $L_{ES}$ , attackers can model and manipulate the  $L_{ES}$  to avoid being detected. To accomplish this, colluding nodes should only offer other colluding nodes in their selections, and they should organize and offer selections to a victim in such a way that they have the least overlap with other malicious selections in the victim's  $L_{ES}$ . The analysis in [27] simulates attackers offering selections that are comprised of nodes *randomly*

chosen from the set of all colluding nodes. By being more intelligent with their selections, colluding attackers can limit the number of nodes in extended selections that contribute to  $m$  in the correlation algorithm. The attack works as follows:

#### Intelligent Selection Attack

1. For every victim,  $v$ , construct a list of selections,  $S_v$ , comprised of only colluding nodes such that there is no overlap in node subnets between any selection entry in  $S_v$ .
2. Maintain a global pointer,  $p_g$ , that keeps reference to a selection in  $S_v$  to be offered in the next attack attempt.
3. When  $v$  contacts any colluding node to be a first intermediate node and any subsequent node in a new anonymous tunnel, we offer to  $v$  the selection pointed to by  $p_g$  and increment  $p_g$ . If  $p_g$  pointed to the last element in  $S_v$ , we set  $p_g$  to be the first element of  $S_v$  and iterate once again through all elements in the list.

The above attack assumes that a colluding node can determine if it is the first intermediate node *during* tunnel construction. MorphMix makes this difficult by using the same witness mechanism for every step of tunnel construction, including the first. Therefore, a node cannot determine whether it is the first node or a later node from only the messages exchanged in the append protocol. Measuring message delays can help determine the position in the list, but in our attack, a node must decide if it is in the first position before returning the selection, with not enough messages exchanged to measure timings.

We therefore modify our attack to return selections from  $S_v$  for all tunnels arriving at a colluding node from  $v$ , including ones that may have originated from nodes other than  $v$ . After the fact, once the tunnel is fully constructed, it is easy to determine whether it originated from  $v$  by counting the number of links after  $v$ , since all nodes past  $v$  will be colluding. (The effects of variable tunnel lengths will be discussed in Section 4.4.)

## Intelligent Selection Attack (Revised)

1. Whenever  $v$  requests a selection from a colluding node, we begin the attack by assigning a local pointer,  $p_l$  to the selection referenced by  $p_g$  and offer that selection to  $v$ . We cannot verify if  $v$  is the initiator of the tunnel or a node appended to a tunnel started by some other node,  $v'$ .
2. For every successive selection request, we increment  $p_l$  in  $S_v$  and offer the new selection that  $p_l$  points to.
3. After the tunnel is created, we determine if the tunnel initiator was  $v$  or some  $v'$  by measuring the tunnel length.
4. If the tunnel was initiated by  $v$ , we update  $p_g$  to hold the value referenced by  $p_l$ . Otherwise, some  $v'$  was the initiator of this tunnel. Since our attack selections are stored in the  $L_{ES}$  of  $v'$ , they can still be used against  $v$ . In this case,  $p_g$  maintains its original value.

Next we will use simulations to determine how effective this attack is at avoiding the collusion detection mechanism.

## 4.3 Simulation

### 4.3.1 MorphMix Settings

Because MorphMix does not have a substantial user base, we are unable to execute this attack on a live system. Instead, we simulate many tunnel constructions using the CDM from the MorphMix client prototype [28] and investigate the effects of the attack on one node, the victim node. We evaluate how successful the attack is based on how many tunnels we can compromise, what proportion of all tunnels constructed can be compromised, and how long the attack can run successfully.

Users per Subnet	Percentage of Overnet
1	72%
2	14%
3	7%
4	4%
5	2%
6	1%

Table 4.1: Node distribution according to Overnet traffic traces. For example, 72% of Overnet is composed of users coming from unique subnets, 14% of Overnet is composed of users coming from subnets with two active users, etc.

The analysis in [27] simulates construction of 5000 anonymous tunnels from a network of 10,000 MorphMix nodes with every node coming from a unique /16 subnet. We believe this is not indicative of a realistic user distribution for an unstructured, decentralized network such as MorphMix. The real Internet is composed of a high concentration of users from certain subnets and we choose to represent this imbalance. Additionally, each node’s correlation limit in MorphMix is based on the correlations of all recent selections it has seen, both honest and malicious. Because of this, it is important to simulate as realistic a network distribution as possible, namely, one that consists of users coming from both common and unique subnets. We look to a popular P2P system of similar structure, the Overnet/eDonkey file-sharing system, to provide more realistic statistics [1].

Resulting from traffic probes taken during 2003, Overnet consisted of the subnet to node distribution displayed in Table 4.1 [4]. We simulated 5000 tunnel constructions consisting of only honest selections from both the original node distribution in [27] and our own node distribution based on the Overnet traces. The average correlation limit in the original distribution was .145 ( $\sigma = .005$ ) and the average correlation limit using the Overnet trace was .172 ( $\sigma = .005$ ), a significant difference given identical network parameters between the two simulations aside from the underlying node distribution. From Figure 4.3a, we can see that the distance between the peak formed by honest selections and the peak formed by malicious selections is already very small. Increasing the correlation limit by even a small

amount will make this distance even closer and the correlation limit harder to define. While the Overnet spread may not exactly represent a deployed MorphMix system, we believe it is more indicative of P2P use than the one used in the original MorphMix simulation. For this reason, we follow this distribution during our experimentation.

Aside from this change, we use the same fixed tunnel size of 5 nodes and compute the size of the  $L_{ES}$  and the size of node selections identical to [27].<sup>2</sup> Every node in MorphMix maintains virtual links to neighbors that are chosen as the first intermediate node during tunnel construction. Virtual links with other nodes are established more or less randomly from all of the nodes known to a user. We consider that  $C$  tunnels will actually begin with a colluding node and  $(1 - C)$  will begin with an honest node and be impossible to compromise.

In [30], the authors of MorphMix do an additional analysis of the CDM, taking into account node tunnel acceptance rates and uptime probabilities. We have chosen to ignore these additional constraints in our simulation for simplicity, but believe they would only strengthen the success of the attack: while honest nodes may occasionally refuse to accept new tunnels and leave the network due to limited capabilities, malicious nodes are likely to devote more resources to their attacks and have higher acceptance probabilities and network uptimes. Therefore, we can expect even fewer than  $(1 - C)$  tunnels would have a first intermediate honest node.

### 4.3.2 Attacker Settings

Attackers will blindly assume that any initial selection request from  $v$  (and all subsequent selection requests) are contributing to a tunnel initiated by  $v$ . If  $v$  was not the tunnel initiator though, the attack selections destined for  $v$  actually arrived at some other node,  $v'$ , and are stored in that node's  $L_{ES}$ .

If our attack is being executed against many victim nodes in MorphMix, attacker se-

---

<sup>2</sup>In MorphMix, tunnel size is fixed for the duration of the session and has a default value of 5 nodes. While this value can be changed upon restart, we assume most users would keep the default configuration. We address the effects of violating this assumption in Section 4.4.

lections destined for one victim may accidentally be misdirected to a different victim. To minimize the effect these misdirected selections have on the collusion detection mechanism, the attackers should use a different random permutation of  $n_c$  when constructing  $S_v$  for each different victim  $v$ . When  $v'$  receives a selection destined for  $v$ , it will appear as a random sample of nodes from  $S_{v'}$ , and that is in fact how we model misdirected selections in our simulations. More specifically, whenever a node tries to extend a tunnel that starts with an honest node, with probability  $C$  a malicious next node is chosen, who will then return a misdirected selection. The misdirected selection is represented as a random set of nodes from  $n_c$ . If the tunnel is then extended further, we assume the malicious node will carry out the attack and provide more misdirected selections, once again represented by a random sample from  $n_c$ .

We simulate the modified attack as described in Section 4.2.3 by first creating a random permutation of attacking nodes and storing this ordering into  $S_v$ . We select the first  $k$  nodes, where  $k$  is the selection size, and continue to cycle through  $S_v$  to create unique selections.

We briefly explored more sophisticated ways of creating  $S_v$  such that more selections can be made with minimal overlap, however, our initial results showed that even a basic organization of how colluding selections are offered is enough to result in significant attacker success.

### 4.3.3 Attack Execution

We execute the attack during 5000 tunnel construction attempts by a single victim node and calculate how many successful tunnels are constructed. In MorphMix, a node creates, on average, a new anonymous tunnel every 2 minutes. Therefore, creating 5000 tunnels is roughly equivalent to one week of constant MorphMix usage. In Table 4.2a, we can see that the attack results in a significant portion of anonymous tunnels being compromised using intelligent selections. If colluding adversaries control nodes in more than 15% of the represented subnets in MorphMix, they are able to compromise at least that percentage

(a) Uninterrupted attack execution.

$C$	Honest Tunnels	Malicious Tunnels	Percentage Compromised
5%	3337.9	6.8	0.2% ( $\sigma = 0.1\%$ )
10%	2951.4	33.8	1.1% ( $\sigma = 0.2\%$ )
15%	2283.2	470.1	17.1% ( $\sigma = 1.5\%$ )
20%	1930.0	860.4	30.8% ( $\sigma = 1.1\%$ )
30%	1171.5	1384.0	54.2% ( $\sigma = 2.4\%$ )
40%	450.9	1847.5	80.4% ( $\sigma = 2.3\%$ )

(b) Optimized attack execution.

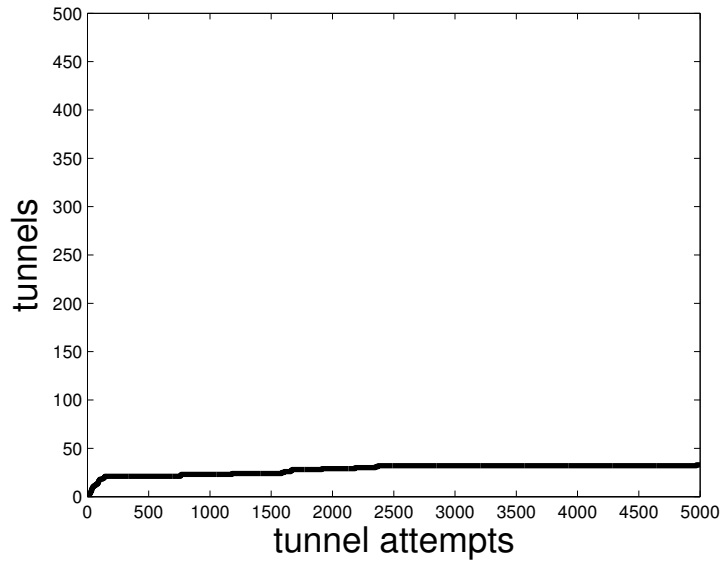
$C$	Honest Tunnels	Malicious Tunnels	Percentage Compromised
5%	4251.9	51.8	1.2% ( $\sigma = .2\%$ )
10%	4161.2	146.9	3.4% ( $\sigma = .2\%$ )

Table 4.2: Tunnel construction for range of attackers.

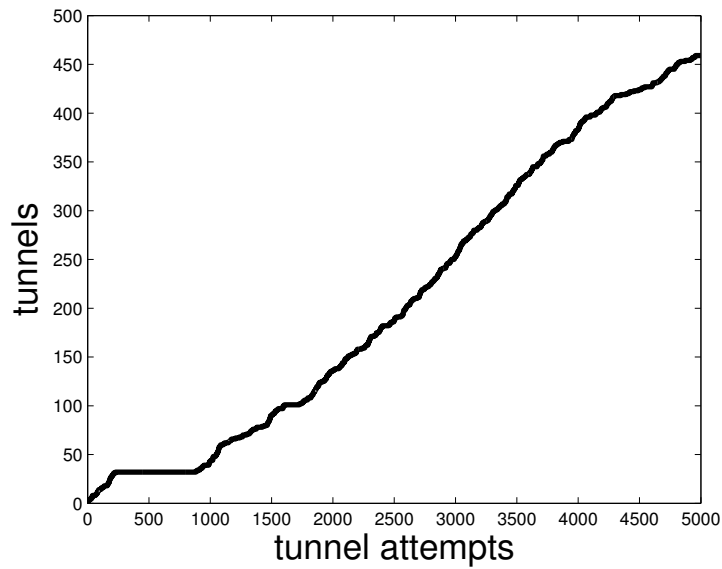
of tunnels constructed by victims. Attacking levels above 30% result in the majority of all constructed tunnels being compromised by an attacker. While adversaries that control nodes in less unique subnets cannot claim quite as high statistics, by slightly adjusting the attack, they can still successfully compromise more than  $C^2$  anonymous tunnels.

#### 4.3.4 Optimized Execution for Smaller Adversaries

The main problem that attackers have when they own few nodes in unique subnets is that they are more limited in the number of unique selections they can create. If they continue the attack uninterrupted, these selections will begin to overlap in a victim's  $L_{ES}$ , causing the correlation and chance of detection to raise. If attackers owning nodes in less than 15% of the unique subnets in MorphMix attack uninterrupted, they will eventually saturate the victim's  $L_{ES}$  and be detected with high probability once they starts repeating selections. In this case, they can optimize their attack by using intelligent selections to build tunnels until they runs out of unique selections and then behave normally until the victim's  $L_{ES}$  has cleared. Because nodes evict the oldest entries from their  $L_{ES}$ , attackers can estimate how long it will take for the victim's  $L_{ES}$  to be cleared based on how often and at what rate

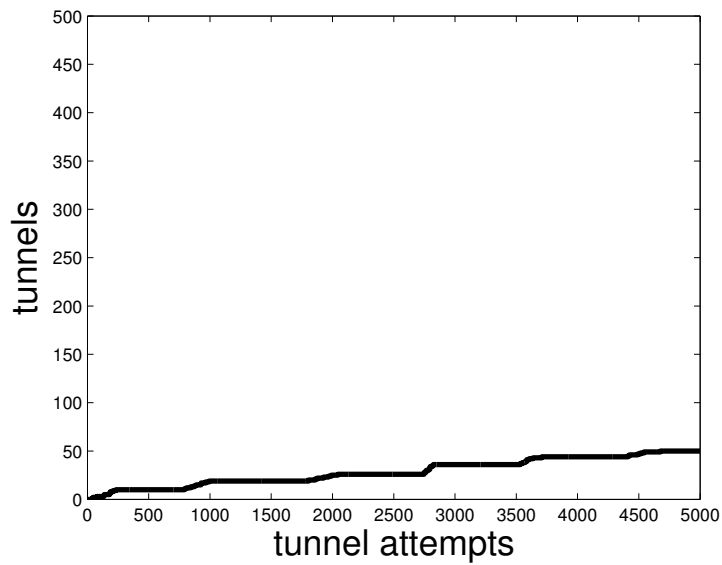


(a)

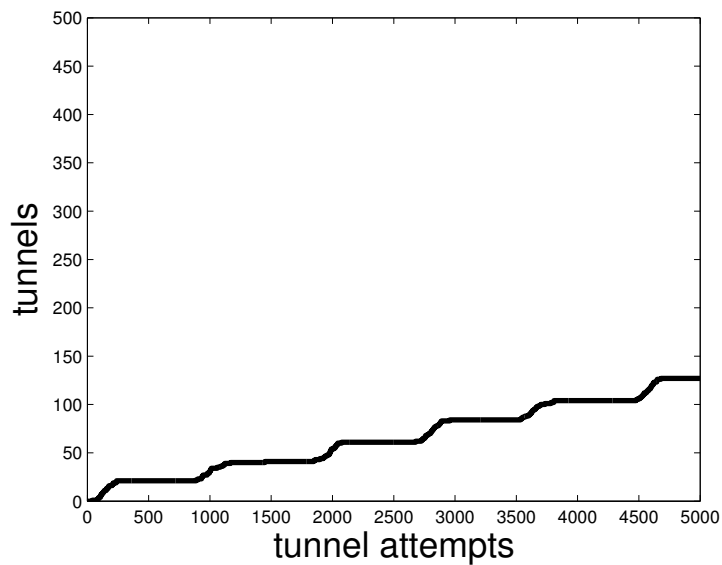


(b)

Figure 4.1: Successfully constructed tunnels from colluding adversaries with (a)  $C = 10\%$  executing an uninterrupted attack and (b)  $C = 15\%$  executing an uninterrupted attack.



(a)



(b)

Figure 4.2: Successfully constructed tunnels from colluding adversaries with (a)  $C = 5\%$  executing an optimized attack and (b)  $C = 10\%$  executing an optimized attack

the victim creates tunnels. Both of these parameters have initial values in each MorphMix client, and even if they are changed, they are limited by a small range of realistic values.

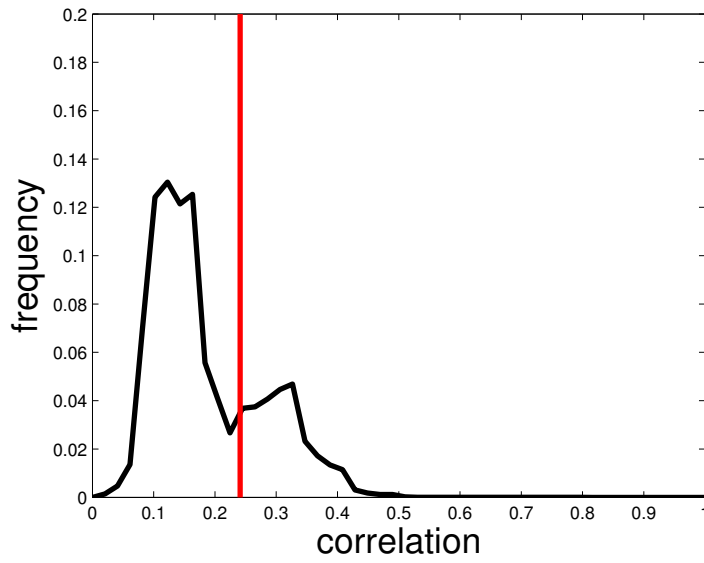
We test this strategy during 5000 tunnel attempts using identical simulation parameters and present the results in Table 4.2b. In Figure 4.2 and Figure 4.1, we compare the results of the uninterrupted and optimized attack for  $C$  ranging from 5% to 15%. We note that attackers with  $C = 10\%$  executing an uninterrupted attack can only compromise around 30 tunnels before they have saturated the  $L_{ES}$  and cannot compromise any more tunnels. However, if they attack until they run out of unique selections and then wait until the node’s  $L_{ES}$  has cleared, they can compromise almost five times as many tunnels and can continue to attack the victim in this manner indefinitely.

This interrupted strategy is only necessary for colluding attackers with limited resources. Specifically, it is only necessary for those with nodes in less than 15% of the uniquely represented subnets in MorphMix. As seen in Figure 4.1b, attackers with  $C = 15\%$  can continue to compromise tunnels indefinitely without waiting for a victim’s  $L_{ES}$  to clear.

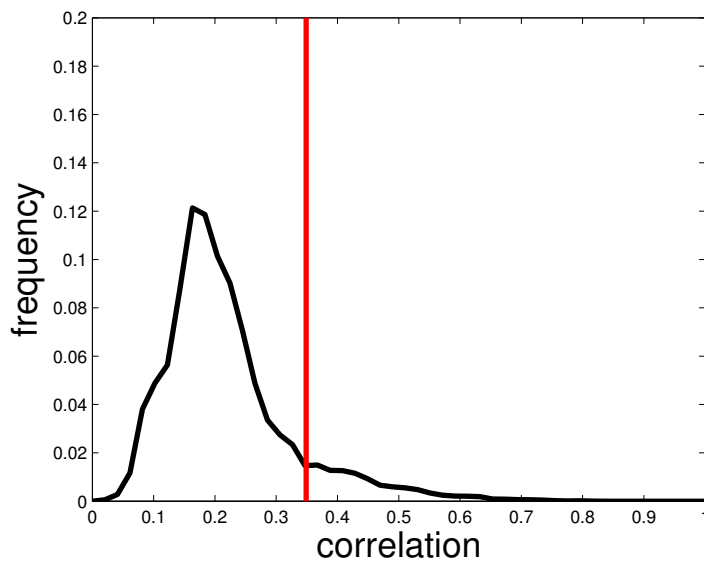
In theory, there is an improved strategy that a limited attacker can use when behaving honestly. Attackers should provide selections that consist of very few malicious nodes and many other unique honest nodes during this honest behavior period. This way, the victim’s  $L_{ES}$  becomes filled with selections that will overlap with future attacking selections, yet make a large contribution to  $u$  in the correlation algorithm. This will, in turn, lower the correlation and decrease the chance of future detection.

## 4.4 Attack Countermeasures

In Section 4.1.2, we reviewed the CDM and how the correlation limit is determined in practice. As shown in [27], when colluding adversaries provide selections of nodes that are *randomly* selected from only participating attackers, the contribution of these selections to the correlation distribution forms a distinguishable second peak to the right of the contribution



(a)



(b)

Figure 4.3: Correlation distribution and correlation limit of (a) random colluding selections and (b) intelligent colluding selections.

of honest selections. We reproduce this effect by simulating an attacker that controls nodes in 20% of the unique subnets in MorphMix and attacks with selections of only randomly chosen malicious nodes. The resulting correlation distribution and correlation limit are displayed in Figure 4.3a. Next, we simulate the same adversary using *intelligent* selections. The results in Figure 4.3b show that using this method destroys the dual-peak characteristic of the correlation distribution. This, in turn, creates a less meaningful correlation limit, crippling the detection mechanism.

An immediate countermeasure to this attack might be to increase the number of nodes in the tunnel and increase the number of entries in the  $L_{ES}$ . Increasing the number of nodes in the tunnel would force the attackers to use more selections for each tunnel. This would cause their attacking selections to overlap much sooner in the  $L_{ES}$ , driving up the correlation before as many tunnels can be compromised. Increasing the number of entries in the  $L_{ES}$  has a similar consequence because it allows each node to store more attacker selections at one time. An immediate drawback to this approach is that it has a two-fold impact on system performance. Increasing the size of the tunnel will increase connection latency as messages will need to be routed through more nodes. Increasing the size of the  $L_{ES}$  will require greater storage and more computation for each execution of the CDM during tunnel construction.

Alternatively, one might introduce variable length tunnels into MorphMix. If an attacker doesn't initially know the true length of the tunnel, it is more difficult to determine if he owns the first and last nodes; however, tunnel length is limited by a small range of realistic values. The analysis in [27] noted that while longer tunnels (eg. 10 nodes) offer greater protection than shorter tunnels (eg. 3 nodes), they also incur a higher connection latency and result in higher bandwidth usage by the MorphMix network. They also will increase the chances of tunnel failure if a node leaves the network or purposefully breaks a connection. Taking this into account, an attacker can estimate the distribution of tunnel lengths in MorphMix and the probability that it has compromised the entire tunnel.

We briefly evaluate a scenario where initiators create anonymous tunnels with lengths between 5 and 7, chosen at random. Because of the variable tunnel lengths, an attacker cannot be positive about whether to roll back selections when the number of appended nodes is either 5 or 6. We use a simple strategy of rolling back whenever the appended tunnel length is less than 6; this results in some number of incorrect rollbacks, which re-send the same selections to the same victim node, and some missed rollbacks, where some selections are skipped and never sent to the victim. However, these problems are relatively infrequent, and our simulations of an adversary who has compromised 20% of the MorphMix nodes can still compromise 18% of all MorphMix tunnels. Thus, the use of variable tunnel lengths slows down, but does not eliminate our attack. The scenario we considered produces a marginal increase in security, but introduces higher latency for constructed tunnels. Introducing even greater variability will result in still higher costs and thus reduced adoption by users.

New users to MorphMix are especially vulnerable to the intelligent selection attack. Since new users enter the system with an empty  $L_{ES}$ , attackers are guaranteed to successfully compromise a significant portion of a new user’s initial tunnels, regardless of the  $L_{ES}$  size. This type of initial behavior in MorphMix will presumably limit its adoption. Most importantly, neither of these methods prevents the attack, and instead, only delays its success. As described in Section 4.3.2, attackers can optimize their attack strategy so that they can still build a significant number of anonymous tunnels given fewer unique attack selections.

The general limitation of the CDM is that it only considers a node’s local knowledge when detecting collusive behavior. Specifically, it distinguishes between honest and colluding selections based only on the selections the individual node has previously seen, not taking into account the behavior of the rest of the network when calculating its own correlation limit. Also, because nodes evict the oldest entries from their  $L_{ES}$ , attackers can estimate when a victim’s  $L_{ES}$  will be cleared of attacking selections and then once again begin the attack. These two factors make it easy for attackers to not only model and manipulate what a node has stored in its  $L_{ES}$ , but improve their attack strategy based on this information.

Although the CDM may be adjusted to capture some possible attack strategies, attackers can stay one step ahead by modeling the state and algorithms of the CDM at each node and crafting the best possible response, consisting of both honest and colluding selections.

An effective collusion detection mechanism for MorphMix requires a more global perspective of the network. One instance of this might be to enforce a double check on any offered selection during tunnel construction. Every time a tunnel initiator wants to append a node to its tunnel, he contacts a unique witness to help establish the symmetric key between the initiator and the new end node. Additionally, it is the witness that chooses which node from the offered selection to use for the next hop. Requiring that a selection correlation fall beneath the correlation limit of the initiator and witness when appending a node may double the chances of it being detected; however, it may also adversely affect the false positive rate when evaluating honest selections. Nevertheless, while this may improve the mechanism's detection rate, it still doesn't provide a thorough view of network behavior and more sophisticated schemes are likely needed.

# 5 Selective Denial of Service Attacks

In this Chapter, we consider the effect attackers that disrupt anonymous communications have on the security of traditional mix systems, as well as on the Hydra-Onion and Cashmere systems that aim to offer reliable mixing. A Denial-of-Service (DOS) path compromise attack lowers anonymity as messages need to get retransmitted to be delivered, uncovering a fundamental limit on the security of all mix systems; Cashmere and Hydra-Onion security is also badly affected by these DOS attacks.

## 5.1 Denial of Service against Conventional Anonymity Systems

### 5.1.1 Tor

#### Background

Tor [12] is a widely used system for low-latency anonymous Internet communication. The network has enjoyed quick growth since its initial deployment in 2003; as of November 2006, Tor is composed of approximately 800 active routers supporting hundreds of thousands of users.

Communication over Tor happens through *tunnels* that are sent via multiple Tor routers. The tunnels are constructed in a telescoping manner and are protected by layered encryption, so that each router only knows the previous and next routers forwarding the tunnel. In Tor, the low-latency nature of the communication allows the first and last router in a tunnel to collude and easily discover that they are forwarding the same data stream by matching packet

timings. Therefore, under conventional analysis, if  $\bar{t}$  is the fraction of all Tor routers that are compromised, then  $\bar{t}^2$  is the probability that any individual tunnel will be compromised [34]. This is the same principle we saw earlier with MorphMix, where we assumed a tunnel was malicious if the first intermediate and last node were compromised.

## Reliability Analysis

The reliability of Tor tunnels has not been previously analyzed but is straightforward to determine. A Tor tunnel that goes through  $l$  routers ( $l$  is typically 3) will fail if any of the routers fail. Therefore, if  $f$  is the probability of a router being reliable,  $R = f^l$  is the probability of the entire tunnel being reliable.<sup>1</sup>

Next, we analyze the effect of a selective denial of service on Tor. We assume that dishonest routers will perform DoS on any tunnel they cannot compromise. This attack is easy to implement: if the adversary acts as a first or last router on a tunnel, the tunnel is observed for a brief period of time and matched against all other tunnels where a colluding router is the last or first router, respectively. If there is a match, the tunnel is compromised; otherwise, the adversary kills the tunnel by no longer forwarding traffic on it. The adversary also kills all tunnels where it is the middle node, unless both the previous and the next hop are also colluding.

Under this attack, a tunnel is reliable only if the first and last nodes are compromised, or if it is composed of only reliable honest nodes. So the overall reliability of Tor in this case is:

$$R_{\text{DoS}} = (1 - t)^2 + (tf)^3$$

Figure 5.1 plots the reliability of Tor under the selective DoS attack as a function of  $t$ , with  $f = 0.99$ . The reliability decreases as the number of compromised nodes grows, until it reaches a minimum at  $t = 0.55$ , at which point it starts to rise again. This is because at

---

<sup>1</sup>This calculation simplifies away the detail that routers are picked without replacement, but with  $l = 3$  and 800+ routers, this is a suitable approximation.

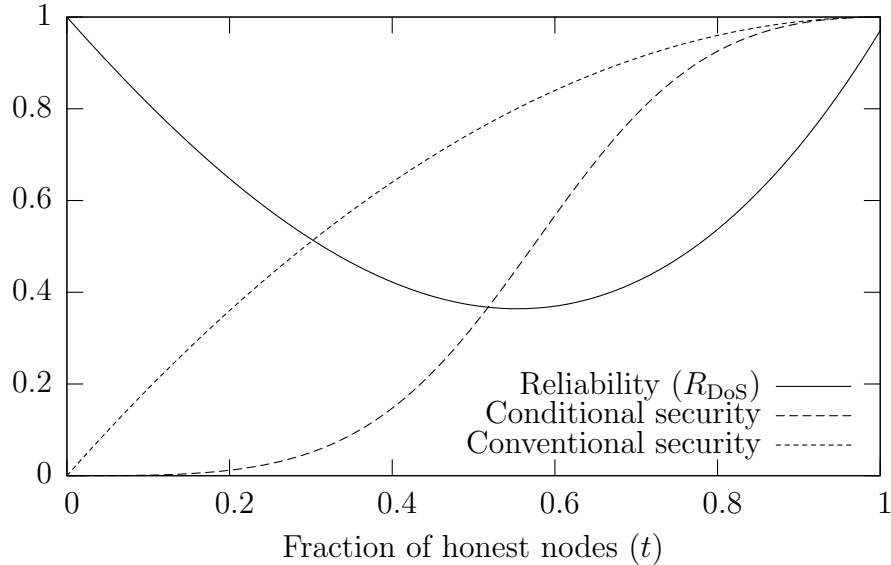


Figure 5.1: Reliability and security analysis of Tor under the selective DoS attack, with  $f = 0.99$ .

that point, the  $(1 - t)^2$  component starts to dominate; that is, the dishonest nodes start to perform DoS on fewer tunnels because they can now compromise more of them.

The second line in Figure 5.1 is the number of secure tunnels, as a fraction of reliable ones; i.e. the conditional probability of a tunnel being secure given that it is reliable. This is a useful calculation, since a Tor user (actually, the Tor software), faced with a non-functioning tunnel, will create a new one in its place, and will repeat this until a working tunnel is constructed; the conditional probability states how likely it is that this final tunnel will be secure. For high values of  $t$ , the line closely matches the conventional security figure of  $t^2$ , but with higher numbers of compromised nodes it quickly diverges. For example, with  $t = 0.5$ , conventional analysis suggests that 75% of all paths should be secure, whereas under the selective-DoS attack, only 33% of the successful paths are uncompromised.

Of course, one hopes that fewer than 50% of Tor routers are dishonest — it would seem difficult for an adversary to compromise 400 out of 800 routers. However, Tor is a volunteer-run network that accepts new routers with minimal verification; it is therefore not out of the question for some organization to contribute many new routers to the system, under different

identities, and compromise a significant percentage of routers.<sup>2</sup> The important point is that the conventional analysis of Tor security significantly underestimates the vulnerability of Tor in this scenario.

### 5.1.2 Vanilla Mix Networks

We next discuss the application of the selective DoS attack to high-latency systems based on mix networks, such as the MixMaster [21] and MixMinion [10] networks, used for sending anonymous email.

#### Background

We refer to the original Mix-Net proposal [6] as *Vanilla mixes*. Vanilla mixes were further developed in Babel [18], and the Mixmaster [21] and Mixminion [10] systems were later deployed. These systems can be classified as high-latency Mix-Nets since they introduce large and variable latencies during batching. Because of this, however, high-latency Mix-Nets are much more robust to timing attacks than systems such as Tor. Only when the adversary controls every mix in the forwarding path will the anonymity of a message be compromised.

#### Reliability

As mix-based systems were developed and deployed, the issue of reliability became apparent as volunteer-run nodes were often unavailable or offline. One approach was to introduce *pingers* to keep track of the reliability of nodes. Pingers attempt to relay traffic through all mixes and keep track of the messages that are eventually delivered. To avoid malicious nodes manipulating the rankings, the pinging traffic is forwarded through the anonymous network itself. Mix clients then select routes based on these reliability rankings. Such a

---

<sup>2</sup>In fact, a smaller scale version of such a Sybil attack [14] has recently been observed on the Tor network.

Variable	Description
$l$	The <i>length</i> of all paths. We assume all copies of the message travel over paths of the same length.
$w$	(for <i>width</i> ) The number of independent paths over which a copy of the message is transmitted.
$t$	The probability a mix is honest. Its converse $\bar{t} = 1 - t$ is the probability a node is in the hands of the adversary. We assume that all nodes when chosen have the same probability of being corrupt, independently of the number of previously honest or corrupt nodes selected.
$f$	The probability an honest node is reliable. Its converse $\bar{f} = 1 - f$ is the probability it is unreliable. This does not apply to corrupt nodes, which are reliable or not depending on the attack strategy – a reliable node relays the message correctly, while an unreliable one is simply offline, and behaves as if it does not exist in the network.

Table 5.1: Variables used in reliability and security analysis

strategy, however, biases the nodes used, may be manipulable by the adversary, and could reduce the anonymity of messages.

An alternative strategy to ensure reliability, used by Mixminion and Mixmaster, is to send copies of the messages, or fragments thereof, through independent paths. There is no interaction between copies of the message traveling on the multiple paths, and the mixes on the different paths operate independently. Furthermore, messages on different paths are bitwise unlinkable amongst themselves, reducing the potential for traffic analysis.

## Conventional Analysis

We first analyze the security and reliability of Vanilla mixes under the simple attacker strategy of forwarding all messages. Our results are straightforward and well-known; we present them here for comparison purposes only. We assume that reliability is achieved by sending multiple copies of the messages, as described above. We introduce several parameters for describing the mix network in Table 5.1; we will use them throughout the rest of the paper.

**Theorem 1** (Security of Vanilla Mixes). *In a Vanilla mix network with parameters  $(l, w, t, f)$ ,*

where adversary nodes simply relay all communications, the probability the message is delivered anonymously is:

$$[1 - (1 - t)^l]^w \tag{5.1}$$

*Sketch.* For the message to be compromised, at least one full route should be composed of dishonest mixes. A full route is honest with probability  $1 - (1 - t)^l$ . The probability all routes are honest gives us the above result.  $\square$

The security of mix networks is significantly higher than low-latency systems, since increasing  $l$  results in an exponential increase of security. For example, with  $l = 5$  (the default used by MixMinion), even if 50% of all mixes are compromised, only 3% of all messages can be read. Cautious users may choose even higher values of  $l$ , so that their messages remain secure even under the most pessimistic assumptions about the number of compromised mixes.

**Theorem 2** (Reliability of Vanilla Mixes). *In a Vanilla mix network with parameters  $(l, w, t, f)$ , where adversary nodes simply relay all communications, the probability the message is correctly delivered is:*

$$1 - [1 - (\bar{t} + t \cdot f)^l]^w \tag{5.2}$$

*Sketch.* For the messages to be delivered there should be at least one full route composed of bad or reliable honest mixes. The probability a single route is such is  $(\bar{t} + t \cdot f)^l$ , and the probability not all routes are unreliable is the probability we seek.  $\square$

### Selective DoS Attack

Similar to the case in Tor, an adversary may choose to apply a selective DoS strategy to maximize the chances of compromising messages. Instead of relaying all messages, bad mixes only relay those messages that they can trace from the beginning to end: the mixes

decrypt as much of the message as they can using the keys of all the colluding mixes and determine whether there is an honest mix somewhere in the chain. Messages that cannot be compromised in this way are either dropped or modified in a subtle way so that they are unrecoverable by the recipient. The sender then has to send more copies of the message to increase its chances of arriving, which in turn increases the chances that the adversary captures the message.

**Theorem 3** (Reliability of Vanilla Mixes against DoS). *In a Vanilla mix network with parameters  $(l, w, t, f)$ , where adversary drops all communications they cannot compromise, the probability the message is correctly delivered is:*

$$1 - (1 - [\bar{t}^l + (t \cdot f)^l])^w \quad (5.3)$$

*Sketch.* For the message to be delivered a path has to be either fully compromised or fully honest and reliable. This occurs with probability  $r = (1 - t)^l + (t \cdot f)^l$ . At least one such path within  $w$  must be picked which happens with probability  $1 - (1 - r)^w$ .  $\square$

The DoS strategy does not affect the probability the message is secure; the results are the same as in Theorem 1. So what advantage does this strategy present? To achieve the same level of reliability, a sender must send the messages more times, which in turn provides more opportunities for the adversary to capture the message. How many more copies of the message should be sent, though?

**Note 1** (Same reliability for plain and DoS). *Given a mix network with parameters  $(l, w_{van}, t, f)$  with a plain adversary leading to messages having a probability of delivery  $p_{van}$ , the number of copies of a message in a network with parameters  $(l, w_{DoS}, t, f)$  with a DoS adversary to achieve the same degree of reliability is:*

$$w_{DoS} = \frac{\log(1 - (\bar{t} + t \cdot f)^l)}{\log(1 - (\bar{t}^l + (t \cdot f)^l))} w_{van} \quad (5.4)$$

*Sketch.* Require the two probabilities of reliable delivery from Theorems 2 and 3 (with  $w$  equal to  $w_{\text{van}}$  and  $w_{\text{DoS}}$ , respectively) be equal and solve for  $w_{\text{DoS}}$ .  $\square$

**Note 2** (Probability of security for a target reliability.). *The probability of security for a particular target reliability  $c$  can easily be calculated:*

$$sec_{\text{vanilla}} = (1 - \bar{t}^l)^{\frac{\log(1-c)}{\log(1-(\bar{t}+t \cdot f)^l)}} \quad (5.5)$$

$$sec_{\text{DoS}} = (1 - \bar{t}^l)^{\frac{\log(1-c)}{\log(1-(\bar{t}^l + (t \cdot f)^l))}} \quad (5.6)$$

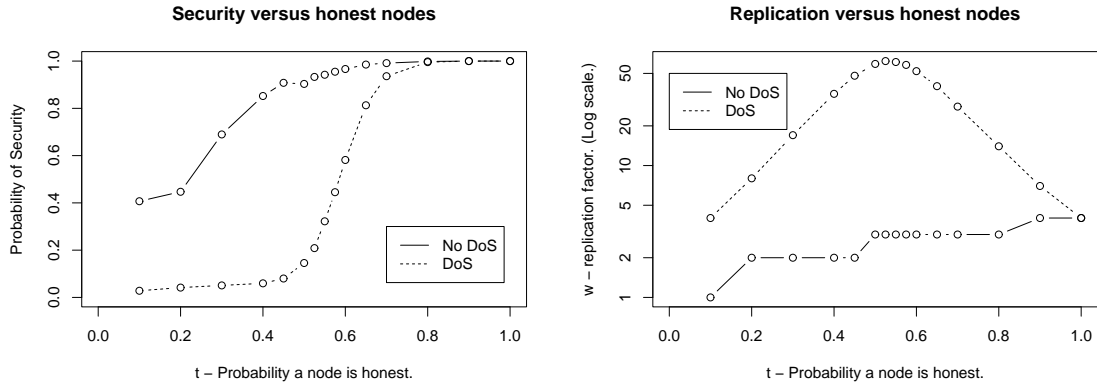
Figure 5.2 presents the results of simulations used to validate our calculations. Nodes in these experiments tuned their sending parameters  $w$  to achieve a reliability of 95%, with fixed  $l = 5$  and  $f = 0.90$ . The parameters  $l$  and  $f$  are chosen to mimic an observed behavior of Mixminion nodes.<sup>3</sup> Figure 5.2(a) presents the fraction of secure messages out of 5000 sample messages sent through the network, for both the passive and DoS attacker strategies (the sample size leads to an error of less than 1% for all our simulation results; the discontinuities in the graph are due to changing values of  $w$ ). It is clear that an attacker who denies service has an advantage, depicted as the gap between the two lines representing the probabilities of success of the two attacks respectively.

Figure 5.2(b) depicts the replication factor  $w$  that honest nodes are forced to use to compensate for network unreliability. Under the DoS strategy, the number of copies of a message can become very large. The number of replicas peaks at about  $t = 0.5$ . For smaller fractions of honest nodes, reliability is guaranteed by the fact that the route is often compromised, and for larger fractions, reliability is restored by the honest nodes. This should act as a clear warning to mix system administrators: reliability is not a measure of security.

The graph representing  $w_{\text{DoS}}$  has a maximum that can easily be calculated. It is interesting that the position of this maximum, which indicates the fraction of nodes that would

---

<sup>3</sup>This is lower than the figure we used for reliability in Tor; Tor nodes are periodically probed by the directory server and ejected from the network if they are not reliable.



(a) Security for different fractions of honest nodes. (b) Number of messages to be sent for different fractions of honest nodes.

Figure 5.2: The effect of different fractions of honest nodes  $t$  on the security of the routes, and the replication factor  $w$ , for a target reliability of 95%. Experimental results: 5000 samples per point,  $l = 5$  (Mixminion),  $f = 0.90$ .

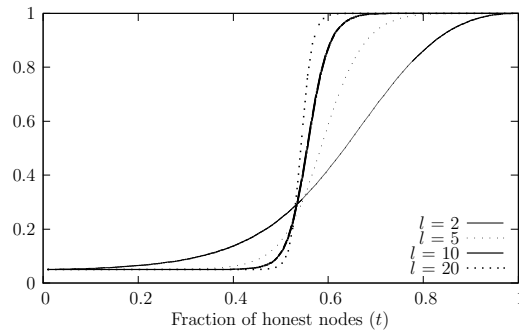


Figure 5.3: Security of Vanilla mixes for different choices of  $l$  under the DoS strategy.

cause a maximal replication, is independent of the reliability achieved. We calculate this analytically.

**Note 3** (Parameter  $t$  for maximal  $w_{\text{DoS}}$ ). *The fraction of honest nodes  $t$  for which the replication factor  $w_{\text{DoS}}$  is maximal is:*

$$t = \frac{1}{f^{l/(l-1)} + 1} \quad (5.7)$$

*Sketch.* We require the derivative  $\frac{dw_{\text{DoS}}}{dt} = 0$ , and solve for  $t$ . □

## Increasing path lengths ( $l$ )

One response to increase security under the DoS attack may be to use longer paths. Conventional analysis suggests that higher values of  $l$  provide exponentially higher security, so nearly arbitrary security levels can easily be achieved by increasing  $l$ . Can the same approach work under the DoS strategy?

Figure 5.3 shows the security achieved for varying values of  $l$  using the same parameters as in Figure 5.2 under the DoS strategy. For higher values of  $t$ , increased values of  $l$  have the expected effect of increasing security. However, for low values of  $t$ , longer paths not only do not help the security, but in fact are a detriment. This is because long paths make it easier for adversaries to perform DoS on paths, and the values of  $w$  required to achieve 95% reliability become so high that there are more opportunities for compromise.

The results show that, when reliability, and not just security, is taken into account, Vanilla mixes have a fundamental limit on the number of compromised mixes. When we have a majority of corrupt nodes ( $t \leq 0.5$ ), Vanilla mixes are “unsafe at any path length.”

## 5.2 Denial of Service against Systems Featuring Reliability

### 5.2.1 Cashmere

#### Background

Cashmere [37] is an anonymous routing layer that uses *relay groups* instead of single-node mixes to provide increased connection reliability.

Each relay group is composed of a set of nodes that share a common public/private key pair. This gives any member of the group the ability to decrypt a layer of the message and forward it to the next relay group. Each node in Cashmere is assigned a unique nodeID and

each relay group a unique groupID such that a node is a member of a relay group if the groupID is a prefix of its nodeID.

Cashmere is implemented on top of the Pastry [31] structured overlay and makes use of its anycast mechanism to route a message toward any node with the correct groupID prefix; the node that receives the message is named the *relay group root*. The Pastry mechanisms for maintaining reliability ensure that such a node will be found as long as at least one member of the relay group is reliable. The root decrypts the message, broadcasts the payload to all members of his relay group, and then sends the message to the next relay group in the forwarding path. Since the payload of the message is encrypted with the destination's public key, independent of the forwarding path, a destination can receive the message before it reaches the end of the relay path.

### Security and Reliability of Cashmere

In the presence of a passive adversary, honest and reliable, as well as dishonest nodes will forward traffic appropriately. As long as there is at least one of these nodes acting as the relay group root for every relay until the destination, the connection will remain reliable. Since any of the nodes in a relay group can decrypt the current layer of the forwarding path, connections may be insecure whenever there is at least one dishonest mix present in every relay group leading up to the destination. Since the destination itself is not revealed in the message, but instead is chosen among the members of all relay groups, we also require the destination to be dishonest to consider a message compromised. In other words, we are measuring sender anonymity. If we consider Cashmere is used as a TCP relay for anonymous traffic, sender anonymity is similar to the linkability we previously discussed for other Mix-Net systems. The analysis of linkability when a Cashmere node is the actual destination is more complicated and beyond the scope of this work.

**Theorem 4** (Reliability of Cashmere). *In Cashmere with parameters  $(l, w, t, f)$  where dishonest nodes simply relay all communications, the probability the message is delivered reliably*

is:

$$\sum_{i=0}^{l-1} \frac{1}{l} (1 - (t\bar{f})^w)^i (\bar{t} + tf) \quad (5.8)$$

*Sketch.* The probability that a relay group is reliable, or has at least one honest, reliable or dishonest node present, is  $1 - (t\bar{f})^w$ . To ensure message reliability, each relay group before the one that contains the destination must be reliable, and the destination must itself be reliable. The destination is in each relay group with probability  $1/l$ .  $\square$

**Theorem 5** (Security of Cashmere). *In Cashmere with parameters  $(l, w, t, f)$  where dishonest nodes simply relay all communications, the probability the message is delivered anonymously is:*

$$\sum_{i=0}^{l-1} \frac{1}{l} (1 - t^w)^i \bar{t} \quad (5.9)$$

*Sketch.* For message anonymity to be compromised, each relay group leading up the destination, plus the destination itself must be compromised. The probability that at least one node in the relay group is malicious is  $1 - t^w$ , as must be the case for every relay group leading up the destination. Finally, the destination itself must be malicious for the route to be compromised.  $\square$

### Cashmere under a DoS adversary

Cashmere routing is affected by DoS attacks when any of the relay group roots are dishonest. Unless the adversary has compromised the entire forwarding path, he will drop any connection that goes through a relay root he controls and the connection will have to be rebuilt.

**Theorem 6** (Reliability of Cashmere under DoS). *In Cashmere with parameters  $(l, w, t, f)$  where dishonest nodes drop all communications they cannot compromise, the probability the*

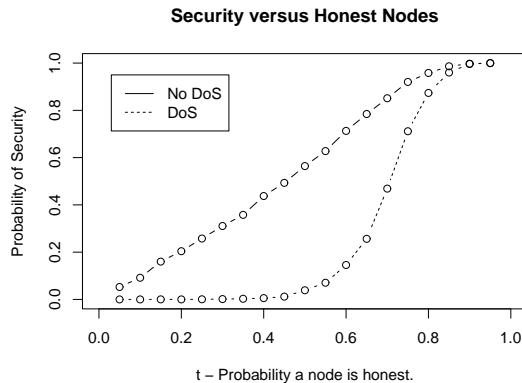


Figure 5.4: The security of Cashmere under different fractions of honest nodes with an expected reliability of 100%. Experimental results: 5000 samples per point,  $l = 5$ , and  $f = .90$ .

message is delivered reliably is:

$$\sum_{i=0}^{l-1} \frac{1}{l} \left( (1 - t^w)^i \bar{t} + \left( \frac{tf}{tf + \bar{t}} \right)^i tf \right) \quad (5.10)$$

*Sketch.* The probability that a message is delivered reliably is the probability that every relay group until the destination delivers the message reliably. This means that either every relay root and the destination are reliable and honest, or the entire path is compromised and thus remains reliable. A relay root is chosen out of only reliable nodes, so it is reliable and honest with probability  $tf/(tf + \bar{t})$ .  $\square$

Figure 5.4 presents the results of Cashmere simulations under a passive and DoS attacker. In these experiments, honest nodes had  $f = .90$ , connection lengths were  $l = 5$ , and the group size  $w = 5$ , as recommended by the Cashmere authors. This setup produces nearly 100% reliability under a passive adversary. Note that it is impossible to increase reliability under the DoS strategy by increasing  $w$ , since the adversary need only to capture the root node to block the message. The graph depicts the fraction of successful connections that remain secure; the DoS strategy is very effective at reducing this number quickly.

These results highlight that under the DoS strategy, both the reliability and security of

Cashmere are strictly worse than for Vanilla mixes with equivalent  $w$ . This is because to deny service, the adversary must capture only a single relay root that precedes the destination, whereas Vanilla mixes require an adversary on every path. Similarly, to violate security, the adversary needs only a single adversary in each group, rather than an entire compromised path in Vanilla mixes. By failing to consider denial of service as a security concern, the authors have created great potential for the selective DoS attack to succeed.

The graph also shows that, under the parameters we considered, Vanilla mixes offer substantially greater security even under the simple adversary strategy. Cashmere *is* useful when there are few compromised nodes and very frequent failures. With  $t = 1.0$  and  $f = 0.5$ , the reliability of Cashmere with  $w = 5$  is 94%, conditioned on the destination being reliable. To achieve the same reliability in Vanilla mixes,  $w = 89$  would be necessary!

## 5.2.2 Hydra-Onions

### Background

The Hydra-Onion system was designed to resist active adversaries dropping onions during transmission. [19] Just as in vanilla mixes,  $w$  copies<sup>4</sup> of a Hydra-Onion are sent in a cascade. However, at each step, a mix will forward two copies of the Hydra-Onion to two different mix servers at the next step.

Each onion has the following format:

$$\begin{aligned}
 O_i = \{ & Enc_{J_{i,1}}(J_{i+1,1}, J_{i+1,a(1)}, k_{i+1}), \\
 & Enc_{J_{i,2}}(J_{i+1,2}, J_{i+1,a(2)}, k_{i+1}), \\
 & \dots, \\
 & SEnc_{k_{i+1}}(O_{i+1}) \}
 \end{aligned}$$

---

<sup>4</sup>The authors of the Hydra-Onion system call this parameter  $k$ ; however, we use  $w$  for consistent presentation.

where  $J_{i,j}$  are the identities of the mixes at step  $i$ , and  $a()$  is a permutation of nodes with  $a(i) \neq i$  for all  $i$ . In this case,  $Enc_J$  is the hybrid asymmetric/symmetric encryption under the public key of  $J$  and  $SEnc_k$  is the symmetric encryption under key  $k$ . Each mix server decrypts the piece of the onion encrypted under its key and learns the identities of two servers in the next step as well as the symmetric decryption key for the next layer of the onion. This communication pattern is displayed in Figure 5.7(a).

### Security and Reliability of Hydra-Onions

Since any of the mixes at step  $i$  can decrypt the Hydra-Onion  $O_i$ , a Hydra-Onion is insecure whenever there is at least one dishonest mix at each step:

**Theorem 7** (Security of Hydra-Onions). *Given a Hydra-Onion mix with parameters  $(l, w, t, f)$ , the probability that a message is secure is:*

$$1 - (1 - t^w)^l \tag{5.11}$$

The reliability of Hydra-Onions is somewhat harder to ascertain, due to the randomized forwarding nature of the mixes. The intuition behind the design is that random graphs are expanders, and therefore, a single Hydra-Onion will quickly replicate to fill the  $w - 1$  missing ones.

To evaluate Hydra-Onion reliability, we have developed a simulation of the scheme. We create a network of  $l$  by  $w$  mixes, and connect the nodes using randomly generated permutations  $a$ . Each of the nodes is assigned to be honest with probability  $t$  and, if honest, to be reliable with probability  $f$ .

In the case of simple attacker strategy, we assume that dishonest nodes are always reliable. To simulate the reliability given the DoS attacker strategy, we first determine whether there is at least one dishonest mix at each step. In this case, the onion is compromised and

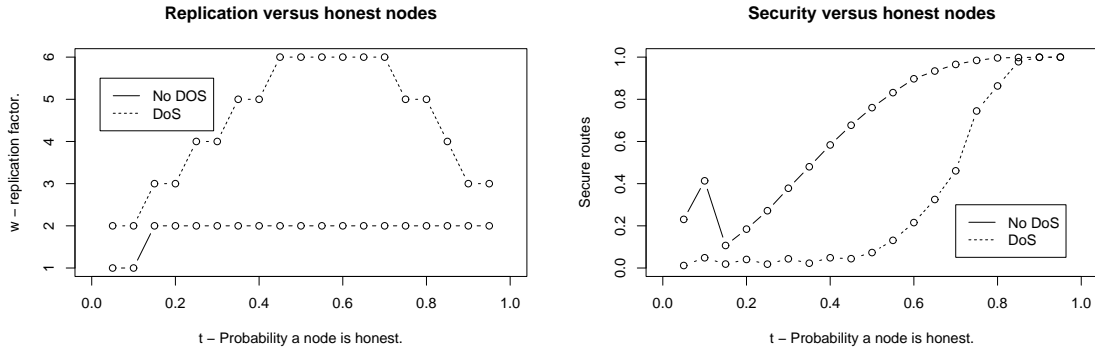


Figure 5.5: Replication factor  $w$  that achieves reliability of 95% for Hydra-Onions under different fraction of honest nodes  $t$ , and the corresponding security. Analytical results.  $l = 5, f = 0.90$ .

the dishonest nodes are reliable and forward all messages. Otherwise, the dishonest nodes perform a denial of service and drop all traffic sent to them.

## Analysis

Figure 5.5 shows the analysis of Hydra-Onions under both the simple adversary strategy and the DoS strategy. Setting  $f = 0.9, l = 5$  and varying  $t$ , we increased  $w$  until we could obtain 95% reliability. The figures plot the  $w$  required to achieve this reliability, as well as the security at that  $w$ . The sample error is not shown since it is less than 1%, and the lack of monotonicity under low values of  $t$  is due to the changing and quantised values chosen for  $w$ .

As designed, Hydra-Onions are effective at providing reliability in the face of denial-of-service: even under heavy denial of service,  $w = 6$  suffices to achieve 95% reliability. However, this is done at the expense of security: increasing values of  $w$  very quickly decrease the security of Hydra-Onions, as a single compromised mix at each step suffices to compromise the entire onion. When 15% of nodes are malicious, 5% of all onions are compromised, and when the fraction of malicious nodes rises to 30%, over half of all paths are compromised. 30%, and even 15%, may sound like high fractions of attackers, but recall

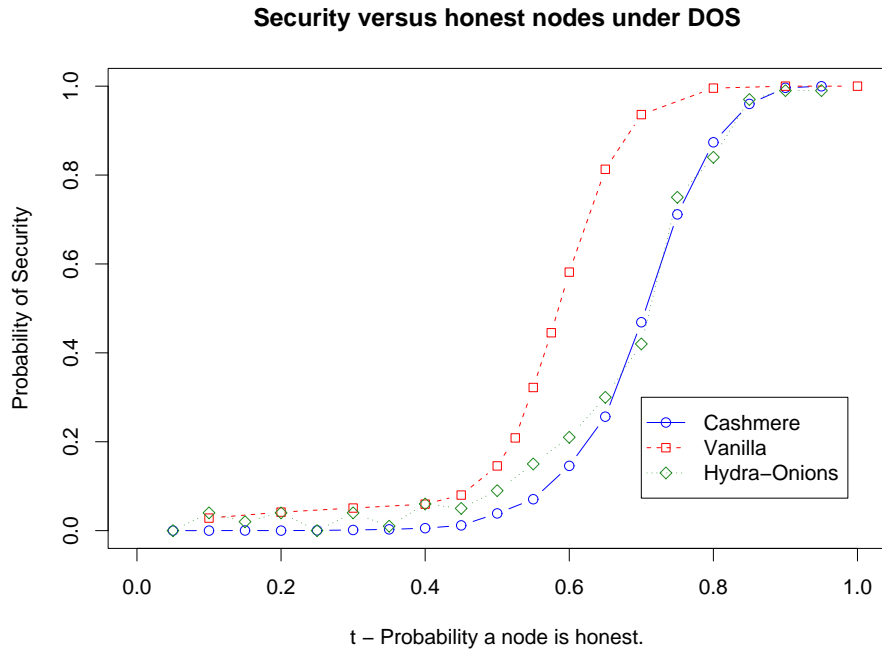


Figure 5.6: Comparing the security of Vanilla mixes, Cashmere, and Hydra-Onions under DOS.

that the conventional analysis of a mix network with 5 hops that ignores DoS suggests that path compromise occurs with probability 0.25% and 0.008% respectively with these fractions of attackers. For comparison, Vanilla mixes with 30% attackers are able to achieve 93.6% security, albeit with a width of 28.

Therefore, Hydra-Onions are not a good tool when a significant number of mixes are compromised. As can be seen from the right limit of the graph, they are also inefficient when nearly all nodes are honest: with 95% honest nodes, Hydra-Onions use a width of 3 in the DoS strategy, which has a communications cost equivalent to a Vanilla mix width of 6, since each mix sends two onions. Vanilla mixes under the same parameters require  $w = 5$  to achieve 95% reliability, and provide better security. (Though both schemes achieve over 99.99% security with these parameters.) For comparison, the security of Vanilla mixes, Cashmere, and Hydra-Onions is presented in Figure 5.6.

The main advantage of Hydra-Onions seems to be when most nodes are honest, but not

reliable (either due to inherent reliability problems or external DoS attacks.) For example, with  $f = 0.5$ , Vanilla mixes require  $w = 95$  to achieve 95% reliability, even when no nodes are compromised, whereas Hydra-Onions only require  $w = 19$  in the same situation. However, a simpler variant of Hydra-Onions proposed by the same authors, called DUO-Onions, may be more appropriate for this case. DUO-Onions [19] are designed to handle fail-stop failures, such as unreliable nodes or DoS, by iteratively picking the next mix in a list whenever the first choice is unreachable. DUO-Onions have the advantage of using dramatically less bandwidth in the case that nodes *are* reliable, and only sending extra onions as necessitated by failures. They are unable to address Byzantine faults of the forwarding mixes, but as our analysis shows, neither are Hydra-Onions.

## 5.3 Attacks

So far we have examined the security and reliability of Vanilla mixes, as well as mixes that were designed with reliability in mind, against passive and denial of service adversaries. In both cases, we have assumed that a message is compromised, only if the adversary has the cryptographic material necessary to uncover its route from beginning to end. In practice, this means controlling a node at each stage of the message’s path. The DoS attack strategy simply increases the odds of this being the case.

In this section we show that in robust mix systems the anonymity of messages may be greatly reduced even if the full path of a message is not controlled by malicious nodes. The robustness mechanisms introduced are indeed susceptible to more complex traffic analysis attacks than the simple Vanilla mixes.

### 5.3.1 Replay attacks against Hydra-Onions and Cashmere

Traditional mix systems such as Mixmaster and Mixminion prevent messages from being relayed multiple times by a single mix (Type I “cypherpunk” remailers do not, however).

This makes traffic analysis more difficult, since an adversary is unable to get multiple measurements to extract the correspondence between the input message and the resulting output message.

The mechanisms by which replay prevention is achieved are security critical, and are the only ones in a mix that requires keeping state. To minimize the space required to keep this state, a mixture of hashing to compress signatures, timestamps to make messages expire, and key rotation to prevent valid decryption are used. The design of replay prevention must also protect against a rather powerful adversary: an adversary submitting any function of a message to the mix should gain no information as to the destination of the original message. Given an adversary with such powers, replay attacks become a simple case of the more generic tagging attacks, where modified messages are injected into the network to extract information about their destination.

As a result of the attackers capabilities, naive approaches, such as keeping a list of the hash of the messages processed is not sufficient. Modifying a single bit of the input would produce a different hash, bypassing the replay prevention mechanism. This in turn may lead to the message being processed and to leaking information, unless a separate integrity checking mechanism is implemented. For technical reasons, integrity checking messages is hard in mix systems that support replies.

The usual heuristic for implementing replay prevention is to keep a hash, or another one way derivative, of the symmetric key used to decrypt the bulk of the mix message, and use a non-malleable (or full CCA2 secure) asymmetric scheme to transport those.

Neither the Hydra-Onion scheme<sup>5</sup>, nor Cashmere, consider the problem of replay prevention, which becomes much more difficult when distributing mixes across multiple nodes. They both provide a very high level description of possible packet formats to use, and Cashmere claims that the choice of the packet format is orthogonal to the routing mechanisms

---

<sup>5</sup>The authors of Hydra-Onions consider the problem of replay prevention in [17]. Sadly, their construction turn out to be insecure [9].

described. Yet, both schemes describe a hybrid packet format, with symmetric keys  $K_i$ , being encrypted using the public asymmetric key of each mix in Hydra-Onions, or the public key of the whole group in Cashmere. Applying our heuristic, nodes may be tempted to simply keep a list of hashes of the processed messages' keys (i.e.  $H_{\text{replay}}(k_i)$ ), and check new messages against it. Sadly, this approach is open to attacks.

Lets assume that the naive, off-the-shelf approach is used to secure Hydra-Onion mixes or Cashmere against replay attacks. In the case of Hydra-Onions, an adversary will try to trace a message he holds through a subsequent set of nodes that are all honest. In case there is one that is dishonest, there is no need to apply traffic analysis, since the mixing stage is transparent to the adversary. To do this, he seeks a node from the next stage of mixing that is offline at the time the message is meant to leave. This node will not be processing the message, and will therefore not have updated its list of processed messages accordingly. The adversary then waits for the node to be available again and sends the message through it. This results in the message (or a variant thereof) to be relayed a second time through the mix, allowing a potential adversary node in the next group to trace the message. A similar attack can be deployed against Cashmere: an adversary observes which nodes in the relaying group were unavailable during the initial transmission, and replays the message through them. If the message is seen twice by any adversary node in the next group, it is traced.

These two attacks illustrate that the naive local storage of messages processed by each mix does not provide a similar degree of replay prevention as traditional mixes. A robust replay prevention scheme would need to determine some sort of consensus about whether an onion has already been forwarded among all other mixes who may have seen it, yet such consensus is both difficult to achieve without sacrificing reliability and in the presence of malicious nodes.

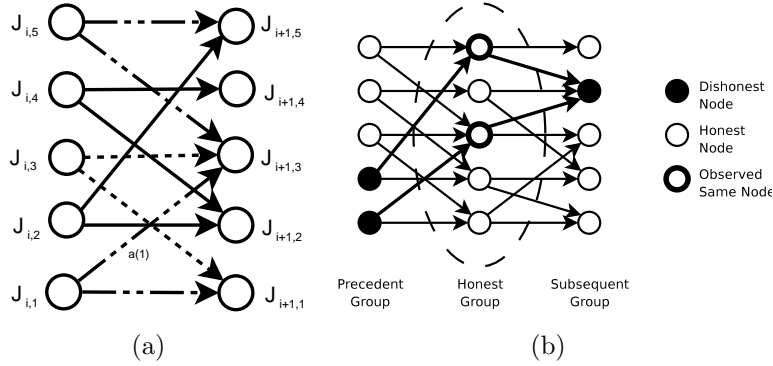


Figure 5.7: (a) Hydra-Onion communication patterns and (b) Illustration of the Hydra-Onion attack.

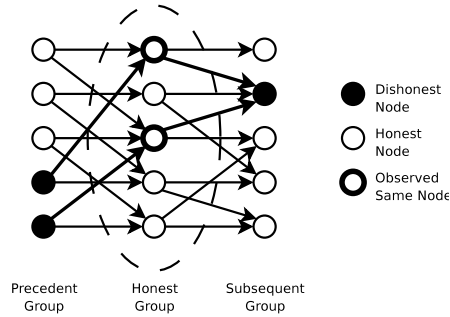


Figure 5.8: Illustration of the Hydra-Onion attack.

### 5.3.2 Bridging Honest Hydra-Onion Groups

In the previous sections we have assumed that if a Hydra-Onion group (of size  $w$ ) is honest, meaning that it is only composed of honest nodes, the mixing step is performed securely, and the message’s anonymity is guaranteed. It turns out that this assumption is simplistic, and further traffic analysis attacks can be used by adversary nodes to “bridge” such honest groups and trace messages relayed through them.

The attack relies on an honest group, composed only of honest nodes, being preceded and succeeded by dishonest groups, containing at least one dishonest node. The aim of the adversary in this case is to trace the message being relayed through the honest group, and link it to the corresponding output message. To achieve this, the attacker nodes in the preceding group observe the identity of some mixes involved in the honest group. The

number of nodes from the honest group that can be identified depends on the number of unique nodes of the honest group that they are intending on sending the message to. It is not absolutely necessary for the dishonest nodes to all be receiving the message (all the nodes that were meant to be forwarding it may be unreliable.) Even if only one dishonest node in the preceding group receives the message, it can then be forwarded to all other dishonest nodes.

The dishonest nodes in the subsequent group also observe a certain number of the same honest nodes emitting a message with the same bit-pattern. Again, the number of nodes from the honest group observed depends on the number of nodes in it that send to dishonest nodes<sup>6</sup>.

Depending on the number of nodes in the honest group that have been identified by the preceding and subsequent dishonest nodes, the adversary can be assured with some probability that the message corresponds to the input message. Given that a set of  $\xi$  common honest nodes are being identified as the destination of the message to be traced, as well as the source of an outgoing message, what is the the probability the outgoing message corresponds to the message?

We have to make assumptions about the absolute numbers of honest nodes in the network, say  $N$  and the volume of secure messages in the network, say  $v$ . The probability that an observed set of  $\xi$  relaying nodes is unique in the network is at least:

A key observation is that for fixed values of other parameters as the size of the network  $N$  grows, the probability of uniqueness, and hence the probability of avoiding false positives when performing the attack, goes to one. Hence, against this attack, larger networks are more vulnerable than small ones. For the special case of  $\xi = 1$ , it is intuitive that the probability of success of this attack is rather slim:  $(1 - w/N)^{v\xi}$ . As the amount of traffic  $v$  in the network increases, the probability of uniqueness quickly drops to zero.

---

<sup>6</sup>We assume throughout that link encryption is used, and therefore a message between two honest nodes cannot be used for this attack.

In the case that other messages are also using the same set  $\xi$  of relays as the message being traced, the quality of the anonymity provided is still extremely reduced. The expected size of the anonymity set the output message benefits from is:

### 5.3.3 Reducing the Anonymity Sets in Cashmere

Cashmere is based on a distributed hash table architecture, namely *Pastry* [32], and heavily uses its features to route messages. In particular, relay groups are defined as sets of nodes sharing the same nodeID prefix, and the “anycast” routing from relay group to relay group is a natural byproduct of Pastry routing. Each Pastry node keeps a record of some other nodes distributed in a particular way around the network, and routes messages based on a longest common prefix strategy. Eventually messages should converge toward a node with the appropriate prefix.

A key weakness of Pastry routing, when it comes to overlaying an anonymity system on it, is that the node receiving the message routed using anycast, is not random. It is determined by the identity of the sender of the message. This is the result of routing tables being persistent and public (or very easily recoverable by traffic analysis), and the routing strategy being deterministic.

An adversary can use the above property to degrade the quality of the anonymity provided by the system. A message relayed by a particular target node may only have arrived from another node that can route toward the target node. We can assume that each node in the network uses a random (but persistent and observable for the adversary) node for each group to route to, so we expect that the number of nodes that can route to any given node to be the total number of nodes in the network (aside from the nodes in the group itself) divided by the number of nodes in the group:

$$\frac{N - w}{w} \tag{5.12}$$

This is only the expected number of possible senders for each node in a group. It is also possible, and highly likely, that some nodes in a group have more, or more interestingly from the adversary point of view, fewer potential senders. The reduction in anonymity sets can only be used in conjunction with other attacks, since by itself, a sparse mix network is still secure even for small path lengths [8]. A closer look at how Pastry routing is determined, and attacks at the Pastry level to capture routes, might lead to further weaknesses.

# 6 Defenses

In this Chapter, we explore how reputation management and Byzantine fault tolerance algorithms can be applied to anonymous communication systems to defend against path compromise strategies.

## 6.1 Path Compromise and Reputation

Biasing tunnel construction is just one method a participating adversary can use to break anonymity, yet as we have seen, an effective one. The adversary tries to make itself look more attractive to be selected as a relay mix, so a reliable reputation mechanism would be an effective counter to this strategy. Such a mechanism would allow us to (1) maintain history on node behavior to identify dishonest activity and (2) distribute this information to other mixes in the system. Two major attempts have been made at reputation management in anonymous communications. In [11], Dingledine et al. present a reputation system for Mix-Net remailers that uses receipts from relay mixes to determine correct or incorrect behavior. Receipt validation contributes to mix reputation. Unfortunately, reputation management is performed by globally trusted witnesses, which limits the scalability of the system and introduces a single-point failure weakness inherent when only a small number of users manage reputation.

The MorphMix CDM had a similar purpose to that of reputation management: detect dishonest mixes and exclude them from relay paths. MorphMix instead attempted to use local view information to address the shortcomings of using a small set of trusted authorities.

However, as we demonstrated in the case of MorphMix, attackers may be able to model a victim’s local knowledge and manipulate its content to prevent detection.

An important aspect of reputation management is verifying mix capabilities. During path construction, many system protocols have been designed to take into account the bandwidth capacity and current mix load before selecting mixes to use in the relay path. This allows systems to provide both reliable and efficient networking. For example, Tor and MorphMix users bias mix selection based on mixes with high bandwidth capabilities, yet this information is supplied directly from the mix and never verified.

When an adversary is able to make itself look more attractive to users trying to create relay paths, it is able to bias its inclusion in anonymous tunnels more so than it would by random selection alone. This is the case if a user is allowed to provide self-supplied characteristics, such as its resource capabilities, or improves its reputation based on false merit. In the case of the attack presented in Chapter 4, we saw that malicious nodes were able to make themselves look more attractive by providing specially crafted “honest-looking” selections that not only fool a victim into thinking they are honest, but additionally lower the collusion threshold such that honest nodes are falsely detected as being dishonest.

### **6.1.1 Path Compromise Detection and Prevention**

To better prevent path compromise attacks, measures need to be taken to disallow false representation. Ideally, we could do global verification of mix capabilities and reputation. Unfortunately, a single trusted party weakens the security of an anonymous network because it becomes one point of failure for the entire system. Because of this, systems like MorphMix have attempted to develop local view techniques as soft mix verification. In Chapter 4, we uncovered weaknesses in this technique that lead us to believe purely local views are not enough to prevent an adversary from altering a victim’s view, especially if the adversary that has compromised a substantial portion of the network.

The attack presented in Chapter 5 does not bias mix selection during any single relay

path construction, however, because the adversary forces a user to reconstruct the path until he has compromised the full route, he biases long-term relay path construction. In this case, we need a mechanism to detect such types of dishonest behavior and a protocol to disseminate this information reliably to the rest of the mixes in the network. The difficulty in this is first, being able to identify malicious behavior and second, knowing which nodes to believe concerning other mixes in the network.

For certain systems, we may be able to add an acknowledgement protocol to identify purposeful connection dropping. For example in Cashmere, the relay group root can drop a message and break a tunnel without detection since other mixes can't identify whether the message was dropped in a previous relay group or dropped by the root. Instead, if each relay group required acknowledgments from multiple mixes in the subsequent relay group, it could be assured that the message was broadcast to at least some set of mixes in the next group. Otherwise it could rebroadcast the message until it has either received a sufficient number of acknowledgments or has reason to believe there is suspicious behavior taking place. Additionally, using a group of trusted servers to monitor statistics of failed connections may shed light on repeated misbehaviors and allow for protocols based on reliable mix reputations. Again though, incorporating "trusted" servers into the design of anonymous communication systems is highly avoided, and it is uncertain if reputation management schemes can be applied to anonymous systems without disrupting mix anonymity and user unlinkability within the system.

## 6.2 BFT and Anonymous Communications

In the absence of reputation management, we might hope to apply Byzantine fault tolerance (BFT) techniques to counter the Byzantine participant. BFT algorithms attempt to solve the Byzantine General's problem [20], which describes a group of Byzantine army generals trying to agree upon a military strategy. Each of the generals is separated by location

and must communicate by messenger, which complicates the decision making process. In particular, there may be treacherous generals that arbitrarily lie to trick the other general's into entering a strategy that is not consistent with the group's desires. BFT systems are able to reach consensus independent of the presence of Byzantine faulty components. It has been shown that more than  $\frac{2}{3}$  of the participants in a system must be non-faulty for BFT techniques to hold.

BFT anonymous communication systems should be able to reach consensus on whether a user is dishonest or not to limit his impact on the network. Unfortunately, there are two major limitations of BFT algorithms that complicate their application to anonymous communication systems.

First, BFT techniques require replicated state between servers, however, this reduces the privacy of mixes. BFT algorithms, such as [5], arrive at decisions based on the state of other mixes in the network, yet this knowledge could easily be used by an attacker to deanonymize relay paths. We saw an example of this in the design of Cashmere as relay roots replicated communication state to their entire groups. Since every root broadcast the message to all members of its relay group, the adversary was much more likely to observe the message at each step in the relay path than it was with the vanilla mix design. Hydra-Onions similarly decreased the security of paths by replicating message state to other mixes. By increasing system reliability and providing a potential design for fault tolerance, the authors of these systems inadvertently reduced security. Indeed, there is a conflict between privacy and fault tolerance in the application of current BFT techniques to anonymous communication systems.

Second, most BFT techniques assume closed group membership of BFT groups, but anonymous communication systems typically have dynamic structures. This creates a "chicken and egg" phenomenon when creating groups for Byzantine agreement: If a system was able to initially form a secure BFT group among participants, they could determine the security protocols necessary to maintain fault tolerance, but how does one initially form a BFT group

without first knowing these protocols. In particular, Byzantine faults corrupt node discovery, yet node participation is expected to fluctuate in most existing anonymous communication systems.

# 7 Conclusion

The threat models used in anonymous communication research range from extremely powerful adversaries with full network coverage to arguably more realistic models that assume a portion of compromised mixes. This work specifically focuses on what we define as a participating Byzantine adversary that contributes to the network and acts arbitrarily malicious. Due to the nature of current anonymous network designs, the resources needed for participation are low, so this adversary is a significant threat. Additionally, an attacker that controls mixes in the network can both behave honestly and dishonestly according to the system protocols, making him hard to detect and defend against.

We introduced one strategy that a Byzantine participant may use to disrupt security. By compromising the relay path, an adversary can trivially trace all messages through the network and link a connection initiator to its destination. In all Mix-Net architectures, there is some probability that an attacker will have control over every mix in the relay path, but a Byzantine participant may be able to increase the chances of this happening. We provided two examples of this form of path compromise attack. The first considered an attacker that broke the assumptions of the MorphMix CDM and compromised significantly more paths than the original analysis suggested were possible. The second attack demonstrated how selectively breaking connections and forcing continual path reformation can result in an adversary compromising half of the paths in a system, revealing a fundamental limit on the security of Mix-Net architectures.

Finally, we provided two possible means to counter a Byzantine participant attempting a path compromise attack: reputation management and BFT algorithms. Unfortunately, both

methods are especially difficult to implement in anonymous communications systems and further consideration is needed before an effective defense to this attacker can be realized.

# References

- [1] eDonkey File Sharing System, 2003.
- [2] O. Berthold, H. Federrath, and S. Köpsell. Web MIXes: A System for Anonymous and Unobservable Internet Access. In *Workshop on Design Issues in Anonymity and Unobservability*, pages 115–129, 2000.
- [3] O. Berthold, A. Pfitzmann, and R. 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.
- [4] R. Bhagwan, S. Savage, and G. Voelker. Understanding Availability. In *2nd International Workshop on Peer-to-Peer Systems*, 2003.
- [5] M. Castro and B. Liskov. Practical byzantine fault tolerance. In *OSDI: Symposium on Operating Systems Design and Implementation*. USENIX Association, Co-sponsored by IEEE TCOS and ACM SIGOPS, 1999.
- [6] D. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.
- [7] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptography*, 1(1):65–75, 1988.
- [8] G. Danezis. Mix-networks with restricted routes. In R. Dingledine, editor, *Privacy Enhancing Technologies*, volume 2760 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2003.
- [9] G. Danezis. Breaking four mix-related schemes based on universal re-encryption. In S. K. Katsikas, J. Lopez, M. Backes, S. Gritzalis, and B. Preneel, editors, *ISC*, volume 4176 of *Lecture Notes in Computer Science*, pages 46–59. Springer, 2006.
- [10] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, May 2003.
- [11] R. Dingledine, M. J. Freedman, D. Hopwood, and D. Molnar. A reputation system to increase mix-net reliability. In I. S. Moskowitz, editor, *Information Hiding*, volume 2137 of *Lecture Notes in Computer Science*, pages 126–141. Springer, 2001.

- [12] R. Dingledine, N. Mathewson, and P. F. Syverson. Tor: The Second-Generation Onion Router. In *USENIX Security Symposium*, pages 303–320, 2004.
- [13] R. Dingledine, V. Shmatikov, and P. Syverson. Synchronous batching: From cascades to free routes, 2004.
- [14] J. R. Douceur. The sybil attack. In P. Druschel, M. F. Kaashoek, and A. I. T. Rowstron, editors, *IPTPS*, volume 2429 of *Lecture Notes in Computer Science*, pages 251–260. Springer, 2002.
- [15] Freedman and Morris. Tarzan: A Peer-to-Peer Anonymizing Network Layer. In *SIGSAC: 9th ACM Conference on Computer and Communications Security*. ACM SIGSAC, 2002.
- [16] F. C. Gartner. Byzantine failures and security: Arbitrary is not (always) random. 2003.
- [17] M. Gomulkiewicz, M. Klonowski, and M. Kutylowski. Onions based on universal re-encryption - anonymous communication immune against repetitive attack. In C. H. Lim and M. Yung, editors, *WISA*, volume 3325 of *Lecture Notes in Computer Science*, pages 400–410. Springer, 2004.
- [18] C. Gülcü and G. Tsudik. Mixing E-mail with Babel. In *Network and Distributed Security Symposium — NDSS '96*, pages 2–16, San Diego, California, February 1996. IEEE.
- [19] J. Iwanik, M. Klonowski, and M. Kutylowski. Duo-onions and hydra-onions – failure and adversary resistant onion protocols. In *IFIP TC-6 TC-11 Conference on Communications and Multimedia Security 2004*, September 2004.
- [20] Lamport, Shostak, and Pease. The byzantine generals problem. In *Advances in Ultra-Dependable Distributed Systems*, N. Suri, C. J. Walter, and M. M. Hugue (Eds.), IEEE Computer Society Press. 1995.
- [21] U. Möller, L. Cottrell, P. Palfrader, and L. Sassaman. Mixmaster Protocol — Version 2. Draft, July 2003.
- [22] S. J. Murdoch and G. Danezis. Low-Cost Traffic Analysis of Tor. In *IEEE Symposium on Security and Privacy*, pages 183–195, 2005.
- [23] L. Øverlier and P. Syverson. Locating hidden servers. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*. IEEE CS, May 2006.
- [24] B. Pfitzmann and A. Pfitzmann. How to break the direct RSA-implementation of MIXes. In *Proceedings of EUROCRYPT 1989*. Springer-Verlag, LNCS 434, 1990.
- [25] J.-F. Raymond. Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems. In *Workshop on Design Issues in Anonymity and Unobservability*, pages 10–29, 2000.
- [26] Reiter and Rubin. Crowds: Anonymity for Web Transactions. *ACMTISS: ACM Transactions on Information and System Security*, 1, 1998.

- [27] M. Rennhard. PhD thesis, Swiss Federal Institute of Technology Zurich.
- [28] M. Rennhard. MorphMix prototype v0.1, 2004.
- [29] M. Rennhard and B. Plattner. Introducing MorphMix: Peer-to-Peer Based Anonymous Internet Usage with Collusion Detection. In *WPES*, pages 91–102, 2002.
- [30] M. Rennhard and B. Plattner. Practical Anonymity for the Masses with MorphMix. In *FC: International Conference on Financial Cryptography*. LNCS, Springer-Verlag, 2004.
- [31] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, Nov. 2001.
- [32] A. I. T. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In R. Guerraoui, editor, *Middleware*, volume 2218 of *Lecture Notes in Computer Science*, pages 329–350. Springer, 2001.
- [33] A. Serjantov, R. Dingledine, and P. Syverson. From a trickle to a flood: Active attacks on several mix types. In F. Petitcolas, editor, *Proceedings of Information Hiding Workshop (IH 2002)*. Springer-Verlag, LNCS 2578, October 2002.
- [34] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr. Towards an Analysis of Onion Routing Security. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 96–114. Springer-Verlag, LNCS 2009, July 2000.
- [35] M. Wright, M. Adler, B. N. Levine, and C. Shields. An analysis of the degradation of anonymous protocols. In *Proceedings of the Network and Distributed Security Symposium - NDSS '02*. IEEE, February 2002.
- [36] M. Wright, M. Adler, B. N. Levine, and C. Shields. Defending anonymous communication against passive logging attacks. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, May 2003.
- [37] L. Zhuang, F. Zhou, B. Y. Zhao, and A. Rowstron. Cashmere: Resilient anonymous routing. In *Proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation (NSDI '05)*, Boston, MA, USA, May 2005.