ON TRAFFIC ANALYSIS IN ANONYMOUS COMMUNICATION NETWORKS

A Dissertation

by

YE ZHU

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2006

Major Subject: Computer Engineering

ON TRAFFIC ANALYSIS IN ANONYMOUS COMMUNICATION NETWORKS

A Dissertation

by

YE ZHU

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Co-Chairs of Committee,	Riccardo Bettati
	A. L. Narasimha Reddy
Committee Members,	Wei Zhao
	Michael Longnecker
Head of Department,	Costas N. Georghiades

August 2006

Major Subject: Computer Engineering

ABSTRACT

On Traffic Analysis in Anonymous Communication Networks. (August 2006)

Ye Zhu, B.S., Shanghai Jiao Tong University;

M.S., Texas A&M University

Co–Chairs of Advisory Committee: Dr. Riccardo Bettati
Dr. A. L. Narasimha Reddy

In this dissertation, we address issues related to traffic analysis attacks and the engineering in anonymous communication networks.

Mixes have been used in many anonymous communication systems and are supposed to provide countermeasures that can defeat various traffic analysis attacks. In this dissertation, we first focus on a particular class of traffic analysis attack, flow correlation attacks, by which an adversary attempts to analyze the network traffic and correlate the traffic of a flow over an input link at a mix with that over an output link of the same mix. Two classes of correlation methods are considered, namely time-domain methods and frequency-domain methods. We find that a mix with any known batching strategy may fail against flow correlation attacks in the sense that, for a given flow over an input link, the adversary can correctly determine which output link is used by the same flow. We theoretically analyze the effectiveness of a mix network under flow correlation attacks.

We extend flow correlation attack to perform flow separation: The flow separation attack separates flow aggregates into either smaller aggregates or individual flows. We apply blind source separation techniques from statistical signal processing to separate the traffic in a mix network. Our experiments show that this attack is effective and scalable. By combining flow separation and frequency spectrum matching method, a passive attacker can get the traffic map of the mix network. We use a non-trivial

network to show that the combined attack works.

The second part of the dissertation focuses on engineering anonymous communication networks. Measures for anonymity in systems must be on one hand simple and concise, and on the other hand reflect the realities of real systems. We propose a new measure for the anonymity degree, which takes into account possible heterogeneity. We model the effectiveness of single mixes or of mix networks in terms of information leakage and measure it in terms of covert channel capacity. The relationship between the anonymity degree and information leakage is described, and an example is shown.

To my family

ACKNOWLEDGMENTS

This dissertation would not have been completed without the help of many people.

First I would like to thank my co-chair Dr. Riccardo Bettati for his guidance and support during my graduate education. Without his broad vision and deep insight, his valuable advice and continuous encouragement, this research would not have been finished. I am grateful for the tremendous time and energy he spent in guiding my research. I am greatly indebted to him.

I would also thank my co-chair, Dr. A. L. Narasimha Reddy for his support to my graduate studies. His classes on computer networking and multimedia networking prepared me well for further research. His invaluable advice has been very helpful to my graduate study.

I am especially grateful to Dr. Wei Zhao. Through working with Dr. Wei Zhao, I leaned a lot on conducting high-quality research and writing research papers. I would also like to thank Dr. Wei Zhao for working on my papers day and night before submission.

I would also like to express my thanks to Dr. Michael Longnecker for his support and guidance as a member of my dissertation committee.

I would also like to thank Dr. Andreas Klappenecker for extremely helpful discussions on my research topics.

At last, I would like thank all the group members in the NetCamo project and the realtime research group for their help and fruitful discussions.

This work is supported in part by the Texas Information Technology and Telecommunication Task Force.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

CHAPTER Page

Page

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

As the Internet is increasingly used in all aspects of daily life, so has the realization that privacy and confidentiality are important requirements for the success of many applications. One particular aspect of privacy that is of importance to users of such applications is *anonymity*: the inability to identify the user as a participant in the application. Anonymity is feasible and beneficial in many scenarios, such as privacy-preserving web browsing, electronic voting, and many other e-business applications. The nature of many such applications requires that the identities of participants remain confidential from either other participants or from a third party.

In a computer system the anonymity of any participant is naturally preserved as long as that participant does not interact with others. Only when he communicates with others his identity is revealed. Hiding the identity of the participant during a communication therefore goes a long way towards preserving the participant's anonymity.

Achieving anonymity in open environments such as the Internet is a challenging problem. Encryption alone cannot preserve the anonymity of communication, since the identities or locations of participants can be easily inferred from data that is used to support the communication, for example packet headers. Additional measures must be put in place to hide the identities of participants.

Chaum [1] proposed the use of special intermediary nodes, or proxies, which he called *mixes*, to relay messages for anonymous email applications. The objective of a mix is to split the communication between sender and receiver into two separate

---

The journal model is *IEEE Transactions on Automatic Control.*

communications, and so sever the communication between sender and receiver.

An observer may monitor incoming and outgoing traffic at the mix, and so infer at least part of the sender-receiver relationship. To prevent this, a mix may delay, batch, and reorder packets to disrupt the packet-level timing correlation of packets that enter and leave the mix. We observe that single mix presents a single point of failure: When the single mix is compromised and the attacker has access to packet inside the mix, the anonymity of the users is compromised as well. The solution is to relay messages through multiple, mutually non-trusting mixes. In such a system, multiple mixes form a *mix network*, and a sender chooses a path through the mix network to communicate to the receiver. In general the sender uses source routing and encrypts messages in an onion-like way [2]: each intermediate mix gets the address of the next mix after decrypting the message and relays the "thinner" stripped message to the next mix with its own address as the source address.

The original Chaum mix operates on entire messages (originally e-mail) at a time, and therefore does not need to pay particular attention to latency added to message delivery by the mixes. Increasingly, the data exchanged among participants in networked applications, for example in file sharing, exceeds by far the capacity of mixes. As a result, current mixes operate on *individual packets* in a flow rather than on entire messages. In conjunction with source routing at the sender, this allows for very efficient network-level implementation of mix networks.

A. Problem Statement

This dissertation is concerned with the question if current anonymous communication systems can achieve anonymity given all these current anonymity techniques.

The anonymity techniques proposed or implemented are as follows:

- Aggregating: It is a common sense that it is easier to hide in a "crowd". By aggregating packets from a lot of users, the anonymity system can more efficiently hide the communication relationship inside anonymity networks.

- Batching: Packets arriving at the mix will be buffered first and then sent out in a batch. The trigger for batch transmission can be timer-based, threshold-based, or, a combination of both.

- Reordering: The order of packet departures will be randomly arranged so that less information on the packet departure order can be inferred from the packet arrival order or vice versa.

- Pooling: In a pooling based mix system, an arrival packet will be sent out with a probability each time the packet has a chance to be sent out.

- Padding: Mix system can insert dummy traffic into the outgoing traffic to further break the correlation between incoming and outgoing packets. But due to its cost on the network bandwidth and other practical reasons, padding is not used in the current anonymity systems.

- Rerouting: Instead of following the shortest path between the sender and the receiver, a traffic flow in anonymity networks usually use a longer path which is randomly chosen by the source routing mechanism at the sender.

These anonymity techniques are used in various combinations in current anonymity networks. For example, the pooling mechanism can be used in combination with batching to further "mix" the packets from different users.

In this dissertation we limit ourselves to *passive attacks*, meaning that the attackers observe the packets as they traverse the network and refrain from *actively*

modifying the traffic (for example by dropping individual packets). A passive attacker has no ability to disturb the existing traffic, such as by inserting packets, dropping packets, or modifying packets. Such *active attacks* can be very effective since they can "inject" traffic patterns into the system that are easy to monitor, or they can traffic to saturate network links and thus allow for a study of the system in overload mode. While highly effective, active attacks can be easily detected and shut down. Under these general assumptions, we would like to investigate if the current anonymity system can achieve anonymity given the proposed or implemented anonymity techniques.

B. Summary of Results

In this thesis we first focus on a class of *flow correlation* attacks that can determine the communication relationship inside an anonymity network by estimating dependence between individual traffic flows at the input of a mix and the aggregate traffic flows at its output. We formally proved that given enough data, flow correlation attacks can break anonymity of many current anonymous communication systems with a probability arbitrary close to 100% and derive the formula to show the effects of varying the parameters that control the anonymity system on the latter's effectiveness.

We then propose a traffic analysis method (which we call flow separation) to separate individual traffic flows based on the observations of aggregate traffic flows. The method employs blind source separation, a classical method in statistical signal processing to separate individual signal components given mixtures of signal components. We show how flow separation can effectively separate aggregates of flows into individual flows or smaller aggregates as they traverse mixes in the network. In conjunction with flow correlation, the attack can successfully determine the commu-

nication relationship.

We also apply traffic analysis to wireless ad-hoc networks. We adopt the threat model proposed in [3], where a sensor network deployed in a field is used to monitor the wireless signals transmitted by wireless nodes inside the field. These sensors used can be either part of a separate monitoring or surveillance infrastructure or consist simply of the other participants in the abhor network. We apply traffic analysis on the aggregate traffic "heard" by the sensors to separate individual flows sent by wireless nodes. The attacks based on the method can both accurately and precisely estimate the location of wireless nodes.

In the second part of the dissertation, we address the question of how to *engineering* good anonymity networks. To engineer an anonymity network, it is important to have a measure which can take into account the topology of the planned anonymity infrastructure. For this, we proposed an anonymity degree to capture the quality of anonymity networks. Our definition can generalize the information-theoretic definitions proposed in [4, 5]. We also proposed a new class of anonymity channels, which we call *anonymity-based covert channels*. We show how the capacity of anonymity-based covert channels can be used to provide simple description of non-perfect mix networks, and can be used to formulate bounds on the provided anonymity.

C. Organization

The rest of the dissertation is organized as follows:

Chapter II reviews the related work, including evolution of anonymity systems, attacks on anonymity systems, the metrics to measure the anonymity provided by anonymity networks, covert channels, and locating wireless nodes.

Chapter III describes the flow correlation attack in anonymity networks. The

objective of the flow correlation attack is to correlate an incoming flow at a mix with several aggregate outgoing flows. In this chapter, we propose two measures to correlate traffic flows: *mutual information*, an information-theoretical measure, and *spectrum matching*. We verified the proposed attacks through experimentation in a testbed and propose a countermeasure.

Modeling and theoretical analysis of flow correlation attacks are presented in Chapter IV. In this chapter, a framework to model the effectiveness of flow correlation attacks is proposed. The framework provides a guideline for anonymity network designers on how to select system parameters.

In Chapter V, we introduce a traffic analysis method based on flow separation and its application in mix network. As opposed to flow correlation, where an individual flow is correlated to aggregates of output flows, the separation attack partitions both incoming and outgoing flows into small aggregates. This traffic analysis method can also be used to pre-condition collected data for traffic analysis attacks. In this case the possibly large aggregates of flows are separated into either individual flows or small aggregates, thus greatly simplifying the work of subsequent correlation steps. We investigate the flow separation method under different combinations of traffic, and we evaluate its scalability. A simulation of flow separation attacks against an anonymity network is used to demonstrate the strength of the method.

A traffic analysis method for wireless networks is proposed in Chapter VI. The method can be used to detect location of wireless nodes. Further attacks on this analysis method can disclose motion information of wireless nodes and communication relationship between wireless nodes.

Chapter VII presents a new anonymity degree and its relationship to information leakage through the anonymity network in the form of covert channel. We propose a new type of covert channel, called anonymity-based covert channel. We investigate

the relationship between anonymity degree and information leakage for both single mix case and mix network case.

We conclude this dissertation and outline future work in Chapter VIII.

CHAPTER II

RELATED WORK

A. Evolution of Anonymity Systems

The oldest form of support for anonymity in interpersonal transactions is perhaps the use of cash. When buying problematic goods, a purchaser would like to use cash because transaction through cash will leave no records about the parties involved in the transaction. Similarly, a purchaser may be worried about identity theft, and so opt for a cash-based transaction, which leaves no record that could be mis-used. In modern society, secure and untraceable electronic cash (E-cash) system has been proposed in [6, 7].

For anonymous email applications, Chaum [1] proposed to use relay servers, called *mixes*, which reroute messages that are encrypted by the public keys of the mixes. An encrypted message is analogous to an onion constructed by a sender, who sends the onion to the first mix. Using its private key, the first mix peels off the first layer. Inside the first layer is the second mix's address and the rest of the onion, which is encrypted with the second mix's public key. After retrieving the second mix's address, the first mix forwards the peeled onion. This process proceeds in this recursive way until the core part of the onion is forwarded to the receiver. Chaum also proposed return address and digital pseudonyms for users to communicate with each other anonymously.

A number of evolutions of Chaum's mix were proposed over the years. Helsingius [8] implemented the first Internet anonymous *remailer*, which is a single application proxy that just replaces the original email's source address with the remailer's address. It has no reply function and is subject to a number of attacks, which we will describe

in Section II.B. Eric Hughes and Hal Finney [9] built the *cypherpunk remailer*, a real distributed mix network with reply functions that uses PGP to encrypt and decrypt messages. Gülcü and Tsudik [10] developed a relatively full-fledged anonymous email system, called *Babel*. Their reply technique does not need the sender to remember the secret seed to decrypt the reply message. Cottrell [11] developed *Mixmaster* which counters a global passive attack by using message padding and also counters trickle and flood attacks [10, 12] by using a pool batching strategy. Mixmaster does not have a reply function. Danezis, Dingledine and Mathewson [13] developed *Mixminion*, with consideration for a relatively complete set of attacks that researchers have found [12, 14, 15, 16, 17, 18]. The authors in [13] suggest a list of research topics for future study.

The mix networks described above are called *message-based* networks because they forward entire messages, for example email messages, in a store-and-forward fashion. More recently, message-based mix networks have been extended to *flow-based*, also called *low-latency* anonymous communication networks for low-latency communication. Low-latency mixes operate at a per-packet (instead of per-message) level. Low-latency anonymous communication can be further divided into systems using core mix networks and peer-to-peer networks. In a system using a core mix network, users connect to a pool of mixes, which provides anonymous communication, and users select a forwarding path through this core network to the receiver. *Onion routing* [2] and *Freedom* [19] belong to this category. In contrast, in a system using a peer-to-peer network, every node in the network is a mix, but it can also be a sender and receiver. Obviously, a peer-to-peer mix network can be very large and may provide better anonymity in the case when many participants use the anonymity

service and sufficient traffic is generated around the network. *Crowds* [20][1] , *Tarzan* [21] and P^5 [22] belong to this category.

A typical example of low-latency mix is Tor [23], the second-generation onion router, developed for circuit-based low-latency anonymous communication. The Tor network supports anonymous transport of TCP streams such as HTTP sessions. It can provide perfect forward secrecy and support hidden server. Tor network is available for public use and already has more than 50 nodes [24].

Following the realization that serious privacy issues are at stake when location information is accessible in many pervasive applications and wireless networks (see, for example, the work of the IETF Working Group on Geographic Location/Privacy (geopriv) [25]), several communication systems to preserve anonymity and *location privacy* in ad-hoc and infrastructure-based wireless networks have been proposed. ANODR [26] protects route anonymity by a onion-based encryption and routing protocol. The data transmission in ANODR is based on broadcast, and identity disclosure is prevented by the use of broadcast MAC addresses. SDDR [27] also uses an onion-routing scheme for routing in wireless ad-hoc networks. ASR [28] was designed to provide stronger anonymity by preventing the nodes en route to know the hop count to the sender or receiver. Its data transmission is broadcast-based and it assumes the presence of a shared secret between nodes. For infrastructure-based networks, Gruteser et al. [29] proposed the use of disposable MAC addresses in order to prevent tracking of mobile hosts.

A completely different approach is taken in DC networks (see [30]) where each participant shares secret coin flips with other pairs and announce the parity of the

[1]Although Crowds may not use the classical Chaum's mix, for simplicity, we still use the name of "mix" to refer to a single anonymity network hop, and our theory can be applied to all rerouting-based anonymity networks.

observed flip to all other participants and to the receiver. The total parity should be even, since each flip is announced twice. By incorrectly stating the parity the sender has seen, this causes the parity to be odd. Thus the sender can send a message to the receiver. The receiver receives the message whenever it finds the parity to be odd. Nobody except the sender knows who sent the message. This scheme relies on an underlying broadcast medium, which comes at a great expense as the number of participants grows. Due to this lack of scalability none of the currently deployed systems employs this method. In the following we will focus on rerouting-based systems.

B. Anonymity Attacks

This dissertation is interested in the study of passive traffic analysis attacks against low-latency[2] anonymous communication systems. In [31, 32], a quantitative performance analysis is given for an anonymous web server that applies both encryption and anonymizing proxies. The analysis takes advantage of the fact that a number of HTTP features, such as the number and size of objects, can be used as signatures to identify web pages with some accuracy. An observer could monitor the size of HTTP objects requested by the browser and compare them with database of previously collected object size. Unless an anonymizer addresses this issue, these signatures are visible to the adversary.

Serjantov and Sewell [33] analyzed the possibility of a lone flow along an input link of a mix in peer-to-peer anonymity systems. If the rate of this lone input flow is approximately equal to the rate of a flow out of the mix, this pair of input and

[2]We use an operational definition for "low-latency". We call a communication system low-latency in this context when it does not unduly disrupt TCP connections under normal load conditions. Using this definition, TOR is clearly low-latency.

output flows are correlated. They also discussed possible traffic features used to trace a flow. Other analysis focus on the anonymity degradation when some mixes are compromised, e.g., [20].

Levine et al. [34] describe an approach to discover communication relationship. In their particular case, they have access to per-flow packet data. This is the case, for example, when the attacker has access to the unprotected flows entering and leaving the mix network. Similarly, the mixes at the two ends of the flow may be controlled by the attacker. Given this information, the attacker can use cross correlation to measure the similarity between individual flows.

Danezis [35] describes an attack on the *Continuous Mix* : in such a mix packets get individually delayed according to some probability distribution. Since the packet delays are independent, the departure distribution of the packets of a flow can be accurately described (if one ignores queuing) by convoluting the packet-arrival and the delay distribution. This can be used as a basis for measuring similarities among flows.

C. Anonymity Degree, Covert Channel

To capture the effectiveness of anonymity systems under anonymity attacks, a number of different anonymity degree definitions have been proposed: The first *anonymity degree* measure, proposed in [20], is defined as the probability of not being identified by the attacker. It focuses on each user and does not capture the anonymity of the whole system. Berthold *et al.* [16] propose an anonymity degree based on the number of the users of an anonymity system. There is an ongoing debate about what the role of the number of users is in providing anonymity. Intuitively, the larger the crowd, the easier it is for an individual to hide in it. In practice, however, attacks

proceed by isolating users or groups of users that are more likely to be participants in a communication. This was first considered in the *anonymity set*, introduced in [30]. The anonymity set describes the set of *suspected* senders or receivers of a message. The size of the anonymity set is used in [36] as the anonymity degree.

The anonymity set measure does not take into account that some members in the anonymity set are inherently more likely to be receivers or senders of a message; for example as a result of *a priori* information. A big step forward was done by Serjantov and Danezis [5], by Diaz *et al.* [4] and by Guan *et al.* [37] by proposing anonymity measures that consider probability distributions in the anonymity set. All these measures are based on entropy and can differentiate two anonymity sets that have identical sizes, but different distributions. The measure in [4] normalizes the anonymity degree to discount for the anonymity set size.

A number of efforts have studied the relation between covert channels and anonymity systems. Moskowitz *et al.* [38] focus on the covert channel over a mix-firewall between two enclaves. The covert channel in this case is established by the channel receiver determining whether an anonymized sender is transmitting packets. Newman *et al.* [39] focus on the covert channel over a timed mix. The authors in [40] make a series of excellent observations about the relation between covert channels and anonymity systems. They illustrate this relation by describing the linkage between the lack of complete anonymity (quasi-anonymity) and the covert communication over different type of mixes and propose to use of this covert channel capacity as a metric for anonymity.

D. Related Work on Locating Wireless Nodes

Numerous papers have been published on locating wireless nodes. Many of them are based on the characteristic of physical signals, such as Received Signal Strength (RSS) [41, 42], Angle of Arrival (AOA) [43, 44], and Time of Arrival (TOA) [45]. Complex processing methods on collected data, such as triangulation [46], Kalman filter [45], and robotics-based approaches [41] are needed to deal with the physical signal's non-linearity, noise, and the complex correlations caused by multi-path effects, interference, and absorption. Elnahrawy et al. [47] point out a number of fundamental limits associated with the use of signal strength for example and claim that the limits are unlikely to be transcended.

Senders can easily counter location estimation attacks based on signal-strength by fluctuating the transmission power. This has been proposed in Whisper [48]. Location privacy attacks using Angle-of-Arrival data assume that sensors have directional capabilities, which adds greatly to the cost of the sensor network. One objective of Chapter VI is to illustrate how most of current anonymity methods for wireless networks, such as encryption, MAC address hiding, signal power fluctuations, link padding, and others are of limited effectiveness in 802.11-style setting. For this, we assume that the sensors used do not make use of information that can be hidden by anonymity measures. Therefore we assume that sensors have no access to header data, such as sender or receiver information, or packet data, or signal strength, or directional information.

In general, schemes that rely on physical-level, analog signals require large volumes of data to be transfered over the wireless sensor network for further analysis. In comparison, the schemes proposed in Chapter VI rely on highly aggregated packet-count data which can be easily propagated across a low-bandwidth infrastructure.

Spatial and temporal redundancy of the packet-count data from different sensors can be exploited to further reduce traffic volume by using compression methods such as ESPIHT [49].

CHAPTER III

WEAKNESSES IN CURRENT ANONYMOUS COMMUNICATION SYSTEMS

A. Motivation

In this chapter, we study to which level current anonymous communication systems can achieve anonymity. Little is known about the *effectiveness* of mixes in providing anonymity. Although significant efforts have been put forth in researching anonymous communication since Chaum, only recently systematic studies have appeared to quantitatively capture the effect of the traffic perturbation caused by the various mechanisms in the mixes (batching, pooling, and so on) on the anonymity degree. As we will describe in subchapter III.E.3 any such disturbance of traffic can lead to a decrease in the goodput as perceived by users, and the desired level of anonymity must be traded off against this cost. Quantitative studies on the effectiveness of anonymity measures are therefore important to assess the improvement of anonymity that one attains for any given cost. Moreover, few quantitative guidelines exist on how different perturbation mechanisms perform. In fact, a number of current anonymous communication systems (for example Tarzan [21]) employ rather heavy-handed methods, such as link padding (generating large amounts of dummy traffic in addition to any cross traffic already present in the network to confuse the observer) as defense against traffic analysis, with little regard for cost and without a clear understanding of how these measures actually benefit anonymity in the system. This chapter focuses on the quantitative evaluation of mix performance. We focus our analysis on a particular type of attack, which we call *flow correlation attack*. In general, flow correlation attacks attempt to reduce the anonymity degree by estimating the path of flows through the mix network. Flow correlation analyzes the traffic on a set of

links (observation points) inside the network, and estimates the likelihood for each link to be on the path of the flow under consideration. An adversary analyzes the network traffic with the intention of identifying which of several output ports a flow at an input port of a mix is taking. Obviously, flow correlation helps the adversary identify the path of a flow and consequently reveal other mission critical information related to the flow, such as the sender and receiver.

B. Network Model

1. Mix and Mix Network

A mix is a relay device for anonymous communication. Figure 1 shows the communication between users that use a single mix. Such a mix can achieve a certain level of communication anonymity: The sender of a message attaches the receiver address to a packet and encrypts it using the mix's public key. Upon receiving a packet, a mix first decodes the packet. Different from an ordinary router, the mix usually will not relay the received packet immediately. Rather, it collects several packets and then sends them out in a *batch*[1]. The order of packets may be altered as well. Techniques such as batching and reordering are considered necessary techniques for mixes to prevent timing-based attacks. The main objective of this chapter is to analyze the effectiveness of mixes against a special class of timing-based attacks.

A *mix network* consists of multiple mixes that are inter-connected by a network. A mix network may provide enhanced anonymity, as payload packets may go through multiple mixes. Even in such a mix network it is important that each individual mix provides a sufficient level of anonymity at an acceptable cost so that the end-to-

[1].In this section we focus our attention on so-called *batching* mixes. Other types of mixes for example stop-and-go or continuous mixes, exist which use per-packet schemes to perturb the traffic.

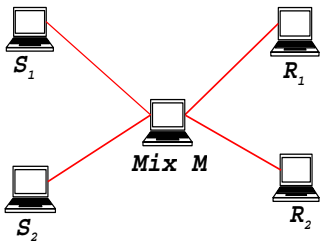


Fig. 1. A Single Mix

end performance can be guaranteed. Thus, our analysis on a single mix provides a foundation for analyzing the end-to-end performance of mix networks. In fact, if we view a mix network (for example Onion routing [2]) as one *super mix*, the analytical techniques in this chapter can be directly applied.

2. Batching Strategies for a Mix

Batching strategies are designed to prevent not only simple timing analysis attacks but also many other forms of attacks ([12, 13]). Serjantov [12] summarizes seven batching strategies that have been proposed in the literature. These seven batching strategies are listed in Table I, in which batching strategies from S_1 to S_4 are denoted as *simple mix*, while batching strategies from S_5 to S_7 are denoted as *pool mix*. Our results show that these strategies may not work under certain timing analysis attacks.

From Table I, we can see that the sending of a batch of packets can be triggered by a number of different events, e.g., queue length reaching a pre-defined threshold, a timer having a time out, or some combination of these two.

Batching is typically accompanied by reordering. In this chapter, the attacks focus on the traffic timing characteristics. As reordering does not change packet interarrival times much for mixes using batching, these attacks (and our analysis)

Table I. Batching Strategies [12]

Glossary

n	queue size
m	threshold to control the packet sending
t	timer's period if a timer is used
f	the minimum number of packets left in the pool for pool Mixes
p	a fraction only used in Timed Dynamic-Pool Mix

Algorithms

Strategy Index	*Name*	*Adjustable Parameters*	*Algorithm*
S_0	Simple Proxy	*none*	no batching or reordering
S_1	Threshold Mix	$< m >$	if $n = m$, send n packets
S_2	Timed Mix	$< t >$	if timer times out, send n packets
S_3	Threshold Or Timed Mix	$< m, t >$	if timer times out, send n packets; elseif $n = m$ {send n packets; reset the timer}
S_4	Threshold and Timed Mix	$< m, t >$	if (timer times out) and $(n \geq m)$, send n packets
S_5	Threshold Pool Mix	$< m, f >$	if $n = m + f$, send m randomly chosen packets
S_6	Timed Pool Mix	$< t, f >$	if (timer times out) and $(n > f)$, send $n - f$ randomly chosen packets
S_7	Timed Dynamic-Pool Mix	$< m, t, f, p >$	if (timer times out) and $(n \geq m + f)$, send $\max(1, \lfloor p(n - f) \rfloor)$ randomly chosen packets

are largely unaffected by reordering. Thus, our results are applicable to systems that use any kind of reordering methods. As such, in the rest of this chapter, we will not discuss reordering techniques further.

Any of the batching strategies can be implemented either at link-level or inside the mix:

- Link-Based Batching: With this method, each output link has a separate queue. A newly arrived packet is put into a queue depending on its destination (and hence the link associated with the queue). Once a batch is ready from a particular queue (per the batching strategy), the packets are taken out of the queue and transmitted over the corresponding link.

- Mix-Based Batching: In this batching scheme, the entire mix maintains a single queue of packets. The selected batching strategy is applied to this queue. That is, once a batch is ready (per the batching strategy), the packets are removed from the queue and transmitted over the appropriate links, based on the packets' destination.

Each of these two methods has its own advantages and disadvantages. The control of link-based batching is distributed inside the mix and hence it may have good efficiency. On the other hand, mix-based batching uses only one queue and hence is easier to manage. We consider both methods in this chapter.

C. Threat Model

In this chapter, we assume that the adversary applies a timing analysis attack on some sample of observed traffic ([50, 51]). We summarize the threat model as follows.

The adversary observes input and output links of a mix, collects the packet interarrival times, and analyzes them. This type of attack is *passive*, since traffic is

not actively altered (by, say, dropping, inserting, and/or modifying packets during a communication session), and is therefore often difficult to detect. This type of attack can be easily staged on wired and wireless links [52] by a variety of agents, such as malicious ISPs or governments [53]. We make the simplifying assumption that the traffic characteristic of the flow under consideration (the *input flow*) is known. This can be the case for example because the flow traffic characteristic is indeed observable at the input or at the input of the mix network.

To maximize the power of the adversary, we assume that she makes observations on all the links of the mix network. Such an adversary is called *global* as opposed to a *local* adversary, which has access to a single observation point, or maybe a small collection thereof.

The mix's infrastructure and strategies are known to the adversary. This is a typical assumption in the study of security systems.

The adversary cannot correlate (based on packet timing, content, or size) a packet on a input link to another packet on the output link. Packet correlation based on packet timing is prevented by batching, and correlation based on content and packet size is prevented by encryption and packet padding, respectively. Padding of packets is achieved by appending random data at the end of packets as appropriate to ensure that all packets have the same length. It is the intent of the attack to correctly correlate these packets without access to this data.

There has been some discussion about the benefit of *link padding* with dummy packets. In such a system, additional packets are generated by mixes to further increase the number of packets on the links and so make traffic analysis harder. To simplify the following discussion, we assume that dummy traffic is not used in the mix network. Some of the modern anonymous communication systems such as Onion Routing [54] and Tor [23] do not use dummy traffic because of its heavy consumption

of bandwidth and the general lack of understanding of to what extent exactly dummy packets contribute to anonymity.

Finally, we assume that the specific objective of the adversary is to identify the output link of a traffic flow that appears on an input link. Others have described similar attacks, but under simplified circumstances. Serjantov and Sewell [33], for example, assume that the flow under attack is alone on a link thus making its traffic characteristics immediately visible to the attacker. In this chapter, we consider flows inside (potentially large) aggregates, thus making the attack generally applicable.

D. Traffic Flow Correlation Techniques

This section discusses the traffic flow correlation techniques that may be used by the adversary either to correlate senders and receivers directly or to greatly reduce the searching time for such a correlation in a mix network.

1. Overview

Recall that the adversary's objective is to correlate an incoming flow to an output link at a mix. We call this *flow correlation*. This kind of flow correlation attack is harmful in many scenarios. For example, in Figure 1, the adversary can discover the communication relationship between senders (S_1 and S_2) and receivers (R_1 and R_2) by matching senders' output flows and receivers' input flows. Using the flow correlation attack techniques, the adversary can determine a flow's sender and receiver if she catches a fragment of the flow in the mix network, thus breaking the anonymity despite the mix network. In a mix network, the adversary can even reconstruct the path of this connection by using flow correlation techniques. This subsection discusses the attack in more detail.

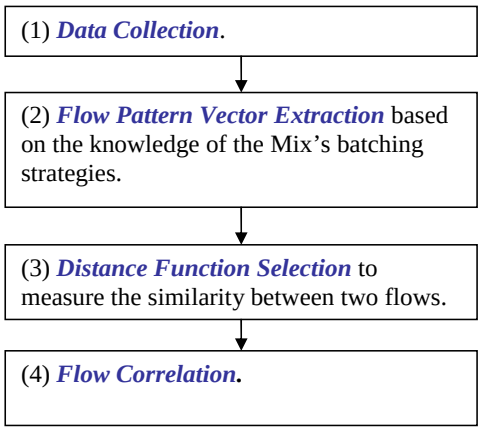


Fig. 2. Typical Flowchart for Flow Correlation

Figure 2 shows a flowchart of the typical procedure that the adversary may use to perform flow correlation. We now describe each step in detail.

a. Step 1: Data Collection

We assume that the adversary is able to collect information about all the packets on both input and output links. For each collected packet, the arrival time is recorded using tools such as tcpdump [55] and Cisco's NetFlow [56]. We assume that all the packets are encrypted and padded to the same size, and hence only arrival time is of interest. The arrival times of packets at input link i form a time series

$$A_i = (a_{i,1}, \cdots, a_{i,n}) \ , \tag{3.1}$$

where $a_{i,k}$ is the arrival time of the k^{th} packet at input link i, and n is the size of the sample collected during a given sampling interval. Similarly, the arrival times of packets at output link j form a time series

$$B_j = (b_{j,1}, \cdots, b_{j,m}) \ , \tag{3.2}$$

where $b_{j,k}$ is the arrival time of the k^{th} packet at output link j, and m is the size of the sample collected during a given sampling interval. The packets leave the mixes in batches. The length of time packets are queued for batching has to be sufficiently short as to not to interfere with latency requirements. The length of a sampling interval is therefore usually much longer than the duration of a batch. Hence, a sampling interval typically contains many batches.

b. Step 2: Flow Pattern Vector Extraction

With the above notation, the strategy of the adversary is to analyze the time series A_i and B_j in order to determine if there is any "similarity" between an input flow and an output flow of the mix. However, a direct analysis over these time series will not be effective. They need to be transformed into so called *pattern vectors* that can facilitate further analysis. We have found that effective transformations depend on the particular batching strategies utilized by the mix. In Section 3, we will discuss specific definitions of transformations for different batching strategies. Currently, for the convenience of discussion, let us assume that A_i is transformed into pattern vector $X_i = (x_{i,1}, \cdots, x_{i,q})$, and time series B_j is transformed into $Y_j = (y_{j,1}, \cdots, y_{j,q})$. For simplicity we assume that the two pattern vectors have the same length.

c. Step 3: Distance Function Selection

We define the distance function $d(X_i, Y_j)$, which measures the "distance" between an input flow at input link i and the traffic at output link j. The smaller the distance, the more likely the flow on an input link is correlated to the corresponding flow on the output link. Clearly, the definition of the distance function is the key in the correlation analysis. Section 2 will discuss two effective distance functions: one is based on mutual information and the other is based on the frequency-spectrum-based

matched filter.

d. Step 4: Flow Correlation

Once the distance function has been defined between an input flow and an output link, we can easily carry out the correlation analysis by selecting the output link whose traffic has the minimum distance to input flow pattern vector X_i.

2. Flow Pattern Vector Extraction

In this subsection, we discuss how to choose pattern vectors X_is and Y_js. We will start with pattern vectors for the output link traffic first. Recall that batching strategies in Table I can be classified into two classes: threshold-triggered batching (S_1, S_3, and S_5)[2] and timer-triggered batching (S_2, S_4, S_6 and S_7). We will see that different classes should have different transformation methods.

For threshold-triggered batching strategies, packets come out from the mix in batches. Hence, the inter-arrival time of packets in a batch is determined by the transmission latency, which is independent of the input flow. Thus, the useful information to the adversary is the number of packets in a batch and the time elapses between two batches. Normalizing this relationship, we define the elements in pattern vector Y_j as follows:

$$Y_{j,k} = \frac{\text{Number of packets in batch k in the sampling interval}}{\text{(Ending time of batch k) - (Ending time of batch k-1)}} \tag{3.3}$$

In the calculation, we may need to truncate the original time series $B_j = (b_{j,1}, b_{j,2}, \cdots, b_{j,n})$ so that only complete batches are used.

[2] S_3 could also be classified as timer-triggered. However, we treat it as threshold triggered because it may send out a batch when the number of packets received by the mix has reached the threshold.

For timer-triggered batching strategies, a batch of packets is sent whenever a timer fires. The length of the time interval between two consecutive timer events is a pre-defined constant of length δ. To compute the pattern vector, we partition the time line into slots of length δ. Thus, following a similar argument made for the threshold-triggered batching strategies, we define the elements in pattern vector Y_j as follows:

$$Y_{j,k} = \frac{\text{Number of packets in the } k^{th}\text{time slot}}{\delta} \tag{3.4}$$

Again, in the calculation, we may need to truncate the original time series B_j so that only complete batches are used.

For the traffic *without batching* (i.e., the baseline strategy S_0 in Table I), we use similar methods defined for timer-triggered batching strategies as shown in (3.4).

The basic idea in the methods for extraction of pattern vectors is to partition a sampling interval into multiple sub-intervals and calculate the average traffic rate in each sub-interval. The above two methods differ on how to partition the interval, depending on which batching strategy is used by the mix. We take a similar approach to extract pattern vectors X_i at the input of mixes. Again, the specific method of sub-interval partition depends on how the mix is batching the packets.

3. Distance Functions

In this chapter, we consider two kinds of distance functions: the first is based on a comparison of mutual information and the second on frequency analysis. The motivation and computation methods are given below.

a. Mutual Information

Mutual information is an information-theoretical measure of the dependence of two random variables. In our scenario, we can view the pattern vectors that represent the input and output flows as samples of random variables. If we consider the pattern vectors X_i and Y_j to be each a sample of the random variables $\mathcal{X}_i$ and $\mathcal{Y}_j$, respectively, then $\{(X_{i,1}, Y_{j,1}), \cdots, (X_{i,q}, Y_{j,q})\}$ correspond to a sample of the joint random variable $(\mathcal{X}_i, \mathcal{Y}_j)$. With these definitions, the distance function $d(X_i, Y_j)$ between pattern vectors X_i and Y_j should be approximately inversely proportional to the mutual information $I(\mathcal{X}_i, \mathcal{Y}_j)$ between $\mathcal{X}_i$ and $\mathcal{Y}_j$,

$$d(X_i, Y_j) = \frac{1}{I(\mathcal{X}_i, \mathcal{Y}_i)} = -\frac{1}{\int\int p(x_i, y_j) \log \frac{p(x_i, y_j)}{p(x_i)p(y_j)}} \tag{3.5}$$

Here, we need to estimate marginal distributions ($p(x_i)$ and $p(y_j)$) and their joint distribution $p(x_i, y_j)$. In this chapter, we use histogram-based estimation of mutual information $\hat{I}(\mathcal{X}_i, \mathcal{Y}_j)$ of continuous distributions [57], which is given as follows.

$$\hat{I}(\mathcal{X}_i, \mathcal{Y}_j) \approx \sum_{u,v} \frac{K_{uv}}{q} \log \frac{K_{uv} N}{K_{u.} K_{.v}} \tag{3.6}$$

where q is the sample size. The sample space is a two-dimensional plane divided into $U \times V$ equally-sized $\Delta X \times \Delta Y$ cells with coordinates (u, v). K_{uv} is the number of samples in the cell (u, v). ΔX and ΔY have to be carefully chosen for an optimal estimation.

b. Frequency Analysis

We use mutual information to measure the distance in the time domain. For the frequency domain, we use spectrum information to measure the distance between the input and outputs. For timer-triggered batching strategies, we therefore use FFT on the sample X_i and Y_j to obtain the frequency spectrum X_i^F and Y_j^F. Then we apply matched filter method over X_i^F and Y_j^F. We take advantage of the fact that frequency components of the input flow traffic carry on to the aggregate flow at the output link. Matched filter is an optimal filter to detect a signal buried in noise. It is optimal in the sense that it can provide the maximum signal-to-noise ratio at its output for a given signal. In particular, by directly applying the theory of matched filters, we can define the distance function $d(X_i, Y_j)$ as the inverse matched filter detector $M(X_i^F, Y_j^F)$,

$$d(X_i, Y_j) = \frac{1}{M(X_i^F, Y_j^F)} = \frac{1}{\frac{<X_i^F, Y_j^F>}{||Y_j^F||}} \tag{3.7}$$

where $< X_i^F, Y_j^F >$ is the inner product of X_i^F and Y_j^F, and $||Y_j^F|| = \sqrt{< Y_j^F, Y_j^F >}$. Please refer to [58] for details about the calculation of FFT over a vector.

E. Empirical Evaluation

In this section, we evaluate the effectiveness of a selection of batching strategies (listed in Table I) for a mix under our flow correlation attacks. We will see the failure of a mix under our traffic flow correlation attacks and batching strategies' influence on TCP flow performance. Our experiments reported here focus on TCP flows because of their dominance in the Internet. However, the results are generally applicable to other kinds of flows.

1. Metrics

We use *detection rate* as a measure of the ability of the mix to protect anonymity. Detection rate here is defined as the ratio of the number of correct detections to the number of attempts. While the detection rate measures the *effectiveness* of the mix, we measure its *efficiency* in terms of quality of service (QoS) perceived by the applications. We use *FTP goodput* as an indication of FTP quality of service (*QoS*). FTP goodput is defined as the rate at which a FTP client receives data from a FTP server. Low levels of FTP goodput indicate that the mix in the given configuration is poorly applicable for low-latency flow-based mix networks.

2. Experiment Network Setup

Figure 3 shows our experimental network setup. Our mix is implemented on Timesys/Real Time Linux operating system for its timer accuracy [59]. The Mix control module that performs the batching and reordering functions is integrated into Linux's firewall system [60] using *Netfilter*; we use the corresponding firewall rules to specify what traffic should be protected. Two delay boxes D_1 and D_2 emulate the Internet propagation delay on different paths.

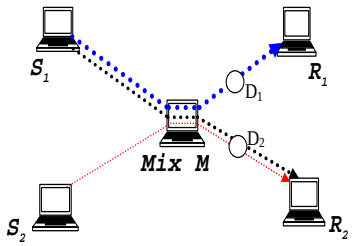


Fig. 3. Experiment Setup

The traffic flows in our experiments are configured as follows: An FTP client on

node R_2 downloads a file from the FTP server on S_2. The traffic from S_1 to R_2 serves as the random noise traffic to the FTP client. The traffic from node S_1 to node R_1 is the cross traffic through mix M from the perspective of the FTP flow. We maintain the traffic rate on both output links of the mix at approximately 500 packets per second (*pps*). The objective of the adversary in this experiment is to identify the output link that carries the FTP flow.

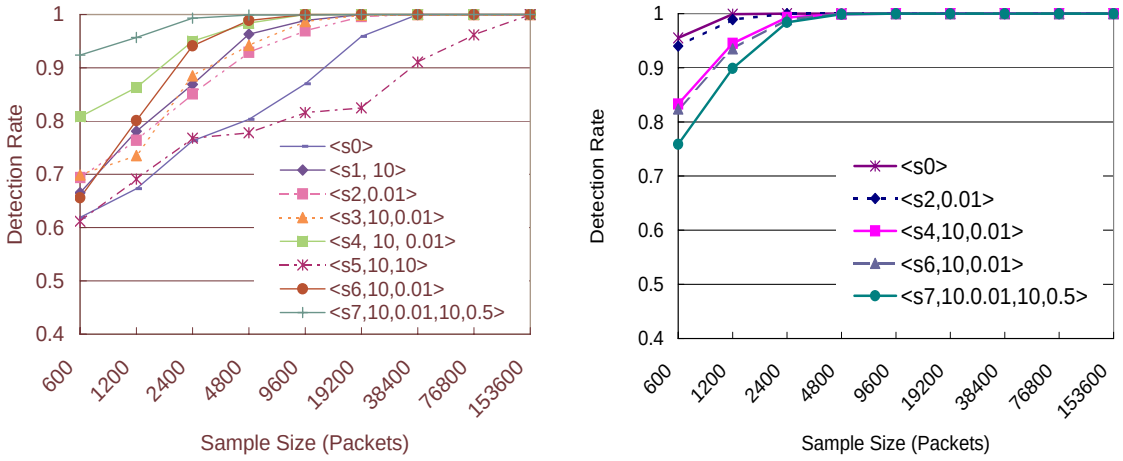
3. Performance Evaluation

a. Effectiveness of Batching Strategies

Figure 4 shows the detection rate for systems using a link-based batching strategy. Figure 5 shows the detection rate for systems using a mix-based batching strategy as a function of the number of packets observed. A sample may include both FTP packets and cross traffic packets while FTP packets account for less than 20% of the number -sample size- of packets. Parameters in the legends of these figures are listed in the same order as in Table I. Based on these results, we make the following observations.

- For all the strategies, the detection rate monotonically increases with increasing amount of available data. The detection rate approaches 100% when the sample size is sufficiently large. This is consistent with intuition, as more data implies that there is more information about the input flow, which in turn improves the detection rate.

- Different strategies display different resistances to flow correlation attacks. In general, pool mixes perform better than simple mixes based on matched filter detector.



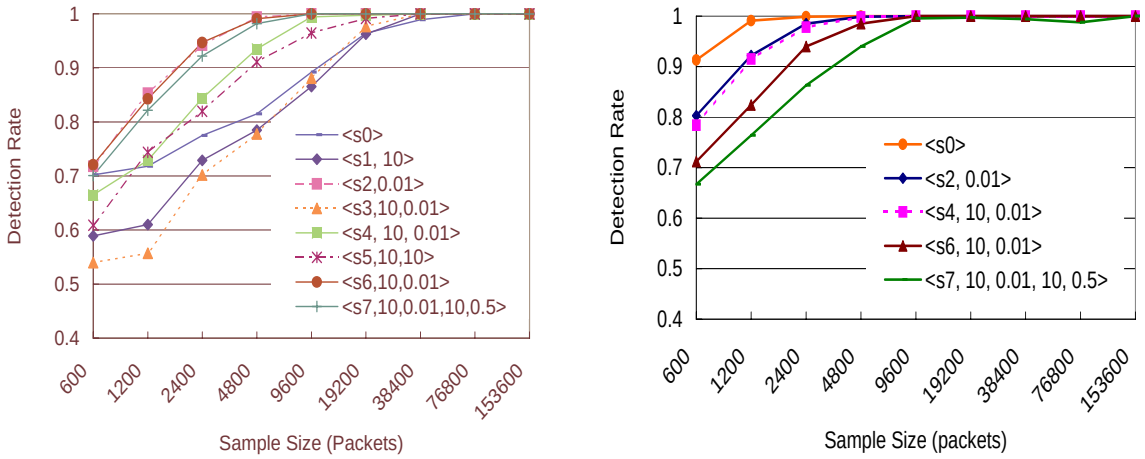
(a) Mutual Information (b) Matched Filter

Fig. 4. Detection Rate for Link-based Batching

- Frequency-analysis-based distance functions typically outperforms mutual-information-based distance functions in terms of detection rate. For many batching strategies, the former performs significantly better. This is because there are phasing issues in frequency-analysis-based attacks. Therefore, lack of synchronization between data collected at input and output port has a minor effect on the effectiveness of the attack.

- To compare mix-based batching strategy with link-based batching strategy, we find that no one dominates the other.

Overall, our data shows that mix using any of the batching strategies S_1, S_2, $\cdots$, S_7 fails under the flow correlation attacks. One of the reasons is that TCP flows often demonstrate interesting patterns such as periodicity of rate change and burstiness, in particular when the TCP loop-control mechanism is triggered by excessive traffic perturbation in the mixes. Figure 4 and 5 show that flow correlation attacks can well explore this pattern difference between TCP flows.



(a) Mutual Information (b) Matched Filter

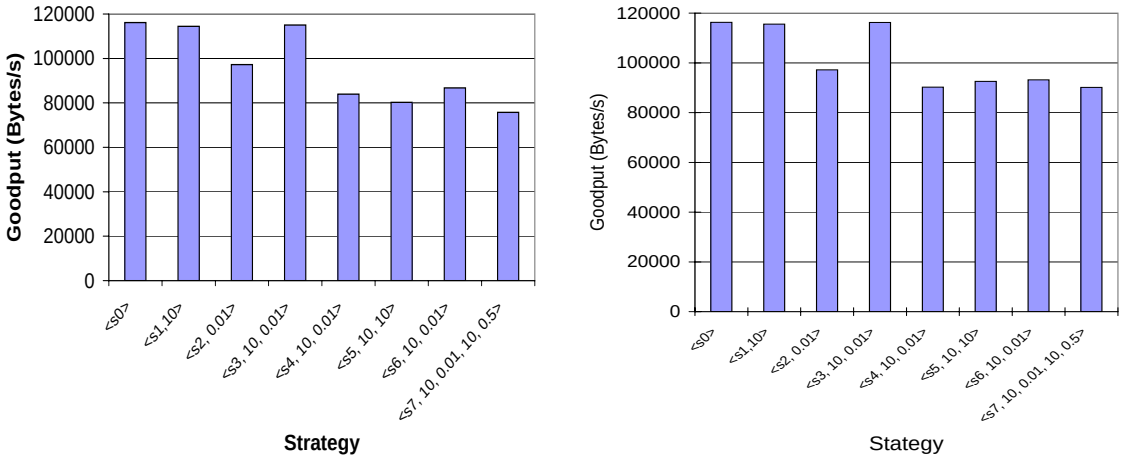
Fig. 5. Detection Rate for Mix-based Batching

b. Efficiency of Batching Strategies

As batching delays packets, one should expect that the overall performance (in terms of throughput) of TCP connections will be impacted by the mixes along their path. Figure 6 quantitatively shows the degradation of FTP goodput for a mix using different batching strategies.

In Figure 6, we compare FTP goodput between a strategy without any batching (S_0) and other batching strategies ($S_1, S_2, \cdots, S_7$). We still use the network setup in Figure 3. The traffic other than FTP is configured as follows: 400pps from S_1 to R_1 and 500pps from S_2 to R_2. Based on these experiments and the results illustrated in Figure 6, we make the following observations:

- FTP goodput is decreased because of the use of batching.



(a) Link Based Batching (b) Mix Based Batching

Fig. 6. FTP Goodput

- Different batching strategies have different impact on the FTP goodput. In general, pool batching strategies (strategy S_5 to S_7) cause a worse FTP goodput than simple batching strategies (strategy S_1 to S_4).

- When the batching in the mixes is excessively aggressive, that is, when batching intervals are too long or threshold values too high, the batching interferes with the time-out behavior of TCP and FTP, and in some cases, FTP aborts. This is the case in particular for threshold triggered mixes with no cross traffic.

F. Sampling Interval Selection

1. Theory and Empirical Proof

From the evaluation above, we can see that the flow correlation attack based on Fourier spectrum is very effective. Sampling interval plays an important role in the effectiveness of Fourier spectrum since we calculate Fourier spectrum over a set of

packet average rate (i.e., the flow feature vector) in the sampling interval. In this section, we discuss how to select the sampling interval, τ, to maximize the effectiveness of flow correlation attacks. We still use FTP as an example for the discussion.

Corollary 1 *A FTP flow with round trip time RTT has a frequency component with the maximum power density at* $1/RTT$*. This frequency component is denoted as the feature frequency of the FTP flow.*

Please refer to Appendix A for the proof. The basic idea is that FTP uses a loop control mechanism. For most of the life time, a FTP flow acts on the information collected in each round trip time, thus demonstrates a strong periodicity at the round trip time RTT.

Based on Corollary 1, we have the following theory for the selection of sampling interval.

Theorem 1 *Assuming that a stable FTP flow on the input link of a mix has a round trip time RTT, to detect the output link of this FTP flow, we need to choose a sampling interval* τ *smaller than or equal to* $RTT/2$*, i.e.,*

$$\tau \leq \frac{RTT}{2} \tag{3.8}$$

Proof:

When we do sampling and calculate the average rate of a FTP flow during the sampling interval, the process corresponds to a *zero-order hold* [61] sampling process. From Corollary 1, we know that a FTP flow's feature frequency is at $1/RTT$, which we have to preserve for the best effectiveness of flow correlation attack. Nyquist's sampling theorem [61] tells us that to preserve this feature frequency, the sampling

rate $1/\tau$ should be at least 2 times the feature frequency. That is,

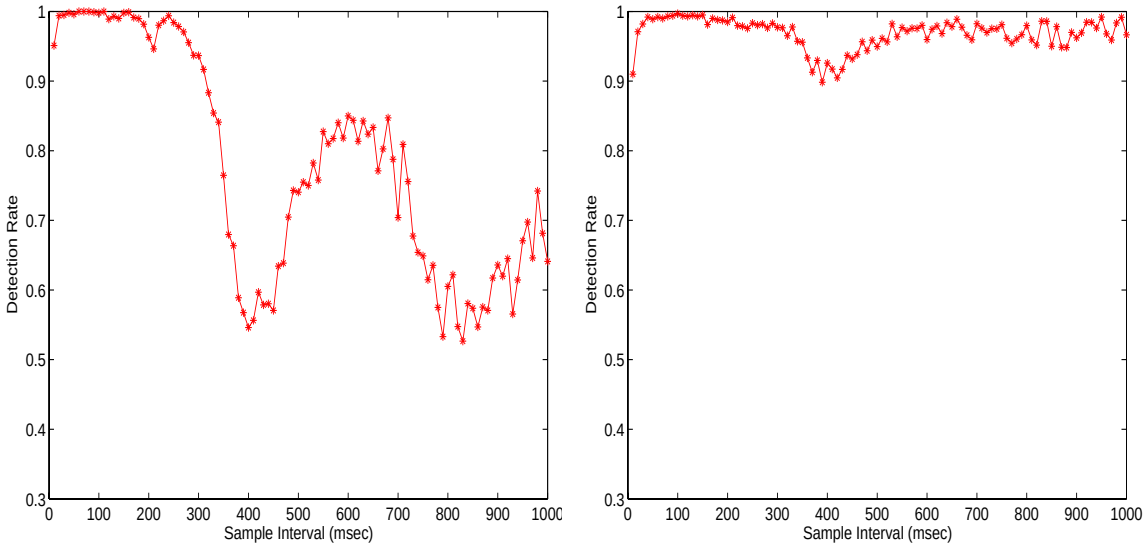
$$\frac{1}{\tau} \geq 2\frac{1}{RTT} \tag{3.9}$$

Thus

$$\tau \leq \frac{RTT}{2}$$

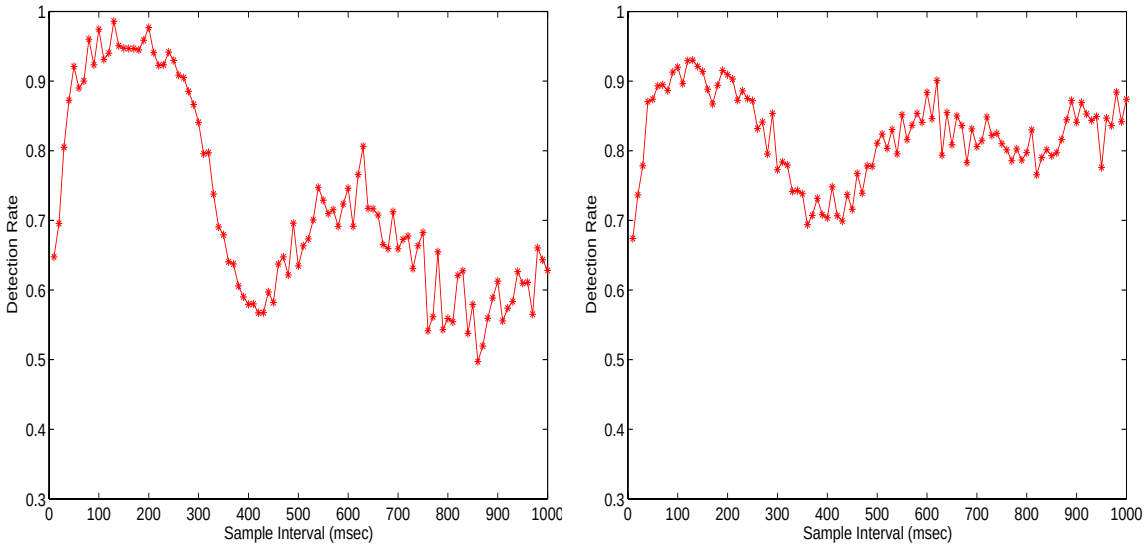
■

Approximately, we can apply Theorem 1 to all the strategies. Figure 7 and 8 show detection rate in terms of sampling interval. RTT of this FTP flow in question is around 300 *milliseconds*. We can see that the maximum detection rate does happens at $RTT/2 = 150$ *milliseconds*.



(a) $< s_0 >$ (b) $< s_2, 0.01 >$

Fig. 7. Detection Rate in Terms of Sampling Interval Based on Matched Filter



(a) $< s_0 >$ (b) $< s_2, 0.01 >$

Fig. 8. Detection Rate in Terms of Sampling Interval Based on Mutual Information

In theory, we can use any sampling interval smaller than half of RTT. In practice, because there exists all kinds of interference from mixes and operating systems, which may introduce high-frequency noise in frequency domain, we prefer to use a sample interval between $[RTT/2, RTT/4]$. In this way, the zero-order hold operator acts as a low-pass filter with frequency response

$$H(f,\tau) = e^{-\frac{j2\pi f\tau}{2}} \left[\frac{2\sin(\frac{2\pi f\tau}{2})}{2\pi}\right], \tag{3.10}$$

Recall τ is the sample interval. The main lobe of $|H(f,\tau)|$ is in the range $|f| < \frac{1}{\tau}$. Thus, our sampling process will smooth the original instantaneous rate and remove a lot of noise. This will help the flow correlation attacks. Figures 7 and 8 show the noise's influence on detection rate: when τ is much smaller than $RTT/2$, the detection rate becomes bad.

2. Discussion

One important point is that in practice, the adversary does not know the exact RTT of special FTP flows. But what the adversary can do is to a priori investigate the mix network and get a rough picture of possible FTP flow RTTs. The sampling interval can be half of the smallest of the possible RTTs or the one which gives the best detection rate. So we can see that flow correlation attacks can even help the adversary to find the concrete RTT of a FTP flow.

Figures 7 and 8 also demonstrate very complicated relationship between detection rate and sampling interval. A possible reason causing this is: although it has a feature frequency component at $1/RTT$, a FTP flow in reality is very complicated. There may exist some minor feature frequencies which are enough to differentiate a FTP flow from others. Figure 9 demonstrates a FTP flow's complicated power spectrum.

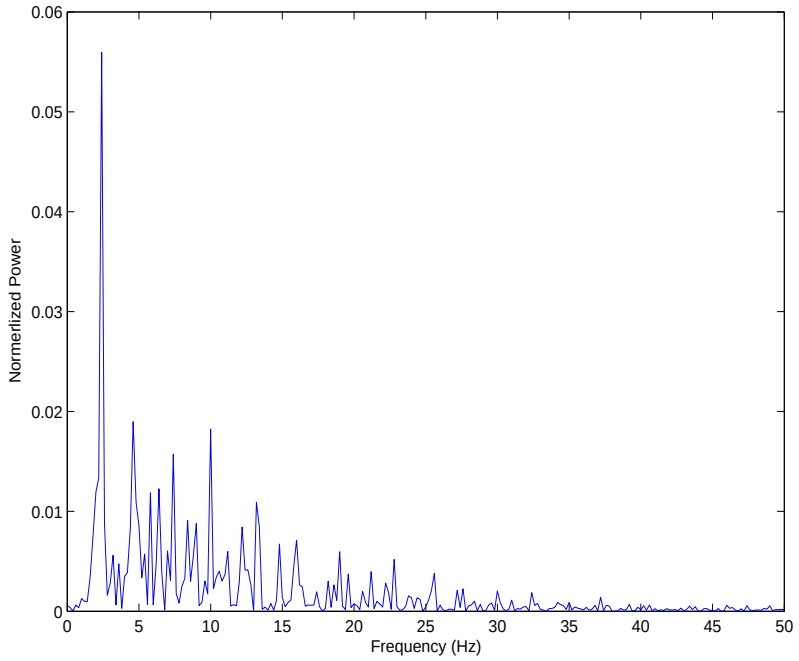


Fig. 9. Power Spectrum of a FTP Flow

G. A Countermeasure and Its Performance

From the discussion above, it is apparent that traditional batching strategies and reordering are not sufficient for mixes to effectively counter flow correlation attacks. Additional measures are needed. In this section, we introduce a relatively efficient and effective countermeasure and evaluate its performance in terms of FTP goodput.

1. Overview

To counter flow correlation attacks, the perfect way is that the input flow vector X_i should have the same distance with each of l output flow vectors $Y_1, \cdots, Y_l$,

$$d(X_i, Y_1) = \cdots = d(X_i, Y_j) = \cdots = d(X_i, Y_l), \tag{3.11}$$

and the only analysis strategy for an adversary would be to randomly guess which output flow is correlated to an input flow. This results in a detection rate of $\frac{1}{l}$.

Recall in (3.5), the distance is defined as the inverse of mutual information of an input flow vector and output flow vector. Thus, one way to achieve (3.11) is to make the input flow vector X_i have the same mutual information with each of l output flow vectors $Y_1, \cdots, Y_l$,

$$I(X_i, Y_1) = \cdots = I(X_i, Y_j) = \cdots = I(X_i, Y_l) \tag{3.12}$$

There are a few ways to achieve (3.12) based on the properties of mutual information:

- Make $I(X_i, Y_1) = \cdots = I(X_i, Y_j) = \cdots = I(X_i, Y_l) = 0$. That is, we make the input flow independent from the output flows. In general, this approach incurs a high cost since we have to shape the input flow and outflow completely different from each other by delaying traffic and inserting a large number of dummy

packets. Tarzan [21]) uses this method. It pads all the edges of a connected (but not fully connected) graph of the mix network to achieve a predefined traffic pattern.

- Make $I(X_i, Y_1) = \cdots = I(X_i, Y_j) = \cdots = I(X_i, Y_l) \neq 0$. There are two ways:
  - Make $I(X_i, Y_1) = \cdots = I(X_i, Y_j) = \cdots = I(X_i, Y_l) = H(X_i)$. One way to do this is to broadcast the input traffic to all the output links. This will increase an heavy overhead obviously.
  - Make $I(X_i, Y_1) = \cdots = I(X_i, Y_j) = \cdots = I(X_i, Y_l) < H(X_i)$. What we have to do is to have all the output flows look identical. Following this analysis, below we develop an adaptive mix output traffic shaping algorithm to reduce the overhead of dummy packets.

2. Adaptive Mix Output Traffic Shaping Algorithm

Because naturally the rates of traffic along all the output links of a mix are different, we have to appropriately insert dummy packets to make all the output flows behave in the same way. A challenge here is to insert a minimum number of dummy packets.

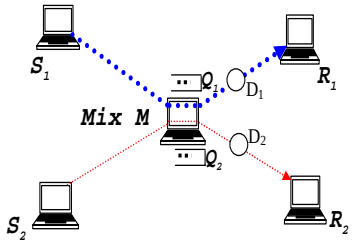


Fig. 10. Network Setup for the New Countermeasure

Such an output-control algorithm is illustrated in Figure 10. Mix M maintains two output queues, Q_1 for the link between Mix M and node R_1, and Q_2 for the link

between Mix M and node R_2. At any time, if each queue has a packet, they are sent out in some pre-defined order, e.g., the packet in Q_1 first and the packet in Q_2 second. By doing so, one of the two queues will be always empty. Let us say, for the moment, that Q_2 is empty. A deadline is assigned to each packet waiting in Q_1. If a packet in Q_1 reaches its deadline, a dummy packet will be generated for Q_2. Then, the payload packet from Q_1 and the dummy packet from Q_2 are sent out in the predefined order. A dummy packet will also be generated for Q_2 if the queue length of Q_1 goes beyond a preset threshold. In this way, we can ensure a maximum delay on each packet, and we also guarantee that neither queue will overflow.

```
Data        : queues, in which packets are kept in deadline order by the mix
Result      : synchronized flows out of the mix
while (1) do
    if (Q1.Length > 0) and (Q2.Length > 0) then
        send the first packet from Q1;
        send the first packet from Q2
    else
        if (Q1.Length > 0) then
            if (Q1.FirstPacket.Deadline > CurrentTime) or (Q1.Length > Q1.Threshold)
            then
                send the first packet from Q1;
                send a dummy packet for Q2
            end
        else
            if (Q2.Length > 0) then
                if (Q2.FirstPacket.Deadline > CurrentTime) or (Q2.Length >
                Q2.Threshold) then
                    send a dummy packet for Q1
                    send the first packet from Q2;
                end
            end
        end
    end
end
```

Fig. 11. Algorithm for Output Traffic Control

Figure 11 gives the new countermeasure algorithm on Mix M for the anonymity system in Figure 10. We can see that the output traffic of the Mix is now synchronized, and the adversary cannot observe any difference among the output flows.

This method can be easily extended and optimized for more complicated cases. The number of virtual output links of a mix can be very large since we assume a peer-to-peer mix network. Since we only maintain virtual queues, the overhead is limited. In the case of a large network with a small number of flows, there still needs to be a lower bound LB_Q of the number of virtual queues required for each mix to maintain anonymity. In other words, we do not necessarily need to synchronize every output link when traffic is slow, but we will synchronize a minimum number LB_Q of links. For example, if there is one virtual queue with a packet whose deadline is reached, we have to send out dummy packets to the other $LB_Q - 1$ virtual links.

Output traffic control is not new and has been proposed for example in [62], where messages at the output ports are forwarded periodically[3]. The algorithm in Figure 11 is more efficient and probably more effective than the approach described in [62]. It is more efficient because packets are forwarded based on each queue's status: once each queue has payload packets, the first packet in each queue is sent out and packets suffer smaller delay at Mixes. It is likely more effective because periodic traffic patterns are very difficult to generate with sufficient accuracy. We showed in NetCamo [51, 63], for example, how high-accuracy traffic analysis can easily break periodic link padding schemes.

3. Performance Evaluation of Output Traffic Control

We are interested in how traffic flows traversing a mix affect each other. In particular, we evaluate the TCP performance. Again FTP is used as an example in the evaluation.

Figure 12 gives the FTP goodput measurement for our new scheme for the net-

[3]The paper is too vaguely written for us to figure out exactly what forwarding mechanism is used.

work setup in Figure 10. We set the threshold of each queue at *50* packets. The path from S_2 to R_2 has FTP traffic and UDP traffic of 400pps. Cross traffic in Figure 12 refers to the UDP traffic along the path S_1 to R_1. Both paths have a propagation delay of *0.3* second. We have the following observations from these experiments.

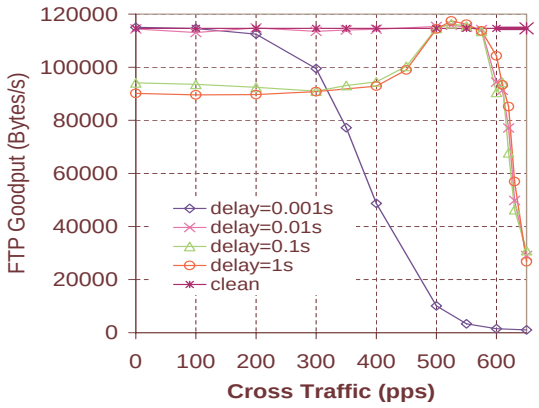


Fig. 12. FTP Goodput Using Output Traffic Control ("clean" means no output traffic control)

While not evident from Figure 12, the observed detection rate of the correlation attack is 50% in all the cases when the new countermeasure is used. This is expected, as the new method can guarantee a detection rate of $1/LB_Q$ where $LB_Q = 2$ in this case.

The goodput for the clean FTP is 114,628.83 bytes/s. When the delay parameter is set to 0.01s, the same goodput is achieved as long as the cross traffic is less than 525 pps. This is very significant. It indicates that, once the delay parameter is properly selected, our new method can achieve high throughput (as high as the case without mix) while guaranteeing a low detection rate.

For the cases of delay equal to 0.01s, 0.10s, and 1.00s, right after the cross traffic goes beyond 525 pps, all have their goodput drop rapidly. This is due to the fact

that the cross traffic is so heavy that the FTP's TCP protocol detects congestion and adapts accordingly.

It is also interesting to note, that when the cross traffic is low and the value of delay parameter is large (say, the cross traffic is less than 500 pps and delay is equal to 0.10s or 1.00s), the goodput is low (about 93,000 bytes/s). This is consistent with intuition: if the cross traffic is low and delay is large, then the traffic of our FTP flow may have to wait longer than in other cases, resulting in a reduction of goodput.

Finally, in the case when the value of delay parameter is small, say, equal to 0.001s, the curve of goodput is monotonically decreasing. In this case, it is likely that a packet from the FTP flow will be transmitted due to the deadline expiration, rather than the arrival of a packet from the cross traffic. Thus, the cross traffic always contributes negatively to the goodput performance here by creating dummy packets.

H. Summary

We formally model the behavior of an adversary who launches flow correlation attacks. In order to successfully identify the path taken by a particular flow, the attacker measures the similarity of traffic flows. Two classes of correlation methods are considered, namely *time-domain* methods and *frequency-domain* methods. In the *time domain*, for example, statistical information about rate distributions is collected, and *mutual information* is used to identify the traffic similarity. Similarly, in the *frequency domain*, we identify traffic similarities by comparing the *Fourier spectra* of timing data. Our experiments indicate that mixes with many currently used batching strategies are weak against flow-correlation attacks, in the sense that attackers can easily determine the path taken by a protected flow.

We measure the effectiveness of a number of popular mix strategies in countering

flow correlation attacks. Mixes with any tested batching strategy may fail under flow-correlation in the sense that, for a given flow over an input link, the adversary can effectively detect which output link is used by the same flow. We use *detection rate*, the probability that the adversary correctly correlates flows into and out of a mix. as the measure of success for the attack. We will show that, given a sufficient amount of data, known mix strategies fail, that is, the attack achieves close to 100% detection rate. This remains true, even in batching strategies that sacrifice QoS concerns (such as a significant TCP goodput reduction) in favor of security.

While many mix strategies rely on other mechanisms in addition to batching alone, it is important to understand the vulnerabilities of batching. In fact, for a given accuracy of collected data, the effectiveness of such attacks depends primarily on the amount of collected data, i.e. on the length of the observation interval. In our experiments, we illustrate this dependency between attack effectiveness for various batching strategies and the amount of data at hand. These results should guide designers of anonymous communication systems in the educated choice of strategy parameters, such as for striping or for path rerouting.

We have analyzed mix networks in terms of their effectiveness in providing anonymity and quality-of-service. Various methods used in mix networks were considered: seven different packet batching strategies and two implementation schemes, namely the link-based batching scheme and mix-based batching scheme. We found that mix networks that use traditional batching strategies, regardless of the implementation scheme, are vulnerable under flow correlation attacks. By using proper statistical analysis, an adversary can accurately determine the output link used by traffic that comes to an input flow of a mix. The detection rate can be as high as 100% as long as enough data is available. This is true even if heavy cross traffic exists. The data collected in this chapter should give designers guidelines for the development

and operation of mix networks.

The failure of traditional mix batching strategies directly leads us to the formation of a new packet control method for mixes in order to overcome their vulnerability to flow correlation attacks. Our new method can achieve a guaranteed low detection rate while maintaining high throughput for normal payload traffic. Our claim is validated by extensive performance data collected from experiments. The new method is flexible in controlling the overhead by adjusting the maximum packet delay.

CHAPTER IV

MODELING THE EFFECTIVENESS OF TIMING ATTACKS

A. Motivation

In Chapter III it has been shown that flow correlation attack could seriously degrade the effectiveness of flow-based anonymity communication systems. In this chapter, we describe a framework for the analytical evaluation of mix networks under flow correlation attack. Our analytical model provides anonymity network designers a guideline in assessing the anonymity provided by the network.

The modeling framework proposed in this chapter is very general. It is not only applicable to the mixes using batching strategies described in Chapter III but also applicable to other types of mixes as well, such as stop-and-go mixes [36] and continuous-time mix [35]. The concept of continuous-time mix is introduced by Danezis in [35]. Danezis proved that the optimal mix strategy (i.e. the strategy that maximizes anonymity) for continuous-time mix is the *Exponential Mix*, i.e. a Stop-and-Go Mix that delays packets individually according to an exponential distribution. In this chapter we also applied our modeling framework to the Exponential Mix.

B. Flow Correlation Attack Revisited

In this section, we will first revisit the flow correlation attack method based on mutual information and then formulate the modeling problem.

1. Problem Definition

Define a traffic flow as a series of packets exchanged between a sender (Alice) and a receiver (Bob) in the network[1]. For the attacker who reconstructs the path of a flow, a fundamental question must be answered: *Given a flow, f, into a mix or mix network, which output link does the flow take?* For example, consider the simplified scenario in Figure 13, where f', c'_1,..., c'_4 are output flows of input flows f, c_1, ..., c_4, respectively. The goal of the adversary is to determine whether input flow f, after passing through the mix, goes through $link_{M \to R_1}$ (link from mix M to R_1) or $link_{M \to R_2}$.

Flow f is not alone in the mix network: First, it is typically not alone on the input link to the mix. Second, significant cross traffic either naturally exists, or is generated by the mix network. We therefore have to assume that there is cross traffic (for example, denoted by c_1, c_2, c_3, and c_4 in Figure 13) interfering with the correlation analysis. In the experiments we will focus on scenarios where long-term average traffic rates on all the output links (for example, $link_{M \to R_1}$ and $link_{M \to R_2}$ in Figure 13) are identical. This renders simple statistical attacks, such as average traffic rate based attacks in [33], invalid. In this section, we will always use the setup of Figure 13 as an example to demonstrate our analysis technique.

2. Flow-Correlation Attack Algorithm

To determine which output link the input flow f uses, an adversary has to collect information and make a determination based on some statistical analysis. In this chapter, we consider the case where the adversary adopts the method based on mutual

[1]Such a flow can be either a TCP connection or a segment of UDP packets that are part of a VOIP connection, or any other sequence of packets that represent a communication session. In the experiments described later we are using the traffic from a FTP session as the flow.

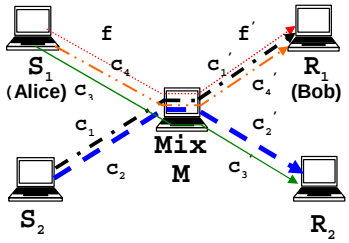


Fig. 13. Mix Setup and Flow Configuration

information of the input flow and the aggregate flows on each output link and chooses the output link whose aggregated flow has the biggest mutual information with the input flow. Specifically, using Figure 13 as the example, the adversary will collect a traffic sample from both input and output links. Then, she calculates mutual information $I(f, l_{M\to R_1})$ and $I(f, l_{M\to R_2})$, where $l_{M\to R_1} = f' + c'_1 + c'_4$ is the aggregated flow on $link_{M\to R_1}$ and $l_{M\to R_2} = c'_2 + c'_3$ is the aggregated flow on $link_{M\to R_2}$. A decision will then be made in the following way: if $I(f, l_{M\to R_1}) > I(f, l_{M\to R_2})$, the adversary will declare $link_{M\to R_1}$ as f's output link. Otherwise, $link_{M\to R_2}$ will be chosen.

The rationale for comparing mutual information is that the correct output link carries the flow, embedded in cross traffic: The input flow and the aggregate output flow are therefore not independent and thus display a non-zero mutual information. In Figure 13, it is statistically likely that input flow f is more similar to the aggregated flow $l_{M\to R_1}$ on $link_{M\to R_1}$ than the aggregated flow $l_{M\to R_2}$ on $link_{M\to R_2}$ since f' is in $l_{M\to R_1}$ [2].

[2]We tacitly assume that incoming flows are unrelated and thus statistically independent from each other.

3. Mutual Information Estimation

From the discussion above, we can see that an accurate estimation of mutual information of input and output traffic is critical for the effectiveness of the type of flow-correlation attacks considered here. In order to develop a model for the effectiveness of attacks and of counter measures it is therefore important to in turn develop a model for the accuracy of the mutual information estimator available to the attacker.

We assume that the adversary uses the following packet counting scheme to estimate the mutual information between the input flow f and any aggregated flow l on an output link.

- The adversary collects (by, say, monitoring the packets on a link) a sample of traffic traces of the input flow f and the aggregated flow l.
- Each traffic trace is divided into time segments. The length of the segments is T, which is denoted as *sampling interval*. The number of the segments in a trace is denoted N and is called *sample size* in this chapter.
- The number of packets in each segment of both traces is counted. Let a and b represent the random variables of the numbers of packets in a segment of traffic trace from an input flow and output link aggregated flow, respectively.

  Two time series can be obtained:

  - The input flow packet count time series

    $$f_T = \{a_1, \cdots, a_N\}$$

    is the series of number of packets a_i in the i^{th} segment of the input traffic

flow trace. Note that $a_i \in \{0, \cdots, r\}$, where

$$r = \max(a) \ . \tag{4.1}$$

– The output link aggregated flow packet count time series

$$l_T = \{b_1, \cdots, b_N\}$$

is the series of number of packets b_i in the i^{th} segment of the output link traffic flow trace. Note that $b_i \in \{0, \cdots, s\}$, where

$$s = \max(b) \ . \tag{4.2}$$

- Based on the time series f_T and l_T, a joint time series is developed as follows:

$$J_T = \{(a_1, b_1), \cdots, (a_N, b_N)\} \tag{4.3}$$

where a_i and b_i are elements in time series of f_T and l_T, respectively.

- Finally, the mutual information of the input flow and the output link flow is estimated by the following formula:

$$\hat{I}(f, l) \approx \sum_{a=0}^{r} \sum_{b=0}^{s} \hat{p}(a, b) \log \frac{\hat{p}(a, b)}{\hat{p}(a)\hat{p}(b)} \tag{4.4}$$

where $\hat{p}(a)$, $\hat{p}(b)$, and $\hat{p}(a, b)$ are the frequencies of a, b, and (a, b) within f_T, l_T, and J_T, respectively.[3]

[3]In the following we will need to distinguish the frequency of an event as sampled from the collected data from the underlying distribution of the same event. We use the notation $\hat{p}$ for the frequency and p for the underlying distribution. Similarly, we use $\hat{I}$ to denote the estimated mutual information based on the sampled time series f and l. We denote the actual mutual information based on the underlying distributions as I.

C. Derivation of Detection Rate

The *detection rate* v measures the effectiveness of the attack and is defined as the probability that the adversary correctly recognizes the output link of the input flow f. Without loss of generality, we assume that the input flow f uses the mix's first output link i.e. $link_{M \to R_1}$ in Figure 13. Based on the algorithm described in Section B.2, the general formula to compute detection rate is as follows:

$$v = Pr\left(\hat{I}(f, l_{M \to R_1}) > \hat{I}(f, l_{M \to R_2}),\ \cdots, \hat{I}(f, l_{M \to R_1}) > \hat{I}(f, l_{M \to R_n})\right) \tag{4.5}$$

In the following we derive a formula to estimate v as a function of the type of traffic in the network and of the mix's traffic perturbation strategy.

1. Distribution of the Mutual Information

The attacker makes her decision by comparing the estimated mutual information, based on the sampled data, instead of actual mutual information based on the underlying probability. The effectiveness (i.e. the detection rate) of the attack therefore suffers if insufficient data is available. In the following we will show how the amount of available trace data affects the detection rate of the attack. For this we will show how the estimated mutual information based on histograms of collected trace data affects Equation (4.6) and (4.7). We will also show that the attack is correct; that is, with sufficient trace data available the effectiveness of the attack approaches 100%.

To calculate the detection rate by using (4.5), we need to obtain the probability distribution function of the mutual information estimation $\hat{I}(f, l)$ in (4.4). According to the Central Limit Theorem, for a sufficiently large sample size N, $\hat{I}(f, l)$ should satisfy a normal distribution. To obtain the distribution function, we therefore only need to estimate $\hat{I}(f, l)$'s mean and variance, which are given in Lemma 1 and Lemma

2, respectively. Their proofs can be found in Appendix B.

Lemma 1 *The mean of the mutual information estimation* $\hat{I}(f,l)$ *is given by*

$$E(\hat{I}(f,l)) \approx I(f,l) + \frac{(r-1)(s-1)}{N} \tag{4.6}$$

where $I(f,l)$ *is the original mutual information, and* r *and* s *are defined in (4.1) and (4.2), respectively.*

As described in (4.1) and (4.2), the value for r and s describe the range of possible sample values observed at the input and output ports, respectively. For $10Mb/sec$ links, the maximum numbers of packets observed over a 10 $msec$ interval could be about 10, giving rise to a value of 10 for r and s.

Lemma 2 *The variance of the mutual information estimation* $\hat{I}(f,l)$ *is given by*

$$var(\hat{I}(f,l)) \approx \frac{C_{f,l}}{N} \quad . \tag{4.7}$$

The constant $C_{f,l}$ *is defined as follows*

$$C_{f,l} = \sum_{a,b} p(a,b) \left(\log \frac{p(a,b)}{p(a)p(b)} \right)^2 - \left(\sum_{a,b} p(a,b) \log \frac{p(a,b)}{p(a)p(b)} \right)^2 \tag{4.8}$$

where $p(a,b)$ *is the original probability distribution of* (a,b).

2. Detection Rate Theorem

Based on the distribution function of the estimated mutual information, we can calculate the detection rate by the following theorem. Its proof can be found in Appendix C.

Theorem 2 *For a mix with any number of output links, the detection rate,* v*, is*

given by

$$v \approx 1 - \sqrt{\frac{C_{f,l_{M\to R_1}}}{N}} \times \int_{-\infty}^{-I(f,l_{M\to R_1})\sqrt{\frac{N}{C_{f,l_{M\to R_1}}}}} \mathcal{N}(0,1)dx \tag{4.9}$$

where N is the sample size, $I(f, l_{M\to R_1})$ is the mutual information of the input flow f and its corresponding output link aggregated flow $l_{M\to R_1}$, $\mathcal{N}(0,1)$ is the density function of the standard normal distribution, and $C_{f,l_{M\to R_1}}$ is a constant.

We can make a number of observations on Theorem 2.

- No assumptions are made in Formula (4.9) about the batching strategy of the mix or about the network topology. Theorem 2 is therefore valid for mix networks with arbitrary topology. Similarly, no assumption is about the type of traffic or about the amount of cross traffic. As a result, Theorem 2 is very general.

- Clearly, the detection rate is an increasing function of sample size N. Thus, when sample size N increases, the detection rate approaches 100%. This formally proves the intuitive fact that any mix network will fail and cannot maintain anonymity if the adversary has access to a sufficient amount of traffic data.

3. Joint Distribution of (a, b)

In Theorem 2, both constant C and the original mutual information $I(f, l)$ depend on the joint distribution function $p(a, b)$, which in turn is defined by the strategy of the mix network and the type and amount of traffic in the network. It can be estimated by two methods.

a. Direct Estimation

We can estimate $p(a, b)$ directly from the time series J_T defined in (4.3). Specifically, from J_T, a frequency distribution of (a, b) can be established. Then, we can use

standard statistical techniques to obtain an estimation of $p(a, b)$. See [64] for details.

b. Estimation based on Poisson Assumption

The joint distribution $p(a, b)$ can be calculated as follows:

$$p(a, b) = p(b|a)p(a). \tag{4.10}$$

To calculate the conditional probability $p(b|a)$ in (4.10), we need to apply proper queuing models in accordance to mixing strategies. For example, if the input flow is assumed to be a Poisson process, for a Simple Proxy S_0, a M/D/1 queuing model should be used. For a Timed Mix S_2, we should use an embedded Markov chain. For an exponential mix, we should use a $M/M/\infty$ queue. Please see Appendix D, Appendix E, and Appendix F for a detailed derivation of the probability from the models.

D. Evaluation

In this section, we assess the accuracy of methods we developed to estimate detection rate and to evaluate the performance of mix networks that are under flow-correlation attacks. Unless we specify otherwise, we use the popular ns-2 network simulator [65] for the experimental evaluations. In this section, we first describe our results on mix strategies based on batching and then describe our results for the exponential mix.

1. Evaluation on Batching-based Mix

a. Failure of Mix Network

Before we proceed to evaluate the accuracy of our predictive models for single mixes, we provide data to validate the claim made in Theorem 2: for any size of mix network,

given a sufficient amount of traffic data, the flow correlation attack will ultimately achieve a detection rate of 100%.

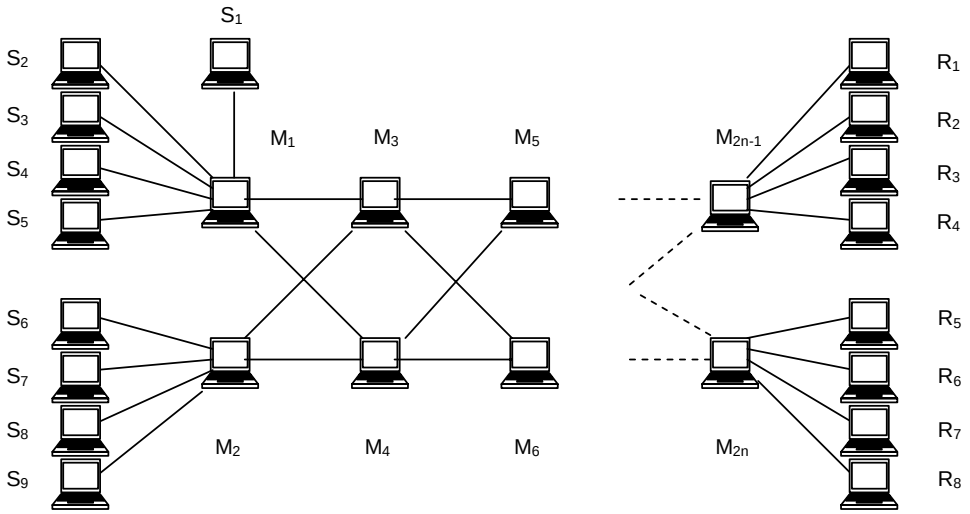


Fig. 14. Topology of Mix Network

The network topology for this experiment is shown in Figure 14: The senders and receivers are connected by a stratified cascade of $2n$ mixes. Each flow traverses n mixes to reach its receivers. Each link between mixes has a bandwidth of 10Mbit/s and propagation delay of 10ms. The senders and receivers are connected to the mix network via links with bandwidth of 100Mbit/s and propagation delay of 1ms. There are nine flows in the network: flow $S_1 \rightarrow R_1$, flow $S_2 \rightarrow R_1$, flow $S_3 \rightarrow R_2$, flow $S_4 \rightarrow R_5$, flow $S_5 \rightarrow R_6$, flow $S_6 \rightarrow R_3$, flow $S_7 \rightarrow R_4$, flow $S_8 \rightarrow R_7$ and flow $S_9 \rightarrow R_8$ respectively. Flow $S_1 \rightarrow R_1$, flow $S_2 \rightarrow R_1$, and flow $S_3 \rightarrow R_2$ traverse odd-numbered mixes only, flow $S_8 \rightarrow R_7$ and flow $S_9 \rightarrow R_8$ traverses even-numbered mixes only, flow $S_4 \rightarrow R_5$, flow $S_5 \rightarrow R_6$, flow $S_6 \rightarrow R_3$, and flow $S_7 \rightarrow R_4$ take the zigzag path between the two horizontal lines of the mixes, and flow $S_1 \rightarrow R_1$ is the flow of interest to us. To ease the control of noise traffic rate, only flow $S_1 \rightarrow R_1$ is

TCP traffic from a FTP session and the other flows are UDP streams with Poisson arrivals. The average traffic rate to all the receivers are adjusted to roughly five times the average rate of flow $S_1 \rightarrow R_1$. The mixes in network are all timed mixes with a batch interval of 10ms.

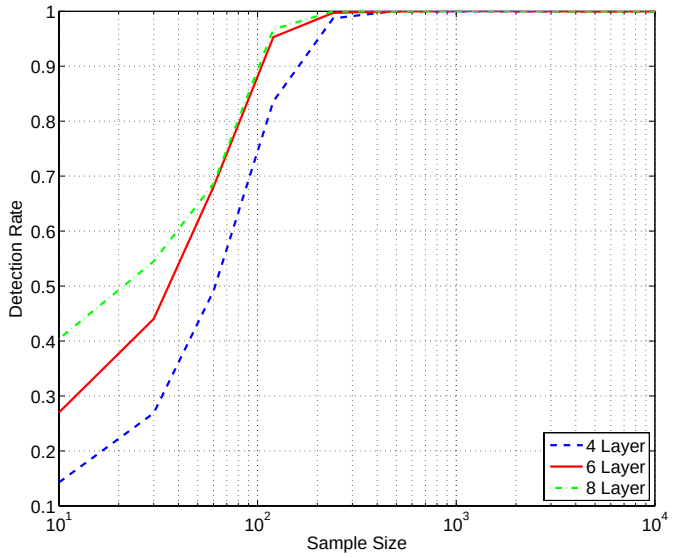


Fig. 15. Effectiveness of Flow Correlation Attack vs size of the mix network (Sample size: number of sample intervals of length 10ms)

Figure 15 shows the detection rate of a flow correlation attack for different numbers of mixes in the network. The length of sampling segments is set to be 10ms. We make the following observations:

- As stated in Theorem 2, the flow correlation attack remains effective as the network size grows.
- In fact, the flow correlation attack achieves higher detection rates for larger mix networks! While we have not analyzed this effect in detail, we conjecture that

the reason is the loop-control mechanism of TCP: The more mixes are on the path, the larger is the burstiness of the TCP flow from Alice to Bob. In turn, this makes Alice's flow more recognizable compared with the background noise traffic.

b. Estimation Error of Detection Rate

In the last section, we derived formula (4.9) to compute the detection rate. This formula is an estimated one due to, at least, the following reasons:

- Error in the Taylor Expansion: In Formula (B.2) (in Appendix A), the computation of mutual information is estimated by a truncated Taylor expansion, which introduces a certain error, given the limited number of terms.
- Error in $p(a, b)$ Computation: As discussed in Section C.3, we introduce two different methods to estimate $p(a, b)$. Either one of them will contribute some error in the estimation of detection rate.

In this subsection, we examine the accuracy of our estimation in order to ensure the performance data we derive in this chapter is meaningful. We use the one-mix network setup in Figure 13.

We define e, the estimation error of detection rate, as follows:

$$e = \frac{|\text{ approximated detection rate} - \text{ exact detection rate}|}{\text{exact detection rate}} \tag{4.11}$$

We obtain the exact detection rate in (4.11) by simulation. In all the experiments mentioned earlier, the traffic average rates on all output links are assumed to be identical. This in turn prevents attacks based on analyzing average traffic rates. The traffic type of payload flow can be either UDP or TCP, with traffic rates of 100 Kbps and 80Kbps, respectively. Compounded with noise traffic, each output link has an

aggregated traffic rate 500 Kbps. The length T of sampling segments is set to be 10ms.

Figure 16 depicts the estimation error in terms of sample size. From this figure, we can make a number of observations:

- For all the traffic types and batching strategies, if the sample size is small (say, less than 100), the estimation error may be more than 5%. Fortunately, the estimation error diminishes and eventually approaches zero when the sample size is sufficiently large. For example, when the sample size is 200, which corresponds to a sample of two seconds, the estimation error for all cases is below 4%. This observation suggests our estimation methods will be quite useful in practical situations.

- Generally speaking, the direct estimation method results in smaller error than the estimation by Poisson assumption. This is to be expected as the traffic on the Internet is not inherently Poisson [66].

- In comparison with the networks using different batching strategies, the estimation errors appear to be similar. However, when we compare networks with different traffic types, the one with UDP traffic seems to result in less error. This is, perhaps, due to the difficulty in statistical modeling of TCP traffic.

c. Detection Rate

Figure 17 shows the detection rate in terms of sample size. We can make the following observations:

- In all cases (of different batching strategies and traffic types), the detection rate approaches 100% when the sample size is sufficiently large. This demonstrates

the challenges posed by flow-correlation attacks and validates the claim we made in Section C.2.

- Even when the sample size is not too large, (say, about 200), the detection rate can be relatively high, typically more than 90% for the shown cases.
- The implication of the above two observations is serious: A mix network would fail to provide anonymity under the flow-correlation attacks if the adversary is allowed to collect its sample for a time period of sufficient length. Note that, by using our formulae, a system designer can predict the situations where the failure may occur and invoke other countermeasures (such as shortening the flow life time, utilizing channel hoping in wireless networks, etc).

d. Minimum Sample Size

As mentioned earlier, one way to provide a countermeasure against flow-correlation attacks is to reduce the flow life time and so prevent the adversary from obtaining a sample that is sufficiently large. To provide some guidelines on this matter, we measure m, the minimum sample size needed in order for the adversary to achieve a given level of detection rate.

In Figure 18, we compare the systems under the measure of the minimum sample size with different traffic type and batching strategies. A number of observations can be made:

- In order to increase the detection rate, a larger minimum sample size is required. For example, in Figure 18 (a), for the case of TCP traffic, when the detection rate requirement changes from 95% to 99%, the minimum sample size increases from about 130 to almost 200. While this observation is expected, our formulae provide useful guidelines for system parameter selection here.

- For UDP traffic, it seems that the batching strategy may not be effective in terms of the minimum sample size. In other words, the difference between Figures 18 (a) and (b) is not significant for the case of UDP traffic.

- However, the effectiveness of batching appears to be much more interesting for the TCP traffic. We observe that the minimum sample size actually *reduces* when we switch the network from using no batching (strategy S_0) to using batching strategy S_2. That is, Figure 18(a) shows that when a mix network does not use any batching and traffic is TCP, a sample of about 290 is needed to achieve a detection rate of 99%, while in Figure 18(b), we see that for a network that does use batching and has the similar rate of TCP traffic, the size of sample is reduced to 210 to achieve the same detection rate of 99%. This is counter intuitive: If we take sample size a measure for the level of difficulty for an adversary, our data show that the adversary has more difficulty to achieve the required detection rate in a network without batching than one with batching. This phenomenon can be explained. When batching is performed, the TCP traffic may start oscillation. Consequently, this oscillation provides a much better signature for the adversary to use in the correlation of traffic on input and output links. We believe this is an important discovery that justifies the necessity of our modeling and evaluation in this chapter. We strongly suggest to always make a thorough evaluation for anonymity systems to be deployed.

2. Evaluation of the Continuous-time Mix

a. Failure of the Continuous-time Mix

Experimental Results: We first show the failure of the continuous-time mix in reality. We implemented the continuous-time mix in Timesys/RealTime Linux op-

erating system [59]. The mix control module that performs the delay function is integrated into the Linux firewall system [60] using Netfilter. The bandwidth of all links is $10Mb/s$. The average delay of the continuous-time mix in this subsection is 20ms.

We consider two cases here: (1) All the traffic is TCP. TTCP [67] is used to generate TCP traffic. There are five TCP flows to Receiver Bob and R_2 respectively. One of the flow to Bob is from Alice; (2) All the traffic is UDP and Poisson. The rate of traffic to Bob and R_2 are around 650 packets/s and the rate of traffic from Alice to Bob is around 200 packets/s.

The result of flow correlation attacks on the continuous-time mix in the testbed is shown in Figure 19. We make the following observations:

- For the continuous-time mix, flow correlation attacks can achieve high detection rate given access to sufficient data. Detection rate increases with the amount of data available. This result and the experimental results in Chapter III empirically give evidence for the correctness of our detection rate formula (4.9).

- Experiments with TCP flows show much higher detection rates than experiments with UDP traffic. The reason for this is very likely that TCP has a significantly stronger signatures, which are easy to detect than UDP traffic.

- Flow correlation attacks can be very efficient. Recall that we use a sampling interval of 10ms. Thus, a sample size of 3000 corresponds to a sample length of 30 seconds. Given access to 30 seconds of data, an attacker can achieve a detection rate of 100% in the case of TCP traffic and a detection rate of around 90% even with such a high load of noise traffic.

Modeling Accuracy by Simulation: We use the ns-2 simulator to evaluate the accuracy of the model described in Appendix F. We consider two cases of traffic

load: *light traffic load* and *heavy traffic load*. We distinguish the two cases to assess the accuracy of the $M/D/1$-based model for the output port, as in the case of light traffic load, the second queue can be largely ignored. In the experiments, we vary the link capacity instead of traffic load, with a 1Tb/s and 5Mb/s capacity for the light and heavy load respectively. The traffic of Alice's flow is Poisson distributed with an average rate of 100 packet/s. The noise traffic to Receiver Bob and R_2 are also Poisson distributed with average rate 400 and 500 packet/s respectively. The link delay between the mix and the receivers is 50ms. The links between senders and mix have 100Mb/s bandwidth and 1ms delay. The continuous-time mix's average delay is set to 20ms.

Figure 20 compares the results obtained from our model and by simulation. We make the following observations:

- The results from the model well match the simulation results. For example, the mean estimation error is only around 5% and the estimation error never exceeds 15%.

- The detection rate is higher in the case of light traffic load. This is because for heavy traffic load the aggregate traffic rate is comparable to the link bandwidth. The output queue will therefore build up and so further perturb the outgoing traffic. This reduces the dependence between the sender's outbound flow and the receiver's inbound flow. Nevertheless, this effect is accurately captured by the $M/D/1$ queue model in this chapter.

b. Impact of Parameter of Continuous Mix

The continuous-time mix with exponentially distributed delay has a single parameter: the average delay t_{avg}. Figure (21) shows the relationship between the detection rate

and the average delay for sample size 60, 480, 3840, and 30720. The sampling interval is set to 10ms. These sample size correspond to sample length of 0.6s, 4.8s, 38.4s and 307.2s. We make the following observations:

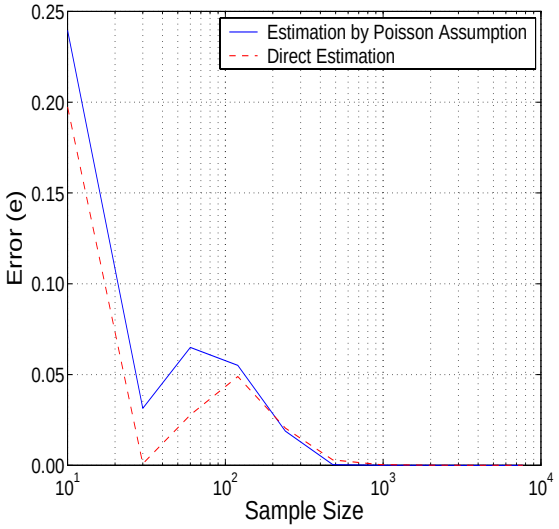
- Detection rate decreases as t_{avg} increases for each case of sample size. This is to be expected: because when t_{avg} increases, the probability for a packet held in the delay module or an incoming packet to leave the mix in the same sample interval will decrease. In turn, this will cause a smaller dependence between the flow of interest and the aggregate traffic containing the flow.

- Detection rate increases as the sample size increases when we fix t_{avg}. This is consistent with the results in Figure 20. Again, it is because the increase of the amount of data for detection will cause more accurate estimation of dependency between the flow of interest and the aggregate traffic flows.

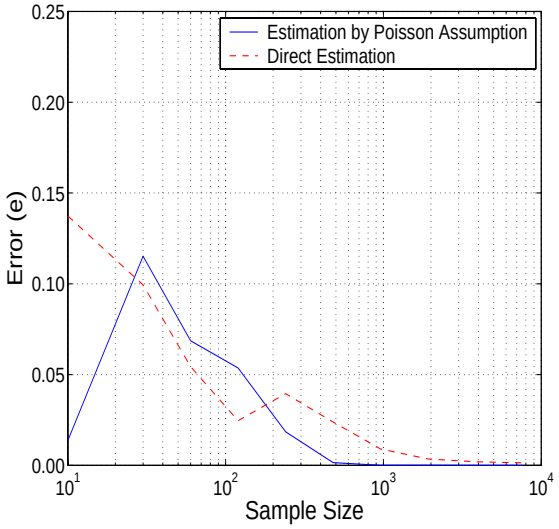
E. Conclusion

We have analyzed the anonymity of mix networks under flow correlation attacks. We present a formal model of the adversary and derive the detection rate as a performance measure of the system. Our theory discloses the underlying principle of flow-correlation attacks. As such, our results are the first to illustrate the quantitative relationship among system parameters, such as sample size, noise level, payload flow rate, and detection rate. Our analysis quantitatively reveals that flow-correlation attacks can seriously degrade anonymity in mix networks. Consequently, our results also provide useful guidelines for the design of future anonymous systems where additional countermeasures must be taken.

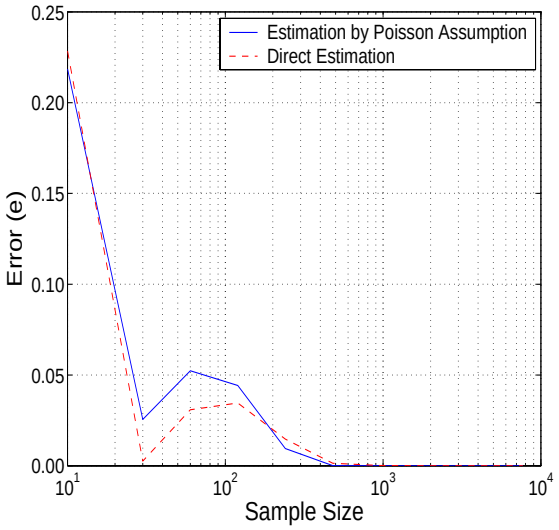
In the following, we will describe another class of traffic analysis attack, the flow separation attack.



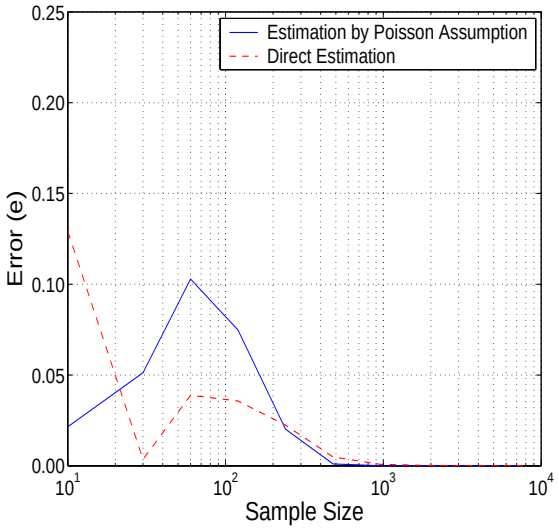
(a) Mix Network without Batching (Strategy S_0) and with UDP Traffic



(b) Mix Network without Batching (Strategy S_0) and with TCP Traffic

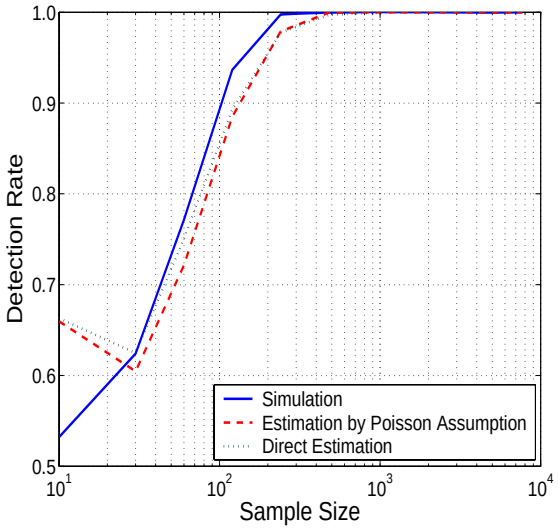


(c) Mix Network with Batching (Strategy S_2) and with UDP Traffic

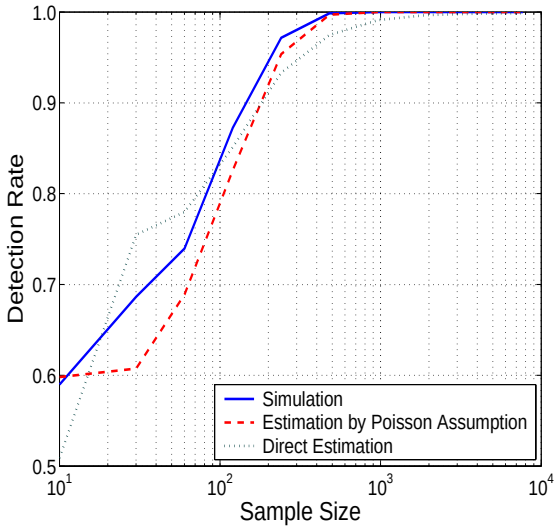


(d) Mix Network with Batching (Strategy S_2) and with TCP Traffic

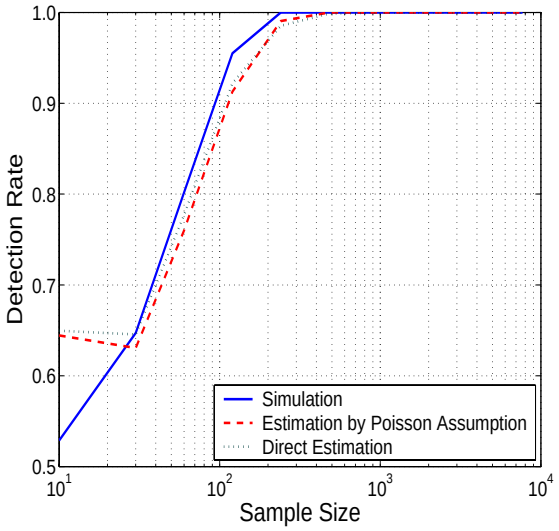
Fig. 16. Estimation Error of Detection Rate (Sample size: number of sample intervals of length 10ms)



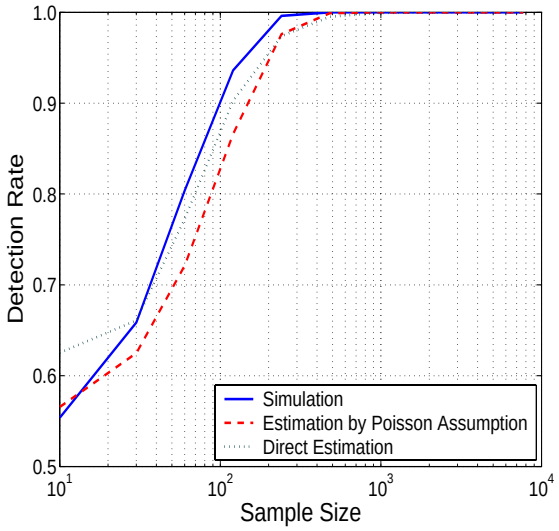
(a) Mix Network without Batching (Strategy S_0) and with UDP Traffic



(b) Mix Network without Batching (Strategy S_0) and with TCP Traffic

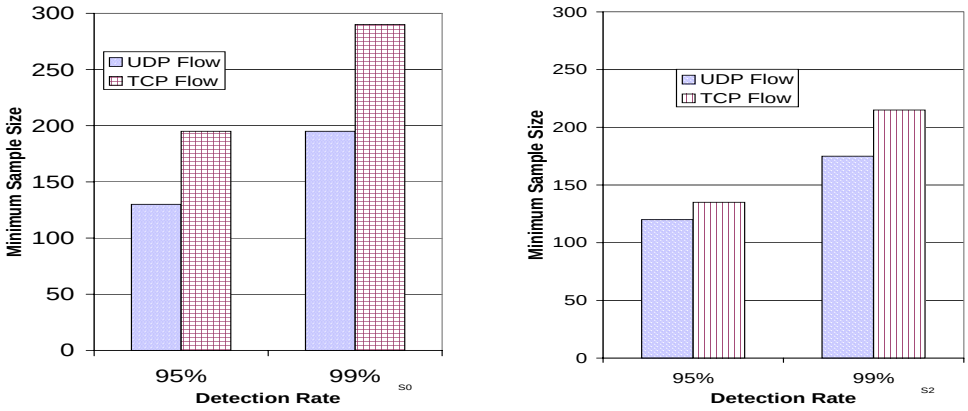


(c) Mix Network with Batching (Strategy S_2) and with UDP Traffic



(d) Mix Network with Batching (Strategy S_2) and with TCP Traffic

Fig. 17. Detection Rate (Sample size: number of sample intervals of length 10ms)



(a) Mix Networks without Batching (Strategy S_0) (b) Mix Networks with Batching (Strategy S_2)

Fig. 18. Minimum Sample Size

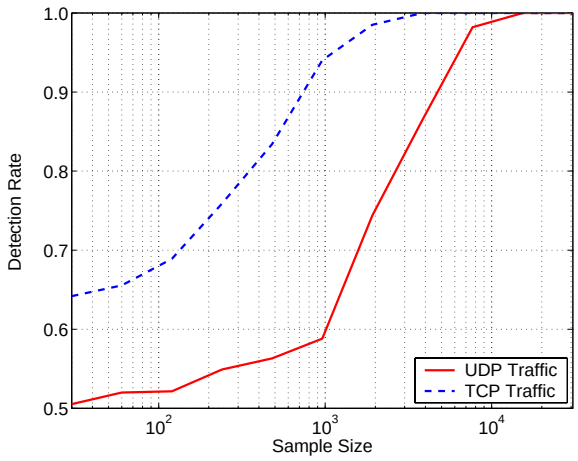
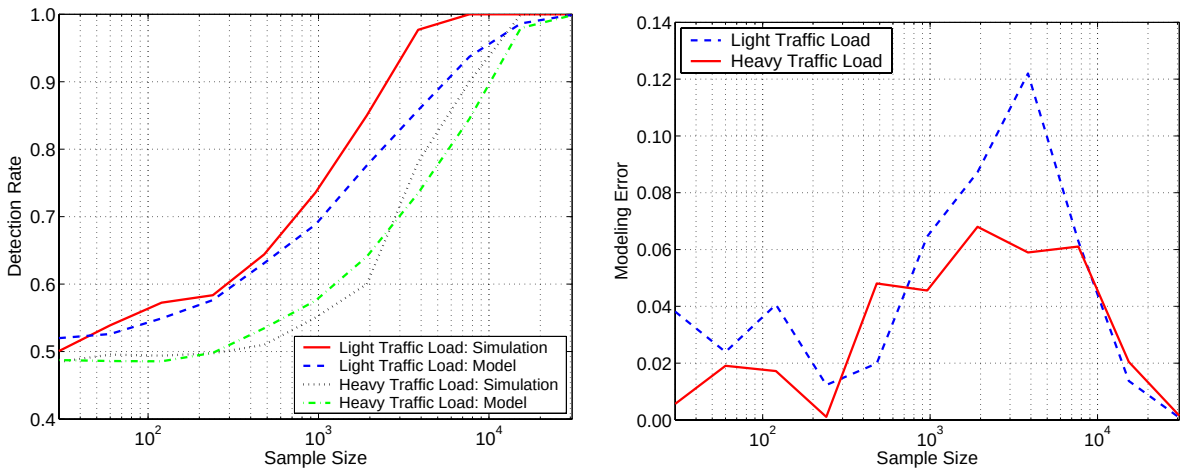


Fig. 19. Effectiveness of Flow Correlation against Continuous-time Mix



(*a*)Detection Rate by Simulation and Theory (*b*)Error of Detection Rate Estimation

Fig. 20. Modeling Error

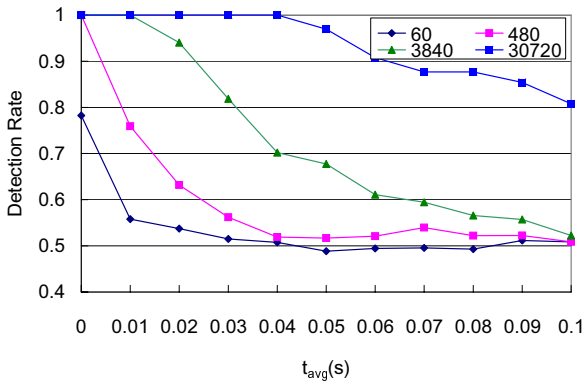


Fig. 21. Impact of Parameter t_{avg}

CHAPTER V

DATA PRE-CONDITIONING FOR TIMING ATTACKS

A. Motivation

In this chapter, we describe a class of attacks on low-latency anonymity networks which we will call *flow separation* attacks. Their aim is to *separate* (as opposed to *identify*) flows inside a network, based on aggregate traffic information only. This attack method can also be used as a data pre-conditioning method to improve the effectiveness of other timing attacks such as flow correlation attacks.

One of the main functions of the mix network is to mix the traffic flows and so render senders or receivers anonymous. Mix networks typically achieve this by perturbing the traffic in (a) the payload domain (through encryption), (b) in the route domain (through re-routing) and (c) in the timing domain (through batching and link padding). By using the flow separation attack, an attacker can separate the flows based on passively collected traffic data. Further attacks by frequency spectrum matching or time domain cross-correlation [34] can then easily determine the path of flows in the mix network if additional knowledge about flows is available.

Another motivation stems from the flow correlation attack described in Chapter III. For flow correlation attacks, we assume the attacker can obtain the packet timing information of the flow of interest. The assumption is valid since the flow-level information can be obtained by eavesdropping packets at the edge of anonymous communication network. But with the aid of flow separation attack, the assumption is not needed any more because the attacker can separate individual flows based on the information of aggregate traffic.

The flow separation attack employs the *blind source separation* model [68], which

was originally defined to solve *cocktail party problem*: The goal of blind source separation in this case is to extract one person's voice signal given a mixtures of voices. Blind source separation algorithms solve the problem based on the independence between voices from different persons. Similarly, in a mix network, we can use blind source separation algorithms to separate independent flows.

B. Threat Model

The threat model in this chapter is similar to the threat model described in Section III.C with two differences as following:

- We do not need the simplifying assumption that the traffic characteristic of the flow under consideration (the *input flow*) is known.
- We focus on mixes operating as simple proxy. No batching or reordering is used. Link padding (with dummy packets) is not used either. This follows the practice of some existing mix networks such as, Tor [23].

Given a mix with observations of aggregate traffic at input ports $I_1, \cdots, I_n$ and output ports $O_1, \cdots, O_n$, the goal of the flow separation attack is to partition the aggregate traffic into either individual flows or small aggregates that contain the individual flows.

C. Flow Separation in Mix Networks

In this section, we will first define the problem in the context of blind source separation and then describe how to apply the flow separation method in a mix network.

1. Blind Source Separation

Blind source separation is a methodology in statistical signal processing to recover unobserved "source" signals from a set of observed mixtures of the signals. The separation is called "blind" to emphasize that the source signals are not observed and that the mixture is a black box to the observer. While no knowledge is available about the mixture, in many cases it can be safely assumed that source signals are independent. In its simplest form [69], the blind source separation model assumes n independent signals $F_1(t), \cdots, F_n(t)$ and n observations of mixture $O_1(t), \cdots, O_n(t)$ where $O_i(t) = \sum_{j=1}^{n} a_{ij} F_j(t)$. The goal of blind source separation is to reconstruct the source signals $F_j(t)$ using only the observed data $O_i(t)$ and the assumption of independence among the signals $F_j(t)$. A Very nice introduction to the statistical principles behind blind source separation is given in [69].

2. Flow Separation as a Blind Source Separation Problem

As in the previous chapters, we define a *flow* as a series of packets that are exchanged between a pair of hosts. Typically, such a flow is identified by a tuple of source/destination addresses and port numbers. Similarly, we define an *aggregate flow* at the *link-level* to be the sum of the packets (belonging to different flows) on the link. We define the *mix-level* aggregate flow as the sum of packets through the same input and output port of a mix. Unless specified, otherwise the word "flow" in the remaining of this chapter means "mix-level aggregate flow" for brevity.

We will show in this chapter that, for the attacker who tries to break the anonymity of a mix, it is very helpful to *separate* the flows through the mix based on the observation of the link traffic. The separation of the flows through the mix can recover the traffic pattern of flows, which can be used in further attacks, such as

the frequency spectrum matching attack described in Section C.3 or the time domain cross-correlation attack [34].

In this chapter, we are interested in the traffic pattern carried in the time series of packet counts during each sample interval T. For example, in Figure 22, the attacker acquires a time series $O_1 = [o_1^1, o_2^1, \cdots, o_n^1]$ of packet counts by observing the link between Sender S_1 and the mix. We use n to denote the *sample size* in this chapter. The attacker's objective is to recover the packet count time series $F_i = [f_1^i, f_2^i, \cdots, f_n^i]$ *for each flow*. For the simplest case, we assume that (a) there is no congestion in the mix and that (b) the time series can be synchronized. (We will relax both assumptions in later sections.) In the example of Figure 22, the time series F_1 is contained in both time series O_1 and O_3 i.e. $O_1 = F_1 + F_2$, $O_3 = F_1 + F_3$. For a mix with j input ports, k output ports and m mix-level aggregate flows, we can rewrite the problem in vector-matrix notation,

$$\begin{pmatrix} O_1 \\ O_2 \\ \vdots \\ O_{j+k} \end{pmatrix} = \mathbf{A}_{(j+k)\times m} \begin{pmatrix} F_1 \\ F_2 \\ \vdots \\ F_m \end{pmatrix} \tag{5.1}$$

where $\mathbf{A_{(j+k)\times m}}$ is called *mixing matrix* in the blind source separation problem [68].

The flow separation can be solved using a number of blind source separation techniques. The rationale for blind source separation relies on the fact that the aggregate flows through a mix are independent from each other, since the aggregate flows are from different sources. Even the flows from a same host, such as F_1 and F_2, can be regarded as independent as they follow different paths and are controlled by different sockets. This independence assumption is of course only valid as long as Sender S_1 is not heavily overloaded, since otherwise one flow would influence the other. Given

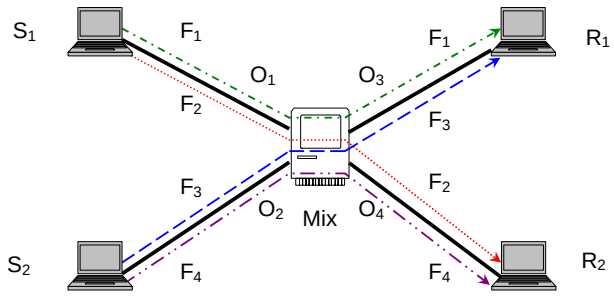


Fig. 22. An Example for Flow Model

the observations $O_1, O_2, \cdots, O_{j+k}$, blind source separation techniques estimate the independent aggregate flows $F_1, F_2, \cdots, F_m$ by maximizing the independence between estimated aggregate flows. The common methods employed in blind source separation are minimization of mutual information [70, 71], maximization of nongaussianity [72, 73] and maximization of likelihood [74, 75]. In the following, we need to keep in mind that flow separation often is not able to separate individual flows. Rather, mix-level aggregates flows that share the links at the observation points form the minimum separable unit.

a. Practical Blind Source Separation in Mix Networks

Basic blind source separation algorithms require the number of observations to be larger than or equal to the number of independent components. For flow separation, this means that $j + k \geq m$, where j and k denote the number of observations at the input and output of the mix, respectively, and m denotes the number of flows. Advanced blind source separation algorithm [76, 77] target over-complete bases problems and can be use for the case where $m > j + k$. But they usually require m, the number of independent flows, to be known. Since all the mix traffic is encrypted and padded, it is hard for the attacker to estimate m. In this chapter, we assume that $m = j + k$. The cost of the assumption is that some independent flows can not be

separated, that is, they remain mixed after the separation step. We will see that this is not a severe constraint, in particular not in mix networks where flows that remain mixed in some separations can be separated using separation results from neighbor mixes.

Unless there is multicast or broadcast traffic through the mix, the $j + k$ observations will have some redundancy, because the summation of all the observations on the input ports are equal to the summation of all the observations on the output ports. In other words, the row vectors of the mixing matrix are linearly dependent. Again, the cost of the redundancy is that some independent flows are not separated.

The flow estimation generated by blind source separation algorithms is usually a lifted, scaled version of the actual flow (of its time series, actually). Sometimes, the estimated flow may be of different sign than the actual flow. Both lifting and scaling does not affect the frequency components of the time series, and so frequency matching can be used to further analyze the generated data.

Furthermore, since the elements of the estimated mixing matrix are not binary, it is not straightforward to tell the direction of each aggregate flow. Some heuristic approach can be used, but we leave this to further research.

In the rest of this chapter, we will show that the issues identified above can be largely solved with the use of appropriate frequency matching.

3. Frequency Matching Attack

After the flows have been separated, a number of flows, each with a given packet-count time series, have been determined to traverse the mix.

Frequency spectrum matching has shown to be particularly effective to further analyze the traffic. The rationale for the use of frequency matching is four-fold: First, the dynamics of a flow, especially a TCP flow [78], is characterized by its periodicities.

By matching the frequency spectrum of a known flow with the frequency spectrums of estimated flows obtained by blind source separation techniques, we can identify the known flow with high accuracy. Second, frequency matching can easily remove the ambiguities introduced by the lifting and scaling in the estimated time series by removing the zero-frequency component. Third, frequency spectrum matching can also be applied on the mix-level aggregate flows, since the different frequency components in each individual flows can characterize the aggregate flow. Fourth, the low frequency components of traffic are often not affected by congestion as they traverse multiple switches and mixes. This is particularly the case for TCP traffic, where the frequency components are largely defined by the behavior at the end hosts. In summary, frequency spectrum analysis has excellent prerequisites to be highly effective.

Even if no information is available about individual flows, the attacker can easily determine if there is communication between two neighboring mixes. Matching the estimated aggregate flows through the neighboring mixes can give attackers more information, such as how many aggregate flows are going through the next mix. In a mix network, an aggregate flow through a mix may split into aggregate flows of smaller size, multiplex with other aggregate flows, or do both. By matching the estimated aggregate flows through neighboring mixes, the attacker can detect the split and multiplex. Based on the information gathered, the attacker can eventually get a detailed map of traffic in a mix network. In Section F, we show a traffic map obtained from the aggregate flow matching.

The sample interval T (see page 66) is important to the frequency spectrum matching. The averaging effect of the sampling over an interval T on the frequency spectrum matching results can be modeled as low-pass filtering. If we are matching TCP flows, it is important to select a proper sample interval to avoid filtering out

interesting TCP frequency components such as round trip time (RTT) and time-out frequencies. More details on selecting T and modeling of the effect of T can be found in Section III.F.

In the following, we will be using frequency matching of the *separated* flows against the *actual* flows in the network to measure the accuracy of the flow separation. The rationale for this method is that a highly accurate flow separation will result in good matching with the component flows, whereas a poor separation will generate separated flows that can not be matched with the actual ones.

D. Evaluation on Single Mix with Different Combinations of Traffic

In this section, we will evaluate the performance of the flow separation for a single mix. We use the blind source separation algorithm proposed in [79] to separate the flows. The accuracy of separation will be measured using frequency matching with actual flows.

1. Metrics

In the following, we will adopt two metrics to evaluate the accuracy of the flow separation. Both metrics are based on a comparison of the separated flows with the actual flows in the mix.

As first performance metric, we use *mean square error (MSE)*, a widely used performance criterion in blind source separation research. Let $F_A = [f_1^A, f_2^A, \cdots, f_n^A]$ represent the time series of the actual flow and $F_B = [f_1^B, f_2^B, \cdots, f_n^B]$ represent the time series estimated by the blind source separation algorithm. To match the time series F_A with F_B, we first need to scale and lift F_B so that they have the same mean

and variance.

$$F'_B = \frac{std(F_A)}{std(F_B)} \cdot (F_B - mean(F_B) \cdot [1, 1, \cdots, 1]) + mean(F_A) \cdot [1, 1, \cdots, 1] \quad , \quad (5.2)$$

where $std(F)$ and $mean(F)$ denote the standard deviation and average of the time series F, respectively. The *mean square error* is defined as follows:

$$\varepsilon_{A,B} = \frac{\|F_A - F'_B\|^2}{n} \quad . \quad (5.3)$$

Since the times series F_B can also be a flipped version of F_A, we also need to match F_A with $-F_B$.

As the second metric, we use what we call *frequency spectrum matching rate*. We define the matching rate to be probability that the separated flow F_B has the highest frequency spectrum cross-correlation with the actual flow F_A.

We note that while the mean square error captures the accuracy of the separation in the time domain, the matching rate captures the effectiveness of the separation in the frequency domain.

2. Experiment Setup

Figure 23 shows the experimental network setup for single mix. We use ns-2 to simulate the network. The links in the figure are all of $10Mbit/s$ bandwidth and $10ms$ delay[1] if not specifically mentioned. In the series of experiments in this section, the mix under study has two input ports and two output ports and four aggregate flows passing through the mix, as shown in Figure 22. We will study mixes with more than two ports in Section E. Unless specified otherwise, we will use time observation intervals of 32 second length and sample interval of $10ms$ length, resulting in time

[1]Senders and receivers can be at a large distance from the mix, potentially connecting through several routers and switches.

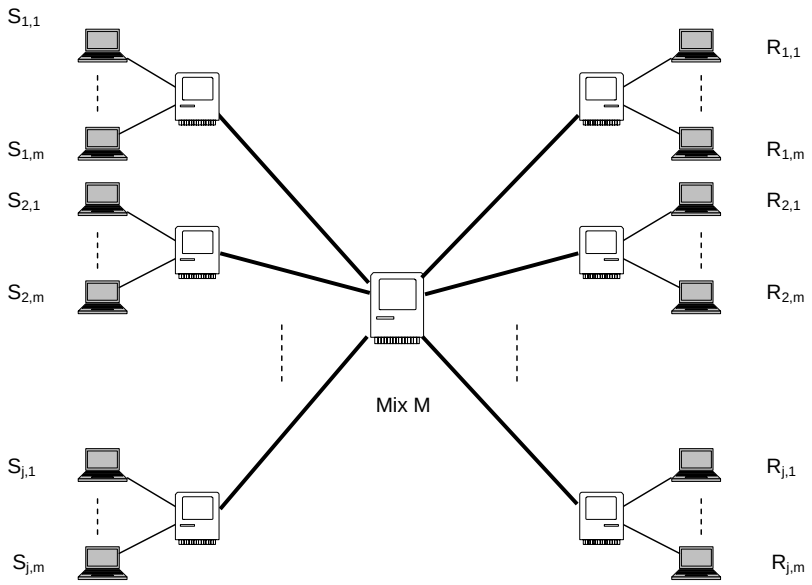


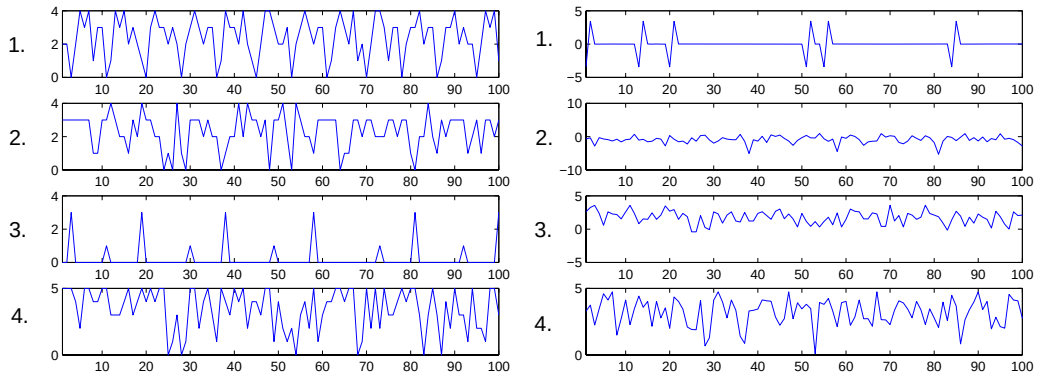
Fig. 23. Experiment Setup for Single Mix

series of size $n = 3200$. Similar results were obtained for shorter observations as well.

3. Different Types of Traffic

In this experiment, four aggregate flows, including one FTP flow, one sequence of HTTP requests, and two on/off UDP flows, are passing through the mix. The parameters for the flows are as follows: Flow 1: FTP flow, with round trip time around $80ms$. Flow 2: UDP-1 flow, on/off traffic, with burst rate $2500kbit/s$, average burst time $13ms$ and average idle time $6ms$. Flow 3: HTTP flows, with average page size 2048 byte. Flow 4: UDP-2, on/off traffic with burst rate $4000kbit/s$, average burst time $12ms$ and average idle time $5ms$. All the random parameters for the flows are exponentially distributed. The flows are passing through the mix as shown in Figure 22.

Figure 24 shows portions of the actual times series (Figure 24(a)) and of the estimated time series (Figure 24(b)). From the figures, it is apparent that the flipped

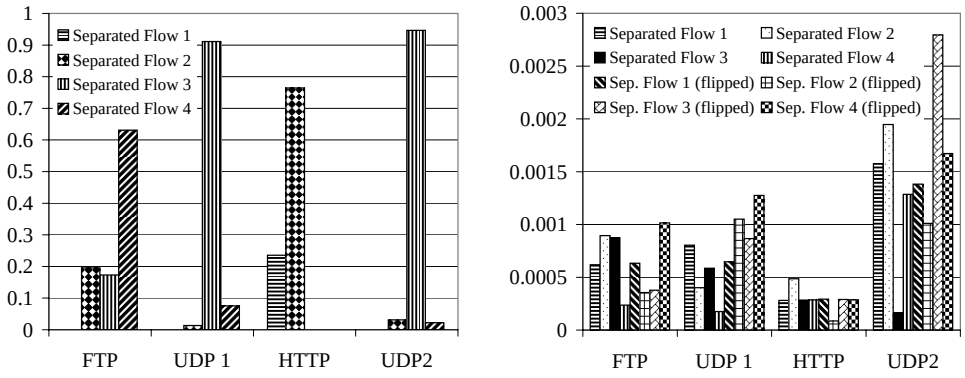


(a) Actual Flows **(b) Separated Flows**

Fig. 24. Example of Flow Separation for Different Types of Traffic

version of the actual flow 3 (HTTP flows) is contained in the estimated flow 2. We also observe the resemblance between actual flow 1 (FTP flow) and estimated flow 4. Estimated flow 1 is clearly not close to any actual flows. This is caused by the redundancy contained in the observations, as described in Section C.2.

Figure 25 shows the separation accuracy using the two metrics defined earlier. We note in Figure 25(b) that both the separated flow and its flipped time series is compared against the actual flows. Both metrics can identify the FTP flow, HTTP flows and one UDP flow. But the two metrics disagree on the other UDP flow. This is because of the redundancy in the observations, and the two UDP flows can not be separated. MSE fails for this case since it is designed for one-to-one flow matching while frequency spectrum matching is more suitable for matching of flows against aggregates. The latter case is more common in the context of flow separation.



(a) Frequency Spectrum Matching Rate **(b) MSE**

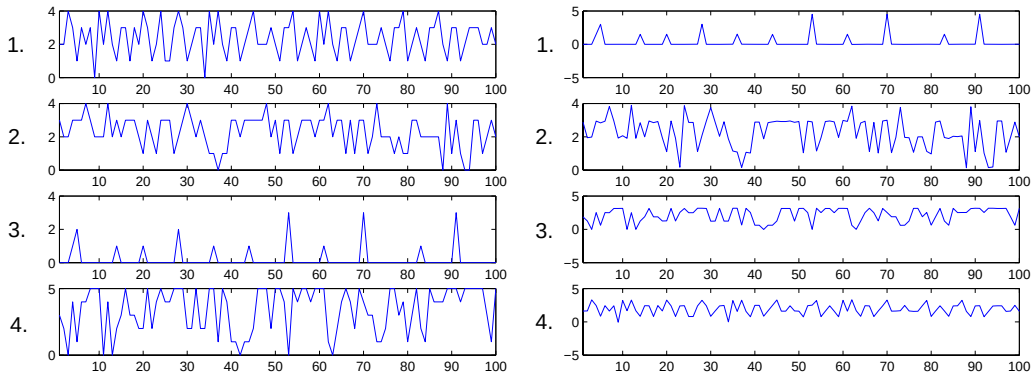
Fig. 25. Performance of Flow Separation for Different Types of Traffic

4. Different Types of Traffic with Multicast Flow

In this experiment, the flow UDP-1 in the previous experiment is multicast to both output ports.

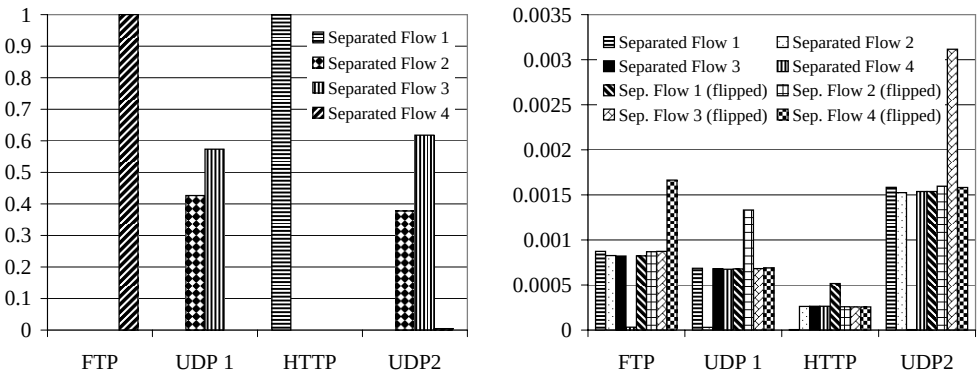
Portions of the actual flows and the estimated flows are shown in Figure 26. We observe the correspondence between the actual flows and estimated flows easily. In comparison with the previous experiment, we can conclude that multicast flows can help the flow separation. The reason is that in this experiment, there is no redundant observation when the multicast flow is passing through the mix.

MSE performance metrics in Figure 27 identify the flows successfully. Frequency spectrum matching successfully determine the FTP and HTTP flows, but does not perform well on the UDP flows. This is because the two UDP flows have approximately same period and the periodical behavior is not strong for exponential on/off traffic.



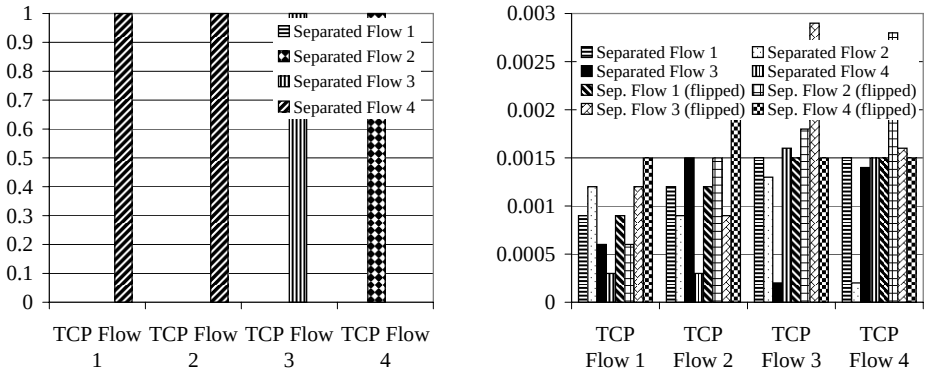
(a) Actual Flows **(b) Separated Flows**

Fig. 26. Example of Flow Separation for Different Types of Traffic (with Multicast Traffic)



(a) Frequency Spectrum Matching Rate **(b) MSE**

Fig. 27. Performance of Flow Separation for Different Types of Traffic (with Multicast Traffic)



(a) Frequency Spectrum Matching Rate **(b) MSE**

Fig. 28. Performance of Flow Separation for TCP-Only Traffic (without Multicast Traffic)

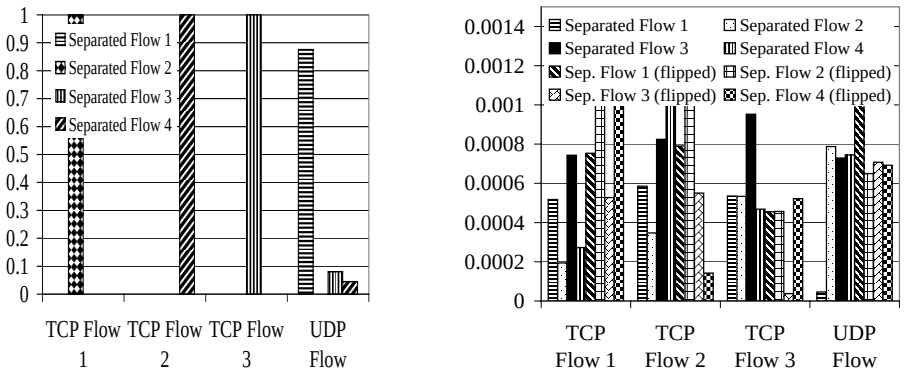
5. TCP-Only Traffic

Since most of the traffic in today's network is TCP traffic, we focus on TCP traffic in the next series of experiments. All the flows in this experiment are FTP flows. To distinguish the flows, we vary the link delays between the sender and mix, with S_1 having $10ms$ link delay to the mix, and S_2 having $15ms$ delay.

Figure 28 shows the flow separation performance. Since there is no multicast traffic, the redundancy in observations results that TCP Flow 1 and TCP Flow 2 are still mixed. But the flows are identified successfully, especially by the frequency spectrum matching method.

6. TCP-Only Traffic with Multicast Flow

In this experiment, we change one FTP flow in the previous experiment to a multicast UDP flow. The UDP flow is exponential on/off traffic with the same parameter as UDP-1 in the experiment of Section 3.



(a) Frequency Spectrum Matching Rate **(b) MSE**

Fig. 29. Performance of Flow Separation for TCP-Only Traffic (with Multicast Traffic)

Figure 29 shows the flow separation performance. Similarly to the effect of multicast flow on different types of traffic, the four flows are separated completely since there are no redundant observations. We can also observe that the frequency spectrum method identifies the FTP flows successfully. But the performance on the exponential on/off UDP flow is not as good as FTP flows because exponential traffic flow's frequency signature is very weak.

E. Evaluation of Scalability of Flow Separation

In this section, we focus on the scalability of flow separation. We evaluate the flow separation performance with respect to (a) increasing the number of flows in mix-level aggregate flows (the number of aggregate flows remains constant), (b) increasing the number of mix-level aggregate flows, and (c) increasing the number of ports per mix. We will show that flow separation remains effective when systems grow in terms of nodes, flows, and amount of traffic.

1. Experiment Setup and Metrics

In this series of experiment, the network setup is as shown in Figure 23: The setup consists of $m \times j$ sender hosts and $m \times j$ receivers hosts. The flows from the senders get routed through j sender-side routers (mixes) to the Mix M, which forwards the traffic to the receiver-side routers (mixes) and finally to the receivers nodes. We note that there is no need to investigate mixes with unequal number of input and output ports(i.e. $j \neq k$, since blind source separation does not distinguish input from output anyway. All the flows are FTP flows. To distinguish different FTP flows, we add $5ms$ delay incrementally to the link connected to each sender.

In this series of experiment, we will limit ourselves to frequency spectrum matching results as the performance metrics, mainly because they are more suitable for redundant observations when there is no multicast traffic. When there are more than one flows in one mix-level aggregate flow, we will use the frequency spectrum of the actual mix-level aggregate flow to match with the separated mix-level aggregate flows. We will show only the maximum matching rate for each actual flow.

2. Scalability: Size of Aggregate Flows

We first leave the number of observations constant and increase the number of FTP flows in each mix-level aggregate flow by adjusting m. The mix under study still has two input ports and two output ports. The directions of the aggregate flows are still as show in Figure 22. But now each aggregate will contain $\frac{m}{2}$ flows.

Figure 30 shows the performance of flow separation for different aggregate sizes. We use $< m, B, T >$ to represent different experiments, where B denotes the bandwidth of each link in the experiment, and T denotes the sample interval.

First, we observe that for the experiments with link bandwidth $10Mbit/s$ and

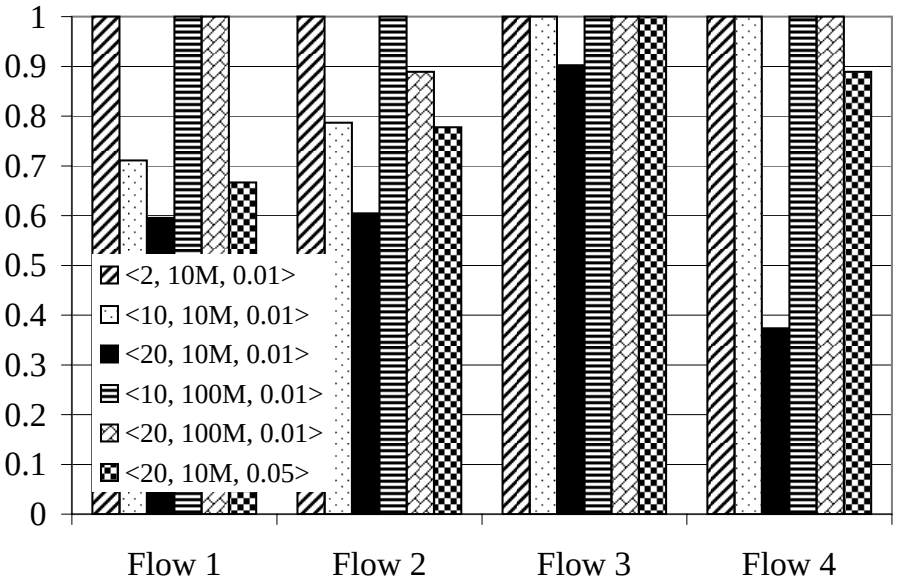


Fig. 30. Frequency Spectrum Matching Rate for Different Size of Aggregate Flows

sample interval $10ms$, the performance decreases when the aggregate size increases. This is because the TCP flows tends to "fill the link bandwidth" when there is enough data to send. When the number of FTP flows increases, congestion happens, and the time series of individual flows get perturbed as they traverse the mix. In addition, packets get dropped, which perturbs the time series even more. The perturbation caused by congestion and by the packet drops degrades the performance.

Second, we observe that if we increase the bandwidth from $10Mbit/s$ to $100Mbit/s$, the performance significantly increases for the same aggregate size and sample interval. Obviously less congestion causes better performance.

Third, for the same aggregate size and link bandwidth, increasing sample intervals increases the performance. Larger sampling intervals reduce the boundary effects caused by non-perfect sampling and by lack of synchronization, and it reduces the effect of congestion as well. As a result, signal noise ratio increases and flow separation performance increases too.

3. Scalability: Number of Aggregate Flows and Number of Flows

In this set of experiments, we change the number of mix ports and the number of flows through the mix. Because of the space limitation, we show three typical experiments here.

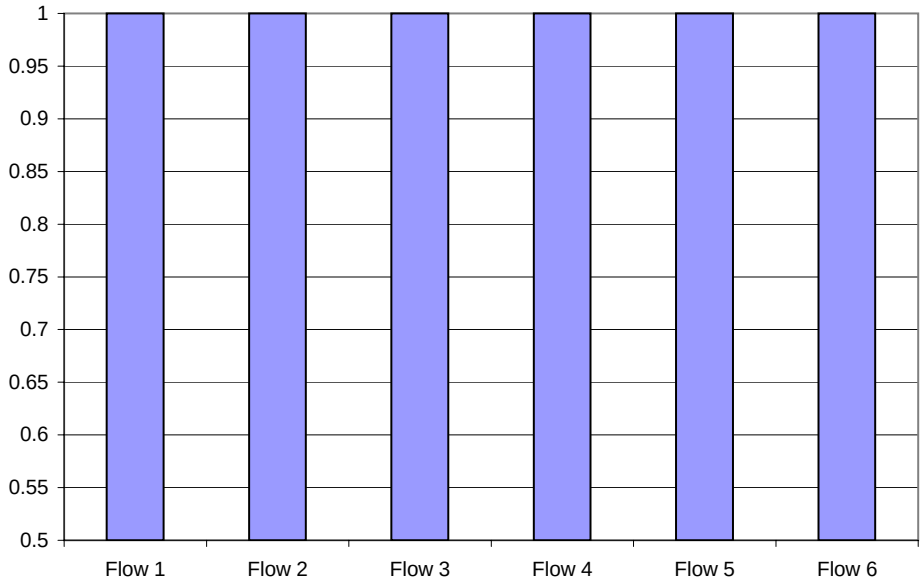
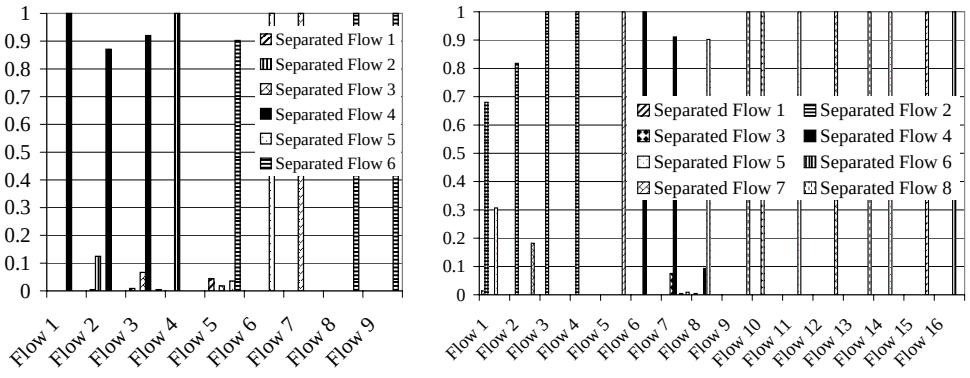


Fig. 31. Frequency Spectrum Matching Rate (3×3 mix, 6 Flows)

Figure 31 shows the performance of experiments on six aggregate flows through a 3×3 mix. We can observe that performance of flow separation remains good when the ports increase as long as the ratio of the number aggregate flows over number of observations (ports) remains constant.

Figure 32(a) and 32(b) show the performance of experiments with nine mix-level aggregate flows through a 3×3 mix and sixteen mix-level aggregate flows through a 4×4 mix, respectively. Our experiments show that when the number of aggregate flow increases, there are more flows that can not be separated. In other words, the number of flows that remain mixed together increases. This is because the ratio of aggregate flows to observations increases, and the blind source separation algorithm needs more



(a) 3 × 3 mix, 9 Flows **(b) 4 × 4 mix, 16 Flows**

Fig. 32. Frequency Spectrum Matching Rate

observations. On the other hand, the matching rate by frequency spectrum matching remains high. This indicates that the flows have been correctly separated.

F. Evaluation for Mix Networks

Flow separation can also be used by a global passive attacker. The attacker can do flow separation at each mix according to observations obtained at that mix. Then the attacker can correlate the separated aggregate flows to derive the traffic map of the whole mix network.

1. Experiment Setup

Figure 33 shows the network setup in this experiment. Eight FTP flows from senders on the left side are traversing the mix network. To distinguish these eight FTP flows, we incrementally add $5ms$ delay to link connected to each sender. To simulate the cross traffic in the mix network, four larger aggregates of flows are added to the mix network. According to the self-similar nature of the network traffic [80], the high-

Table II. Flow Configuration

Flows	Path	Parameters	Throughput (packets/s)
1	$S_1 \rightarrow M_1' \rightarrow M_1 \rightarrow M_3 \rightarrow M_5 \rightarrow M_7 \rightarrow M_9 \rightarrow M_{11} \rightarrow M_5' \rightarrow R_1$	FTP	106.125
2	$S_2 \rightarrow M_1' \rightarrow M_1 \rightarrow M_4 \rightarrow M_5 \rightarrow M_8 \rightarrow M_9 \rightarrow M_{12} \rightarrow M_7' \rightarrow R_5$	FTP	100.791
3	$S_3 \rightarrow M_2' \rightarrow M_1 \rightarrow M_3 \rightarrow M_5 \rightarrow M_7 \rightarrow M_9 \rightarrow M_{11} \rightarrow M_6' \rightarrow R_3$	FTP	95.936
4	$S_4 \rightarrow M_2' \rightarrow M_1 \rightarrow M_4 \rightarrow M_5 \rightarrow M_8 \rightarrow M_9 \rightarrow M_{12} \rightarrow M_8' \rightarrow R_7$	FTP	91.541
5	$S_5 \rightarrow M_3' \rightarrow M_2 \rightarrow M_3 \rightarrow M_6 \rightarrow M_7 \rightarrow M_{10} \rightarrow M_{11} \rightarrow M_5' \rightarrow R_2$	FTP	87.531
6	$S_6 \rightarrow M_3' \rightarrow M_2 \rightarrow M_4 \rightarrow M_6 \rightarrow M_8 \rightarrow M_{10} \rightarrow M_{12} \rightarrow M_7' \rightarrow R_6$	FTP	83.858
7	$S_7 \rightarrow M_4' \rightarrow M_2 \rightarrow M_3 \rightarrow M_6 \rightarrow M_7 \rightarrow M_{10} \rightarrow M_{11} \rightarrow M_6' \rightarrow R_4$	FTP	80.483
8	$S_8 \rightarrow M_4' \rightarrow M_2 \rightarrow M_4 \rightarrow M_6 \rightarrow M_8 \rightarrow M_{10} \rightarrow M_{12} \rightarrow M_8' \rightarrow R_8$	FTP	77.357
9	$\rightarrow M_3 \rightarrow M_5 \rightarrow M_8 \rightarrow M_{10} \rightarrow$	Pareto	319.317
10	$\rightarrow M_3 \rightarrow M_6 \rightarrow M_8 \rightarrow M_9 \rightarrow$	Pareto	318.558
11	$\rightarrow M_4 \rightarrow M_5 \rightarrow M_7 \rightarrow M_{10} \rightarrow$	Pareto	321.806
12	$\rightarrow M_4 \rightarrow M_6 \rightarrow M_7 \rightarrow M_9 \rightarrow$	Pareto	323.36

volume cross traffic is Pareto distributed. The configuration of the flows is shown in Table II.

In the center of the mix network, the traffic volume ratio between link-level aggregate traffic and each individual flow from senders is at least 7 : 1. We assume the attacker can observe links connected to Mix $M_1, M_2, \cdots, M_{12}$. Thus, a flow originating from S_1 can take 2^6 possible paths.

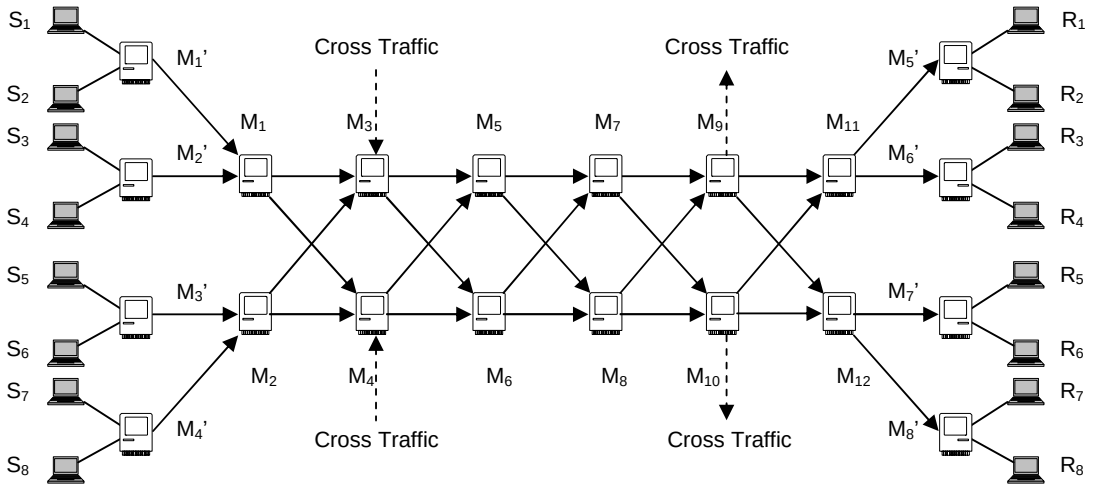


Fig. 33. Experiment Setup of Mix Network

2. Performance Metrics

To evaluate the performance of detecting a flow in the network, we introduce a network-level performance metrics, which is based on the entropy-based anonymity degree proposed in [4, 5]. Suppose we are interested in flow F_x. The attacker can suspect the flow F_x taking a path P_i with probability p_i based on the information gathered from the anonymity attack on the mix network. Assuming there are h possible paths that can be suspected as the path taken by the flow F_x, we define the anonymity degree as

$$D = -\sum_{i=1}^{h} p_i \log_2 p_i \quad . \tag{5.4}$$

Suppose a flow originated from S_1 in Figure 33 is suspected to use each of 2^6 possible paths with equal probability. Then the anonymity degree $D = 6bit$.

3. Performance

Figure 34 shows the mean value of cross correlation using frequency spectrum matching method among the first four FTP flows and separated flows recovered from Mix $1 - 12$. The cross-correlation values less than 0.1 are marked as white. Please note that the cross-correlation values between separated flows recovered from the same mix are also marked as white. This includes the cross-correlation (auto-correlation) for the same separated flow or FTP flow.

From the cross-correlation map shown in Figure 34, we can easily figure out the traffic direction in the mix network.

Figure 35 shows an algorithm to detect a flow say F_x in the network based on flow separation attack and frequency spectrum matching method. The main idea behind the algorithm is to first use the aggregate flow F_{tmp}, which is determined to be on the path previously to match the separated flows on the neighboring mixes. The threshold

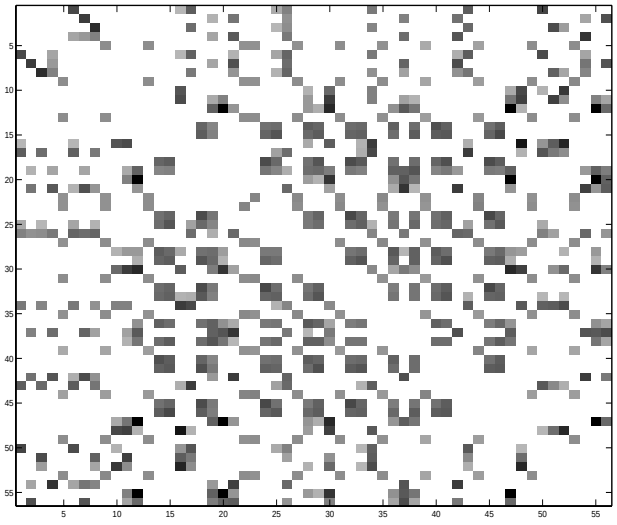


Fig. 34. Mean Value of Cross Correlation between Four FTP Flow and Estimated Flows

threshold_1 is used to determine the Candidate array which includes the separated flows that have some components of the identified aggregate flow F_{tmp}. Then we match the flow F_x with the separated flows in the Candidate array to determine the most closely matching flow on the next hop. The process continues until the correlation is too weak, which is determined by the threshold *threshold_2*. Thresholds *threshold_1* and *threshold_2* can be determined by online learning based either on data collected by attacker or on some heuristics setting. The algorithm works in dynamic programming way. It can be further improved by considering more possible routes and select the one has the largest overall possibilities.

Figure 36 shows the comparison of the anonymity degree before and after the attack. Due to the effectiveness of the attack, the anonymity degree reduces significantly. We use the algorithm described in Appendix A to detect the path of a flow.

```
F_tmp=F_x
M_tmp=M_x
while (mix M_tmp is not a dead-end) do {
        empty Candidate array
        for each mix M_i connected to M_tmp {
                for each flow F'_y separated by flow separation attack on M_i {
                        matching(F_tmp, F'_y)=Cross-correlation coefficient of the frequency
                        spectrums of F_tmp and F'_y
                        if  matching(F_tmp, F'_y)> threshold_1
                                record (F'_y, M_i) into array Candidate
                }
        }
        find the element (F'_max, M_max) in candidate array,  so that
        matching(F_x, F'_max) ≥matching(F_x, F'_y), for any F'_y in Candidate array
        if  matching(F_x, F'_max) <threshold_2
                break
        F_tmp=F'_max
        M_tmp=M_max
        record M_max as a mix on the flow path
}
```

Fig. 35. Flow Detection Algorithm

We set the Thresholds *threshold_1* to zero and *threshold_2* to 0.1 heuristically. The result is based on the observations of 32 seconds of traffic. Our data indicates that similar results can be obtained with significantly smaller observation intervals.

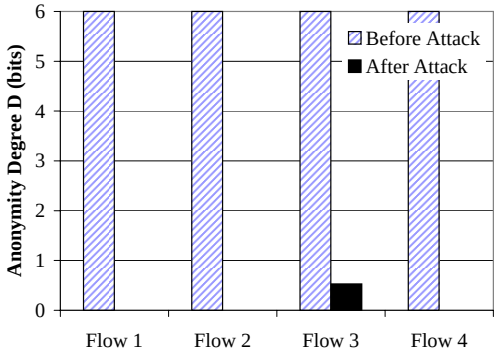


Fig. 36. Anonymity Degree

G. Discussion

The countermeasures to flow separation attack are intuitive.

- Padding the links so that the observations obtained by the passive attacker are identical, or at least mostly redundant.

- Use pool-mix like batching strategies. Pool mixes fire packets with a certain probability p. If the probability p is small enough, the aggregate flows at the output ports can be significantly different from aggregate flows at the input ports. Adding noise in the passive attacker's observations can degrade the performance of flow separation attacks. But the cost will be increased packet transfer latency and lower throughput, especially for TCP traffic.

- Increase the dependency among flows by adding dependent dummy traffic flows to the mix-level aggregate flows.

- Padding each aggregate flow so that the distribution of the packet count is Gaussian. Most blind source separation algorithms fail when the signals mixed are Gaussian distributed. But different classes of blind source separation algorithm that make use of the time structure of the signals can still separate the flows e.g., [81, 82].

In general, it can be said that blind source separation algorithms coping with noisy delayed signals, over-complete base problems are still active research topics in blind source separation research. Flow separation attacks will be more powerful when more advanced algorithms become available.

H. Summary

We proposed a new anonymity attack, called the *flow separation attack* which can be used either alone or in conjunctions with other attacks to significantly reduce the effectiveness of anonymous communication systems. Flow separation attack is based

on the blind source separation algorithms widely used to recover individual signals from mixtures of signals. Our experiments show that the anonymity attack is effective and scalable. With the aid of further attack such as frequency spectrum matching attack, flow separation attack can be used to detect the path taken by a flow in a mix network. Flow separation attack can also be used to simply recover the traffic map of the anonymity network. We discuss the possible usage of flow separation attack in different anonymity network settings, and we elaborate on criteria for its countermeasures.

In the following, we will describe the application of blind source separation in wireless networks.

CHAPTER VI

WIRELESS CONFIDENTIALITY

A. Introduction

With the increasing popularity of 802.11 style wireless networks (WLANs), both in infrastructure and in ad-hoc mode, *location privacy* issues in such networks and in ubiquitous computing environments in general have received great attention. Significant recent work has focused on the identification of location privacy risks associated with the use of WLANs and on the implication caused by the weak location privacy. Location information can be gathered or inferred in a variety of ways, such as through direct identification of nodes at MAC layer or above [26, 29], through physical signal shape and propagation analysis (e.g., [42, 83]), or through tracking of interactions with services and access points (e.g., [84]).

A number of schemes have been proposed to preserve location privacy in such systems. Some schemes hide the node identity through appropriate encryption and the use of one-time MAC addresses [29] or broadcast-only communication [26, 27, 28]. Others attempt to counter signal-level location through manipulation of signal strength [48]. Service access tracking is addressed by schemes such as path perturbation [85], in which nodes report appropriately modified locations whenever they are close to other nodes with the goal to confuse the location tracker.

In this chapter we want to bring attention to the possibility of an attacker compromising -with the help of a network of very simple sensors- the location privacy in a *perfectly anonymized* wireless network. By the sensors being "simple" we mean that they monitor packets at MAC level or above, have no directional capabilities, cannot distinguish packets, cannot relate network packets to senders or receivers,

have only coarse time synchronization support, and have only low-bandwidth links for inter-sensor communication. (While this is not a real limitation for wireless receivers, we don't need support for signal-strength measurement on the sensors either.) Such networks of sensors could be realized either by a number of WLAN users that collude and exchange information or by a separate infrastructure of sensor nodes. Given their limited capabilities, we use the sensors to count packets over intervals of given length, and to forward the resulting time series of packet counts for analysis to a base station. No information is available about how many nodes are present and sending in the area, and the anonymity measures in the WLAN prevent the sensors from distinguishing packets sent from different nodes.

We will describe two statistical signal analysis methods to first estimate the number of nodes in areas of the network (we call this *node density*) and second separate the overall traffic into estimates of actual traffic sent by nodes in the network to pinpoint the location of sending nodes (*node location*). For the node-density estimation we use *Principal Component Analysis* (PCA). PCA is a classical statistical method used to reduce the dimensionality in a dataset. It can represent a dataset of correlated variables with less uncorrelated variables, which are called principal components. For the traffic separation we use the *Blind Source Separation* (BSS) method [68]. BSS was originally developed to solve the *cocktail party problem*, where the goal is to extract one person's voice signal given a mixtures of voices at a cocktail party. BSS algorithms solve the problem by taking advantage of the independence between voices from different persons. Similarly, in wireless networks, we can use BSS algorithms to separate traffic from different wireless nodes. The separated traffic is not in a form that can be associated to any sender node. However, we take advantage of spatial diversity in the collected data in the sensor network to reconstruct the sender location based on the separated traffic.

Our experimental evaluations using a widely accepted packet-level network simulator (ns-2) indicate that the proposed algorithms estimate the node density with high accuracy and that they estimate node locations with both high accuracy and high confidence. The majority of experiments is performed with the intent to simulate naturally occurring (i.e. TCP) traffic. In order to show the effectiveness of the approach, we also simulate a network that uses constant-rate padding of traffic on all nodes: In such a scheme, all nodes send traffic at a constant rate using UDP, independently if they have traffic to send or not. If no traffic is ready, a dummy packet is transmitted instead. Our experiments indicate that traffic padding is largely useless in this setting: it has no impact on the effectiveness of both our node-density and node-location estimators.

We consider these results significant, since they indicate that it is impossible to maintain location privacy in 802.11-style networks against colluding WLAN users or networks of sensors that use simple off-the-shelf technology. Often anonymity measures rely on users being able to "hide in a crowd". Our experiments show that crowds are unable to hide individuals in WLAN settings. BSS algorithms can easily and effectively separate packets from different senders, based on packet-count time series only.

As in many other settings, the traffic analysis mechanisms presented in this chapter can be utilized for intrusion detection. An accurate node density estimation can be used to identify intruding actively sending nodes, independently of how well they are able to masquerade bona fide nodes. Node location estimators can then be used to identify the intruder.

B. Confidentiality Issues in Wireless Networks

Whether to support anonymity and privacy of participants in a network, or to support integrity of the network itself, a variety of information about the network must not be divulged to third parties. In the following we list a number of criteria that both the participants and the wireless network operator may want to keep confidential, in addition to the traditional anonymity criteria discussed in Chapter I:

- Node Identity: The identities of the wireless nodes in a network may need to remain confidential. One problem in such networks is that each node is identified by its Medium-access-control (MAC) address, which is sent with every packet. Wireless anonymity systems such as ANODR [26] use a broadcast address as the MAC address and a random route pseudonym for routing. Similarly, Gruteser and Grunwald [29] propose disposable MAC addresses, where MAC addresses of nodes are changed frequently. If an observer has access to location information, it may make use of it to piece together the profile of a node that uses MAX address recycling. Hoh and Gruteser propose path perturbation [85] to address this.

- Node Location: The network should also prevent the disclosure of the location of wireless nodes. Alternatively, node location information allows for node tracking and node identification, with detrimental effect on all anonymity measures.

- Node density, node number: The number of nodes inside an area should be kept confidential. A good estimation of the number of nodes in the system generally simplifies the separation of flows, and so affects all other anonymizing and confidentiality measures.

- Node Motion: Similarly to node location, information about the movement of wireless nodes should be kept confidential.

In this chapter we describe a methodology to estimate the *node density* and *node locations* in a 802.11 WLAN. In the discussion of our results we will then describe how simple extensions of the proposed scheme can be used to affect the other confidentiality criteria that we discussed in Chapter I.

C. Network Model and Threat Model

In the following we formulate the node density and node location estimation problem using a confidentiality threat paradigm: A network of wireless nodes is passively attacked by a sensor-network based eavesdropper. Node density and location estimation is highly relevant in other settings as well: For example low-cost intrusion detection schemes for ad-hoc networks can perform node density estimations at deployment time and determine if active intruders are present. If so, location estimations can support the localization of the intruder.

1. Network Model

We assume a set of wireless nodes (simply called *nodes* in the following) that communicate over an ad-hoc WLAN using an 802.11-style MAC protocol. We assume that all communications are *perfectly anonymized*: For example, all communications are either broadcast-based so that the anonymity attacker cannot identify the source and destination of a MAC frame [26, 27, 28]. Similarly MAC addresses can be recycled [29] to achieve the same effect. We also assume that all the packets inside the wireless network are encrypted, and only the receiver has the capability to successfully decrypt the packets. Both ANODR [26] and SDDR [27] use onion-based encryption

and routing to provide end-user anonymity and communication path anonymity. The anonymous data transmission in ASR [28] is based on shared secrets between nodes. So we assume the anonymity attacker can not utilize any information of packet content. As a result, no information is divulged to external observers either through packet data or header content.

In addition, we assume that nodes are able to manipulate signal power e.g. [48] so as to render any observed signal strength information difficult to use.

2. Threat Model

Similar to threat models of previously described attacks to anonymous wireless systems (e.g. [3]), our threat model assumes that the communication between nodes is observed by a network of low-cost sensors scattered around a field. The sensors can be either WLAN receivers of a set of colluding users in the area or a separate sensor network infrastructure. The attacker collects packet timing information from the sensors in the field for analysis.

We will demonstrate in this chapter that accurate node density and location estimation can be performed by a low-cost sensor network and appropriate computing capabilities at the back-end. We therefore assume that the passive attacker has the following capabilities:

- Sensor nodes are equipped with off-the-shelf 802.11 receivers.
- Signals power manipulation at the senders and forwarding nodes makes any collected signal strength information unreliable. In addition, we assume that the bandwidth of the sensor network is severely limited, which precludes the exchange of physical-layer signal information. As a result we make no use of signal strength information.

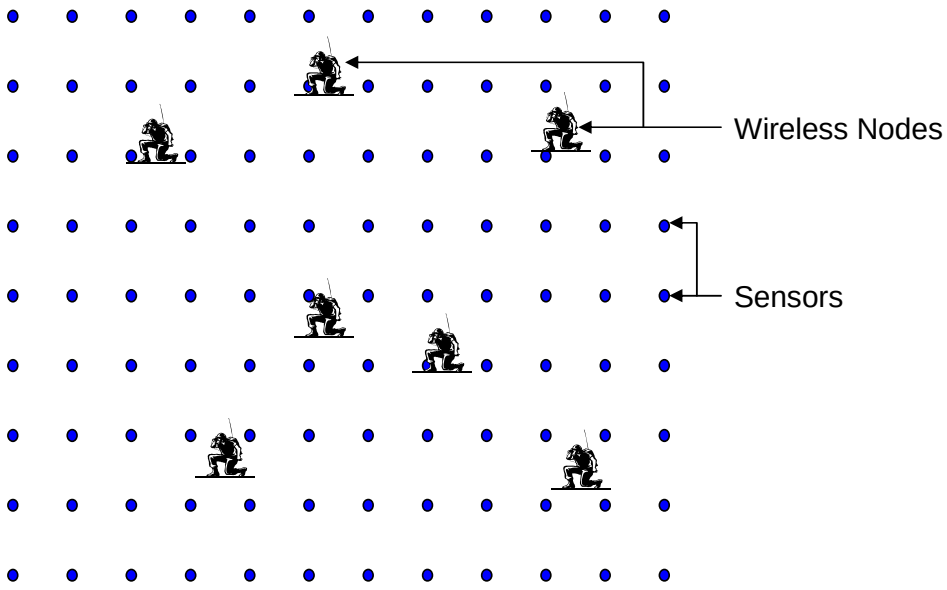


Fig. 37. Threat Model

- Sensors are equipped with omni-directional antennas. No directional information is available.
- Similarly, the time synchronization across sensor nodes is insufficient to allow for signal-propagation based location estimation. As a result, the sensor nodes (approximately) have access to packet timing information only.
- A sensor cannot associate a packet with a sender or receiver node.
- We assume that the network of sensors is sufficiently dense. This typically entails that the number of sensors is larger than the number of mobile nodes.
- The location of each sensor in the sensor network is known. Location information can be gathered in a variety of ways. For example, the sensors may be planted, and their location marked. Alternatively, sensors may have GPS capabilities. Finally, sensors may locate themselves through one of severals schemes

that rely on sparsely-located anchor sensor nodes [86, 87].

- In the following description we assume for simplicity that the sensors whose data is used for analysis form a grid topology as shown in Figure 37. If the sensors are randomly distributed in the field, we can select a subsets of sensors, that are located either on the grid or close to the grid. We expect that the estimation algorithms can be easily extended to cope with random sensor topologies.

- We assume that the communication between sensors does not interfere with the communication between wireless nodes. For example, sensors and wireless nodes may use different communication channels.

D. Data Collection and Pre-processing

The data collected from each sensor is a time series of counts of the packets "overheard" by the sensor. (While sensors cannot decrypt the packets or even associate packets with a mobile node, they can mark the time when a packet is received, and so *count* the number of packets received over any interval.) We use the time series $S_i = [s_i^1, s_i^2, \cdots, s_i^l]$ to denote the series of packet counts detected by Sensor i during a sequence of intervals of length T each. Since there may be several wireless nodes in the field, each sensor may be counting the packets from more than one wireless node. Similarly, the same packet may be counted by multiple sensors.

As for any data gathering application on sensor networks, power consumption and bandwidth limitations are important design issues. In our attack method, only packet *counts* are collected from the sensor, and so the resulting amount of data is significantly less than from collecting, say the time-stamp of each packet. In addition, we can use data compression or coding schemes designed for sensor networks such as ESPIHT [49], MEGA [88] to reduce the data volume that is caused by remaining

spatial redundancy across neighboring nodes or temporal redundancy at individual nodes.

E. Node Density Estimation

Unless *a priori* information is available about the presence of wireless nodes, any location effort must be preceded by some form of estimation of the *number* of nodes present in the area, also called *node density*. We use principal component analysis to estimate the node density. Principal component analysis (PCA) is a classical statistical method widely used in data analysis and compression. It is primarily used to reduce the dimensionality of datasets while best representing data. PCA reduces the dimensionality by transferring the correlated variables into a usually much smaller number of uncorrelated variables. For more details about principal component analysis, readers should refer to [89], for example. In our case, the data collected from the sensors is highly correlated, while the actual packet sending time series $M_i = [m_i^1, m_i^2, \cdots, m_i^l]$ from each mobile node are are significantly less correlated or dependent. Since sensors are geographically close, the sets of wireless nodes observed by neighboring sensors have large overlaps.

Traditionally, PCA requires the computation of the covariance matrix of the input data. In order to determine the dimensionality of the input data, one computes the eigenvalues of the covariance matrix and estimate the dimensionality based on the eigenvalues. Heuristic approaches such as Kaiser criterion [90] and screen test [91] are not reliable. The two so-called information criteria AIC [92] and MDL [93, 94] are widely used to determine the dimensionality. But they assume that the underlying data (in our case, packet count time series from each mobile node) is Gaussian distributed. This is not the case for our application where the packet-count time

series are the result of TCP or other protocol communications. We use an approach described in [95], where the probability of the data for each possible dimensionality is computed using a Bayesian model selection. It is shown that this method is robust against non-Gaussian distributed data.

One key step in PCA is the estimation of the covariance matrix based on the observations collected from the sensors. Given n sensors, this leads to $\frac{n(n-1)}{2}$ parameters to estimate. According to general measurement theory, the length of the data needed is at least five to ten times the number of parameters. So when n increases, the length of the data needed will increase quadratically. Thus it may not be practical to estimate the number of wireless nodes in the entire field directly using PCA. Instead, we can partition the overall area A into smaller *sub-areas* $A_1, A_2, \cdots$, and use the PCA method to estimate the number of nodes inside each sub-area separately. To determine the number of nodes in the entire area A, we can use averaging or cross validation methods to estimate it from the number of nodes inside each smaller area.

F. Node Location Estimation

Once the approximate number of wireless nodes is known, one can proceed to estimate the *location* of individual nodes. Just as for the case of node density estimation, we base the location estimation on the time series S_i of packet counts received by the sensors.

Since all packets are perfectly anonymized, the location estimation has only *aggregated* packet data available, since it cannot distinguish among packets sent by different nodes. In the following we describe how we use Blind Source Separation (BSS) to *de-aggregate* the packet count time series collected at a group of sensors into an estimation of the *per-node* packet count time series M_j of some node j. Based on

the estimated per-node time series we then use a proximity-based scheme to estimate the location of nodes.

In the following we first give an overview of Blind Source Separation and how we apply it to the problem of de-aggregating packet-count information. We then illustrate how we apply BSS to support proximity-based location of nodes.

1. Blind Source Separation

In order to de-aggregate packet count information we use - similarly to the flow separation in Chapter V - Blind source separation [69]. [1]

In our experiments, we use a powerful blind source separation algorithm proposed in [79]. This algorithm can jointly optimize several statistics of the same order, and it combines advantages of other powerful techniques as Fast-ICA [73], JADE [96] and SOBI [97]. The algorithm is shown to achieve a good performance when the amount of data available is small.

While the goal of BSS is to re-construct the original signals M_i (in our case the time series of packet counts sent by individual nodes), in practice the separated signals (we call these *components*) are sometimes only loosely related to the original signals. We categorize these separated components into three types: In the first case, the component is correlated to the time series of packet counts. The separated component in this case may have a different sign than the original signal. We call this type of component *individual* component. In the second case a component correlated to an *aggregate* of signals from several nodes. This happens when the packets of more than two wireless nodes can be "heard" by all the sensors. In such a case, the BSS algorithm would not be able to fully separate the signal mixture into the individual

[1] See Chapter V for a short overview to BSS.

components. Rather, while some components can be successfully separated, others remain aggregated. In the third case, components can represent noise signals. Noise in our case can be caused by packet collisions that prevent some sensors from "hearing" some packets. Noise can also be caused as artifacts from generating the packet timing sequences. For example a packet may be counted in the i^{th} interval for some sensor while for some other sensor the same packet may be counted in the $(i+1)^{th}$ interval due to transmission delay or imperfect timing.

For brevity, we call the second type *aggregate* component and the third type *noise* component.

2. Node Location Estimation Algorithm

In the following we describe an algorithm for the estimation of the location of several sender nodes based only on aggregate packet count time series as received and provided by the sensors. Figure 38 gives an overview of the algorithm: In a first step the sensor data is grouped into data from *blocks* of neighboring sensors. For example, we group the sensors in the grid into blocks of $c \times c$ sensors each as shown in Figure 39. Neighboring blocks are generally overlapping, and as a result sensors generally belong to several sensor blocks. (For the case of a quadratic blocks over a regular grid of sensors, most sensors belong to c^2 blocks.)

For each block of sensors we sequentially apply a BSS algorithm to recover the packet traces of mobile nodes in the sensing range. As a result of this block-by-block separation step, we are left with a large set of components as described in Section 1. Many of these components are either aggregates or noise components. In a second step we eliminate these components by identifying components that were identified in several blocks. This is achieved by identifying clusters of similar components across all blocks. This clustering step generates a set of components that have been detected by

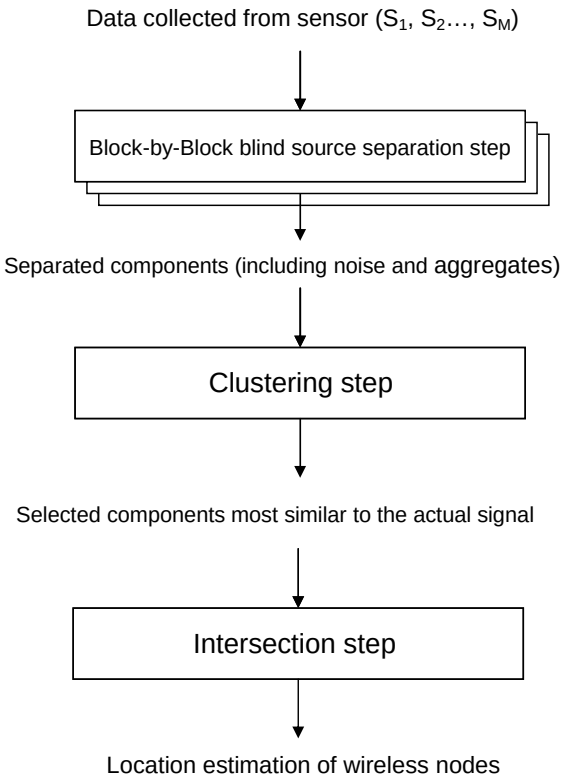


Fig. 38. Location Detection Algorithm

several blocks of sensors, and so are likely to be similar to the original signals. Once the original signals have been estimated through BSS and clustering, we estimate the location of the senders by intersecting the sensing ranges for all blocks that have separated component highly correlated to the original signals.We describe the three steps (BSS, clustering, intersection) in the following in more detail.

a. Blind Source Separation Step

For each sensor block we use a blind source separation algorithm to recover the original packet count time series from each mobile node. For the evaluation in the following sections we use the algorithm proposed in [79]. The output of this step is a set of c^2 *components* as described in Section 1 for each sensor block of size $c \times c$. We use R_i^j

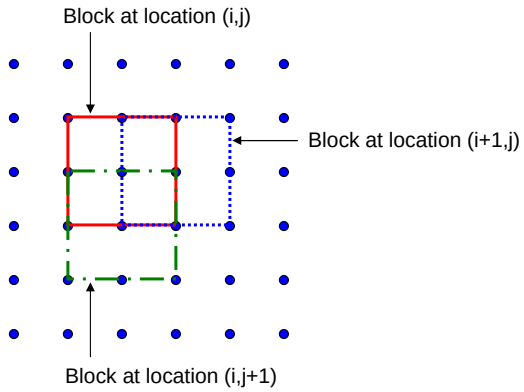


Fig. 39. Sensor Blocks

to represent the j^{th} recovered component from the i^{th} sensor block.

b. Clustering Step

In this step we eliminate those components that are likely to be noise or aggregate components from the set of components generated by the previous step. For this we use the following heuristic: If a component represents a real signal, the same component must likely have been detected and separated in at least a similar form by several sensor blocks. In comparison, a component that was generated because of some interference or other artifact is unlikely to have been generated by more than one block.

Based on this heuristic we identify clusters of similar components by using the cross correlation coefficient as measure for similarity as follows: Suppose that the separated components are of length l. Then these components can be represented as points in an l-dimensional space. We define the *distance* (our measure for similarity) between any of these two components as

$$D(R_i^p, R_j^q) = 1 - \left\| corr(R_i^p, R_j^q) \right\| \quad , \tag{6.1}$$

where R_i^p denotes the p^{th} component recovered from the i^{th} sensor block B_i and $corr(X, Y)$ denotes the correlation coefficient of components X and Y. We use the absolute value of the cross correlation because the separated components may be of different sign than the actual time series. So in this l-dimensional space, the highly correlated (similar) components will cluster together and uncorrelated components will be far from each other.

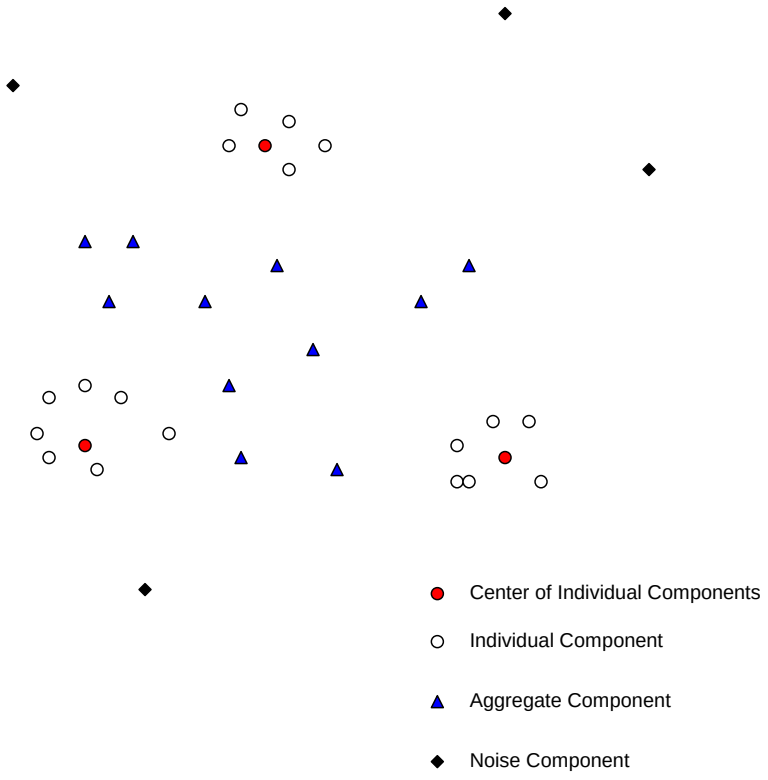


Fig. 40. Visualization of Distance between Separated Components

Figure 40 uses a two-dimensional representation to further illustrate the rationale for the clustering approach in this step: As shown in this figure, the individual components will form clusters since the individual components in a cluster are highly correlated to one of the actual time series M_j. The aggregate components on the

other hand will scatter in-between these clusters because they are correlated to more than one of the actual time series. The noise components will be distant both from each other and from the other components because noise tends to be local to some sensor block.

To eliminate some of the false correlation between components that cannot possibly be caused by the same sender node, we modify the distance function so as to take into account the sensing range of sensors in the blocks:

$$D(R_i^p, R_j^q) = (1 - \left\| corr(R_i^p, R_j^q) \right\|) \cdot Overlap(i, j) \quad , \tag{6.2}$$

where $Overlap(i, j)$ is a binary function defined as

$$Overlap(i, j) = \begin{cases} 1, & \text{if sensing ranges of sensors in Block } B_i \text{ and Block } B_j \text{ overlap,} \\ 0, & \text{otherwise.} \end{cases}$$

In addition to random false correlations, the binary function $Overlap(u, v)$ addresses the case when a flow traverses several hops in the wireless ad-hoc network, and the forwarded flow may give rise to similar components appearing along the path of the flow. This is taken into account by the overlap function.

Each cluster of components has a *center component*, which is the component with the minimum average distance to every component in the cluster. Among all the possible center components we select the center components $R_1, \cdots, R_K$ of the K largest clusters, where K is the estimated number of nodes in the area. (The value for K is either known *a priori* or is estimated using the node density estimation methods described in Section E). We note that it is highly unlikely that either aggregate or noise components are selected as center components: (a) Aggregate components are unlikely in the center of clusters, and (b) noise is local to a small group of sensors at best, and gives rise to a small cluster. As a result, the K selected center components

will be highly correlated to the packet count times series $M_1, \cdots, M_K$ of nodes in the network.

c. Intersection Step

Once the components that likely represent packet count time series of the nodes in the area have been identified, we proceed to determine the *location* of the nodes. We locate a sending node by intersecting the sensing ranges of blocks that are likely to "hear" the node. For this we select sensor blocks that have components that are closely correlated with the likely time series of the nodes in the area[2]. The rationale is that for the sensors in a sensor block to hear a node, they must have sensed a signal that is at least similar to the signal generated by the node. This means that sensor blocks with components that correlate with any of the K center components are likely to hear a sending node. We therefore determine the likely location of a node by geographically intersecting the sensing areas of the sensors in those sensor blocks that have highly correlated component with a center component determined in the previous step (Section b).

When sensor blocks are erroneously classified as hearing a node when they do not, the geographical intersection of their sensing area fails to determine the location of the node. Particular attention must be therefore paid to correctly identify the correct sensor blocks. For example, simply correlating the components of the sensor block with a particular center component may lead to too many false positives because geographically very distant components may occasionally correlate with a particular center component. For example, correlated components may appear along the path

[2]As we illustrate in Section 2, sensor blocks in the immediate neighborhood of a sender node are often not able to successfully separate the component representing the sender node packet time series, and are therefore discarded.

of a connection, and so give rise to very distant geographical sensor blocks that are erroneously classified to be able to hear the node. When in fact the node is well outside their sensing range. In such cases, the geographical area intersection approach fails to locate the node. We therefore use the following method to identify the relevant sensor blocks and perform the area intersections. We borrow techniques from image processing, and therefore make use of its terminology as well.

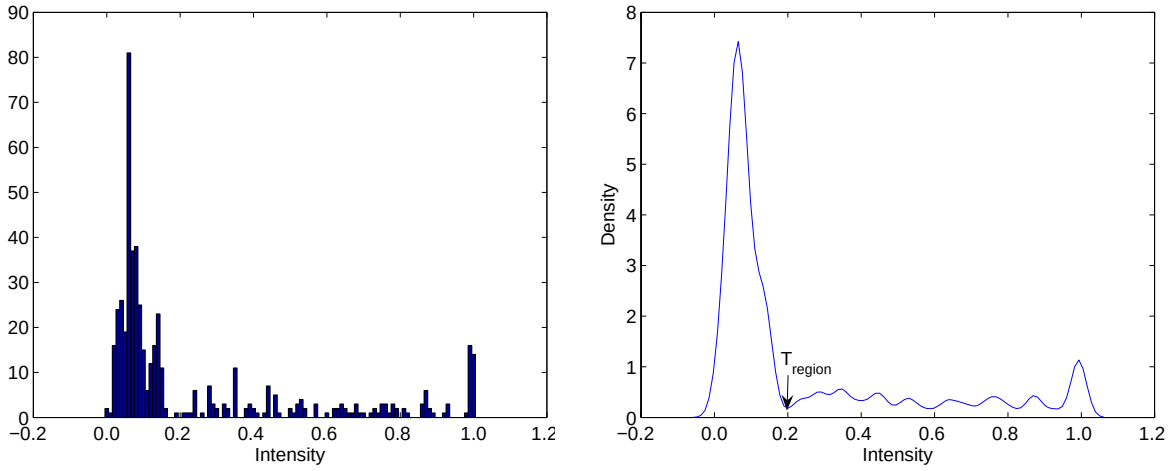
- We generate the "image matrix" IMG_k of correlation coefficients (the "intensity") as follows: each entry in the matrix $IMG_k(i,j)$ represents the maximum correlation between the center component R_k and all c^2 components of the block, say B_u, at location (i,j):

$$IMG_k(i,j) = \max_{1 \leq p \leq c^2} (\|corr(R_u^p, R_k)\|$$

- Apply an edge detection algorithm on IMG_k to find the "edges", i.e. the positions where the intensity in IMG_k changes quickly. In our experiments, we use the zero-crossing edge detection method, with the sensitivity threshold set to zero. This approach finds closed contours of the intensity image. These contours divide the area represented by IMG_k into regions. At this point we have a set of geographical regions each with components that correlate with the center component R_k.[3]

- Now we need to identify the regions that most likely contain sensors that "hear" the sender node. We therefore identify regions where components are highly

[3]The function of the edge detection step is to partition the entire field into areas that need to be considered for intersection and areas that are outside of the sensing range. The rationale is that a sudden drop in the correlation intensity marks the edge of area where a sending node can be "heard". Components outside the sensing range can be safely ignored. As a result, the intersection algorithm becomes significantly more robust.



(a) Histogram **(b) Probability Density Estimation**

Fig. 41. An Example Distribution of Intensity in IMG_k

correlated with the center component R_k. We proceed by first computing the "average intensity" of each region. Regions with an average intensity smaller than some threshold T_{region} (see below) are then discarded.

- We perform the intersection of the sensing area of sensor blocks in the remaining regions as follows: Sort the points $IMG_k(i, j)$ in the remaining regions in order of decreasing intensity. Starting with the highest-intensity point, add subsequent points by intersecting their sensing range. Stop when you either run out of points or the new point's sensing area is disjoint from the computed intersection area, causing the new intersection area to disappear.

- The resulting intersection area is the suspected area of location of a node.

The threshold T_{region} is used to separate correlation levels due to the sensor hearing the node from correlation levels that are merely accidental. In order to determine T_{region}, we take advantage of the fact that the correlation coefficients of "acciden-

tally" correlated components are significantly smaller than components from blocks that hear the node. In addition, due to the central limit theorem, the coefficients of accidentally correlated components are normally distributed.

A representative distribution from our experiments is shown in Figure 41. As we can observe, most of the intensity in IMG_k has a very small value since the components recovered from most sensor blocks are not correlated to the component R_k. We use a kernel density estimation method to estimate the distribution of the intensity in IMG_k and set T_{region} to be at the "right boundary" of the normal distribution. (In our case we do this by setting the threshold T_{region} to be the point where the density is minimal between the peak and 0.5). An example of the density estimation and the selection of T_{region} is shown in Figure 41 (b).

G. Evaluation

We evaluate the effectiveness of the proposed algorithms for node density and node location estimation in a series of simulation experiments in the ns-2 [65] network simulation environment.

1. Experiment Setup

In the following experiments the simulated field is a 1600m × 1600m square area. The distance between two neighboring sensors in the sensor grid is 50m. The location of the wireless node is restricted to a 1000m × 1000m *center area* of the simulated field to eliminate boundary effects. The wireless network interfaces of both wireless nodes and sensors are modeled according to the commercial Lucent WaveLan radio interface, which has a nominal radio range of 250m. For the sensors the transmission function is disabled, so that they can only eavesdrop on the traffic. All simulations

have a duration of 200 seconds. The packet count data is sampled with a sample interval of 50 ms. We place 20 wireless nodes inside the center area. There are 36 randomly generated TCP flows in the wireless ad-hoc network. To make sure that every wireless node sends packets, every wireless node has at least one TCP flow that originates from it. The size of the sensor blocks used in the experiments is 3 by 3.

Since the radio range of wireless nodes is 250 m and every wireless node is sending packets, the interference from different wireless nodes renders the location detection based on physical signal strength hard and inaccurate. (Four wireless nodes can cover nearly all the center area.)

2. Node Density Estimation

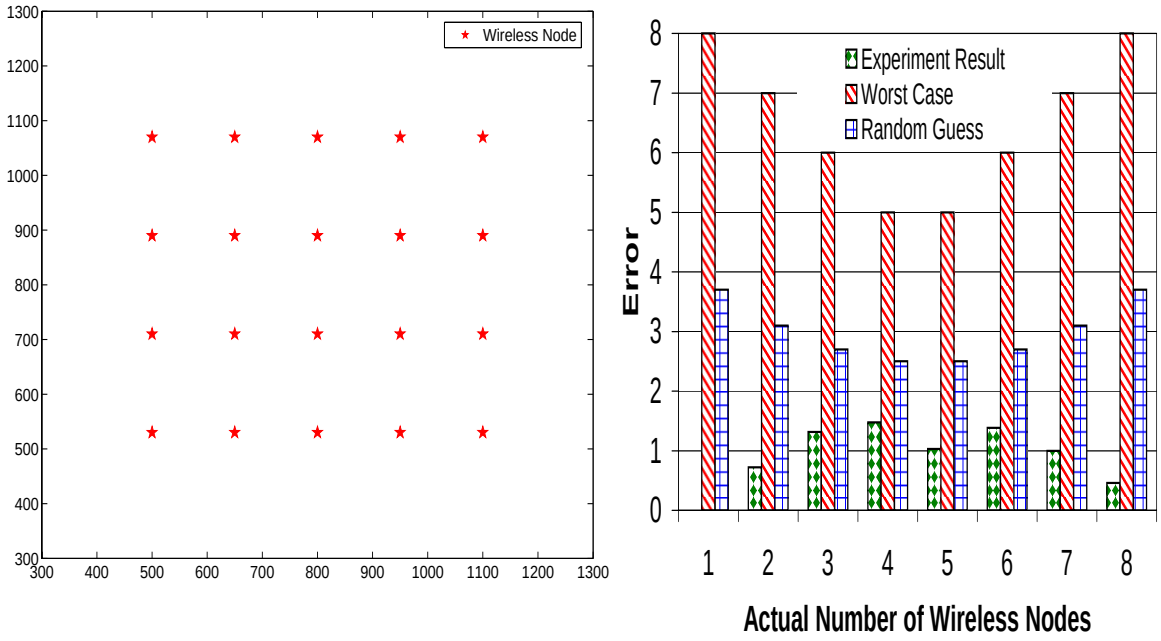
We use the simple arrangement of wireless nodes shown in Figure 42 (a) to evaluate the the node density estimation algorithm. The wireless nodes are arranged in a grid, and the vertical and horizontal distance between neighboring sensors are 180m and 150m respectively. (The horizontal and vertical distances between neighboring sensors are chosen so that every wireless node is in the radio range of the nodes in its 8-neighborhood.)

We use a mean-error metric to evaluate the algorithm:

$$e_n = \sum_{m=0}^{c^2} p(m|n) \, \|m - n\| \quad , \tag{6.3}$$

where $p(m|n)$ denotes the probability of the algorithm deciding that there are m wireless nodes in the sensing area of one sensor block when the actual number of wireless nodes in the sensing area is n. Since in this series of experiments the size of the sensor blocks is 3×3, the number c^2 in Equation (6.3) is 9.

From Figure 42 (b) we observe that the node-density estimation algorithm is off by about 1 node in average. Since the dimensionality estimation algorithm [95] in



(a) Node Arrangement

(b) Mean Error of Node Density Attack Algorithm

Fig. 42. Node Density Estimation

Section E assumes that the number of actual signals is *less* than the original dimension of the data, which is c^2 for our case, we only show the error when the actual number of wireless nodes present ranges from 1 to $c^2 - 1$. Please note that if two wireless nodes are in sufficiently close proximity (e.g. two central nodes in Figure 43) such that every sensor in one sensor block receives packet from both nodes, then it is impossible to differentiate the tow nodes based on the collected data. As a result, the two nodes are considered as a single node.

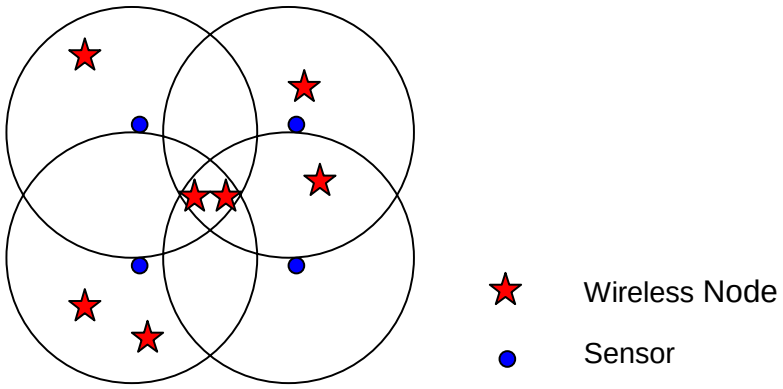


Fig. 43. Nodes in Close Proximity

3. Effectiveness of Location Estimation

a. Performance Metrics

As described in Section F.2.c, the output of the location estimation algorithm is the suspected area of location of a node. To evaluate the performance according to the suspected area, we quantize the area of whole field using 5 m $\times$ 5 m tiles. The suspected area is represented by a set of points inside the suspected area, each point representing the corner of the corresponding tile. Two metrics are used to evaluate the area: One is the *mean error distance* between the points inside the suspected area and the actual location of a wireless node. The other is the *standard deviation of the error distance* between the points inside the suspected area and the actual location of a wireless node. The first one measure the *accuracy* of the detection algorithm and the second measures the *precision* of the detection algorithm. If we cast the evaluation of the estimation algorithm in terms of evaluating a statistical estimator, the accuracy corresponds to the *bias* of the estimator and the precision corresponds to the *variance* of the estimator.

These two metrics are similar to the metrics defined earlier in signal-based location studies [47]: probability of the actual wireless node within the suspected area

(for accuracy) and size of the area (for precision). We feel that error-distance based statistics (mean and standard deviation) better describe the estimator than the metrics used in [47]. First, our accuracy measure describes better how the node is located inside the suspected area (corner vs. center). Second, the *shape* of the suspected area is considered as well. It is of little help, for example, to know that a node is likely located in a small area when the latter is narrow but very long.

To evaluate the intermediate result of the clustering step in Section b, we compare the K selected center components with the actual packet count time series of corresponding wireless nodes. The metrics used for comparison is the absolute value of the cross-correlation. We use absolute value here to account for the possibility that the separated component is of different sign than the time series.

b. Performance

We run our algorithm on three different types of node arrangements: *grid* arrangement, *random* arrangement and *clustered* arrangement. Examples of typical results of our location algorithm are shown in Figure 44 and 45. Please note that in Figure 45 two relatively large suspected areas are removed to prevent overlapping with other suspected areas. The two larger suspected areas are caused by the two pairs of closely located nodes near point A and point B, respectively, in Figure 45. These closely located nodes cannot be differentiated by the sensor grid, so the number of actual differentiable nodes is less than the number of nodes we know. These two larger suspected area are caused by the two "mistakenly" selected center components.

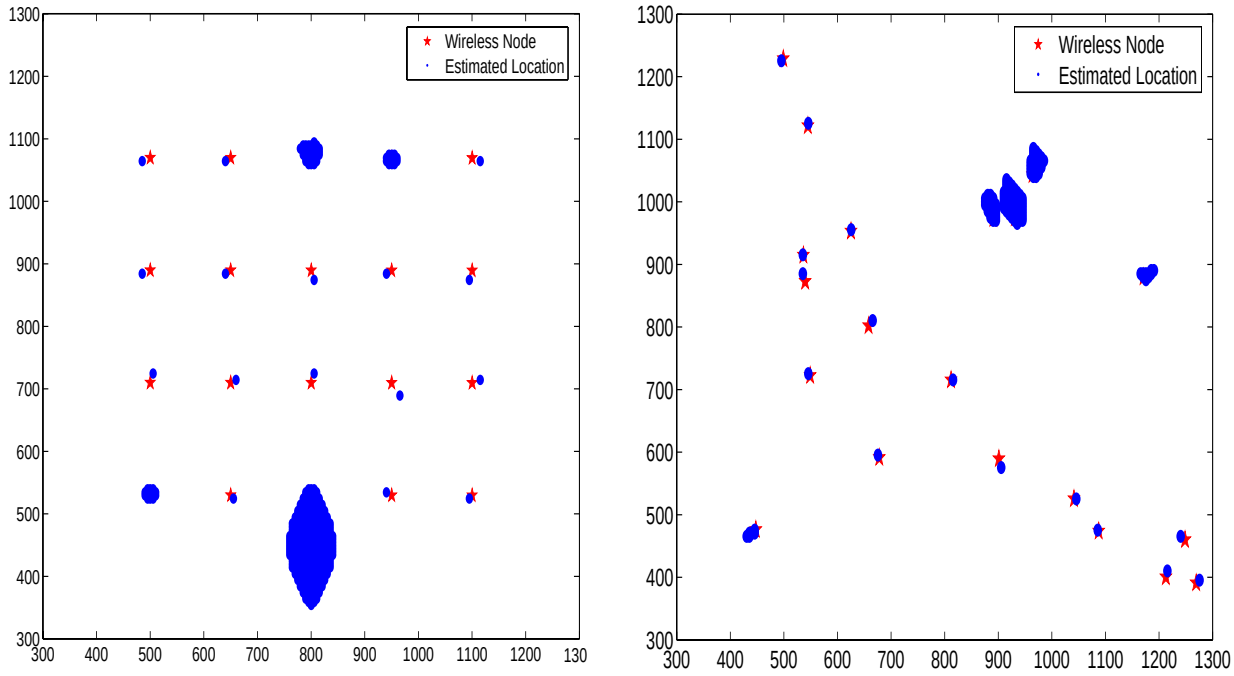
The three typical examples shown in Figure 44 and Figure 45 illustrate that our algorithm can make accurate and precise detection for all three kinds of node arrangements.

To quantitatively evaluate our algorithm, we run extensive simulations for both

random and clustered arrangements. For clustered arrangements, the locations of wireless nodes are generated by a Gaussian distribution with standard deviation of 100m for both x and y axes. The mean of the distribution is chosen to arrange the wireless nodes around the center of the field. Figure 46 shows the accuracy and precision resulting from the experiment. From Figure 46, we can make the following observations. For both types of arrangements, the algorithm accurately locates the nodes. For random arrangements, node location estimations have an error of less than 30m in more than 90% of the cases. For clustered arrangements, more than 85% of detections have an error of less than 30 m.

For both kinds of arrangements, the algorithm precisely locate nodes. For random arrangements, more than 90% of the detections have error standard deviation less than 20m. For clustered arrangements, more than 85% of the detections have error stand deviation less than 20m.

As to be expected, the performance of the algorithm degrades when nodes are clustered. This is because of three reasons: (a) Clustered arrangements tend to have wireless nodes located close enough so that the sensor grid can not differentiate them. (b) For clustered arrangements, increased contention for the wireless medium causes increased dependency between packet sending time series from different wireless nodes. Since BSS relies on the independence of underlying signals, its performance is affected by increased contention in clustered arrangements. (c) For clustered topologies, sensor blocks may overhear packets from more nodes than the number of sensor in the block. Whenever the number of signals exceeds the number of sensors, BSS runs into the so-called *overcomplete base* problem where some components can not be separated. In summary, effect (a) is equivalent to "thinning out" the sensor network. In addition, both effects (b) and (c) cause the performance of blind source separation to degrade.



(a) Grid Topology (b) Random Topology

Fig. 44. Location Estimation on Different Topologies

Figure 46 indicates that in some cases the location estimation algorithm performs significantly worse for random arrangements than for clustered arrangements. While this appears counter-intuitive at first, it is easy to explain: Since clustered nodes are tending to be close to each other, their signals cover a smaller region than the nodes in random arrangements. Therefore, large errors for clustered arrangements are unlikely.

We also evaluate our approach based on topology. The 95% confidence intervals for per-topology mean error distance are from 14.9618 to 58.5623 and from 21.9886 to 42.7431 for random topologies and clustered topologies.

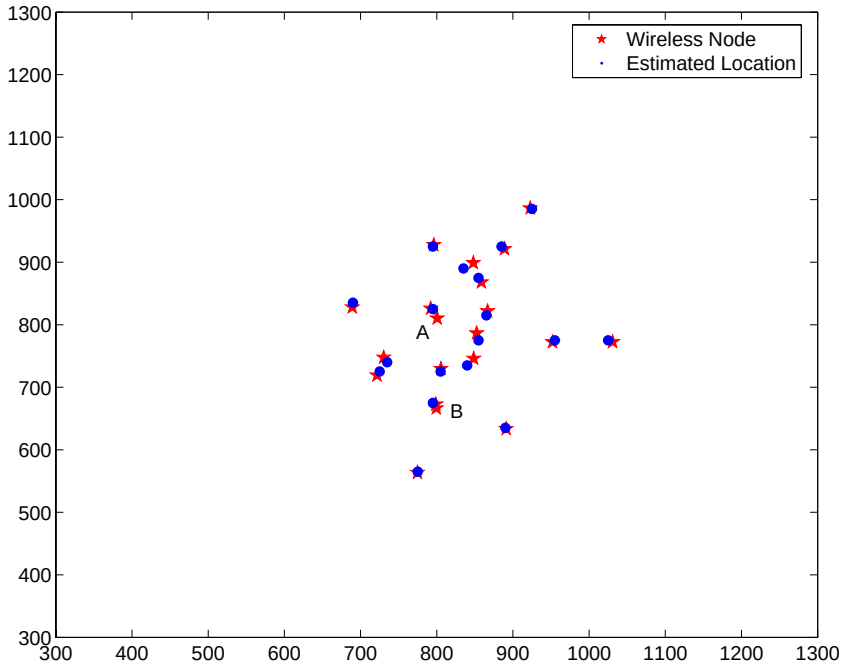
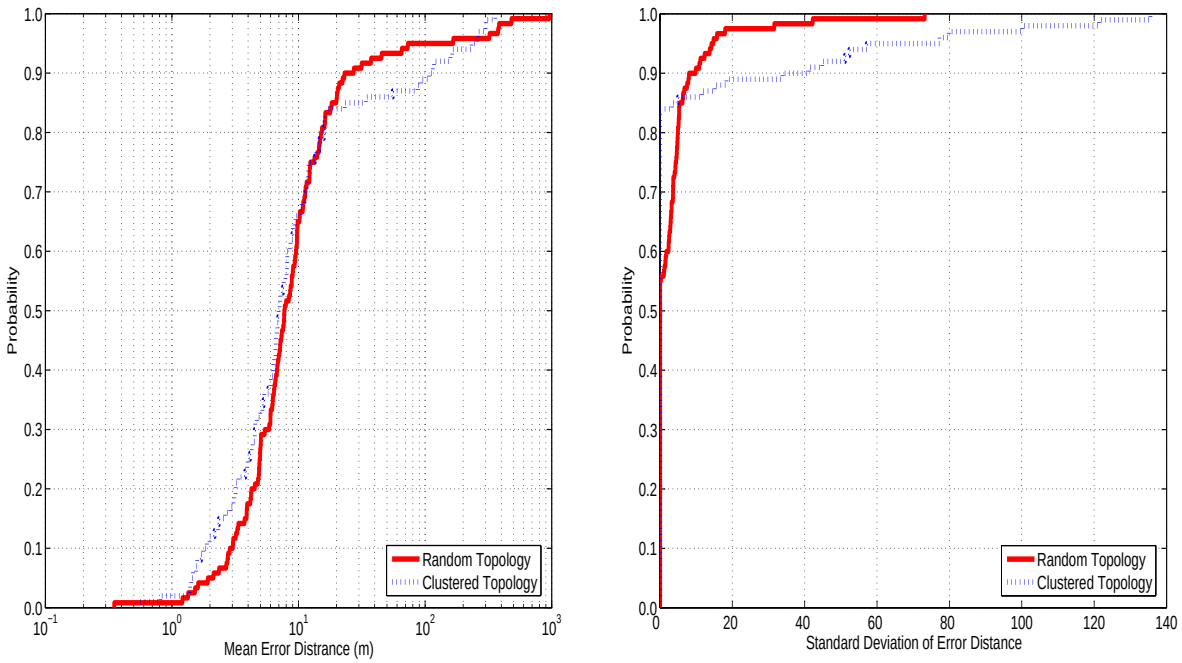


Fig. 45. Location Estimation on a Clustered Topology (Note: Two larger suspected area are removed to prevent overlapping)



(a) Mean Error Distance (b) Standard Deviation of Error Distance

Fig. 46. Performance of Location Estimation Algorithm (Empirical CDF)

c. Constant-Rate Traffic

In this experiment we evaluate our location detection algorithm against constant-rate UDP traffic. The grid topology shown in Figure 42 (a) is used in this experiment. Each wireless node sends UDP packets to one of its neighbors. The choice of neighbor is made so that two loops are formed, with the outside nodes forming an outer loop and the inner nodes an inner loop. The packet sending rate of each wireless node is 40 packet/s and the assumed bandwidth utilization is about 80%. The goal of this setup is to evaluate an arrangement as uniform as possible, thus making the separation and location problem maximally hard.

The results of this experiment are shown in Figure 47. We observe that the

location detection algorithm is also effective against constant-rate traffic and heavy traffic. While the flows are constant-rate at sender application level, they are sufficiently perturbed by the 802.11 MAC protocol, which adds enough timing signature to the flows, and so helps to separate the traffic. This experiment also illustrates that traffic padding at network layer or above is largely ineffective. A MAC level traffic padding scheme that consider both the media control protocol and bandwidth efficiency is needed.

H. Discussion

The mechanisms used to estimate density and location information can be used to infer additional information about the wireless nodes, as well:

1. Traditional sender/receiver/route anonymity

For each intensity image IMG_k, we can apply an edge detection algorithm to reveal the sender/receiver relationship as well as information about the communication path. The edge detection algorithm used for this purpose is different from the algorithm used in Section F.2.c, where we use the *Zero Crossing* method to detect large jump in the intensity image IMG_k. Here we apply the *Canny* [98] method instead, which is good at detecting weak edges, to visualize the relationship between sender and receiver. The result of an example attack is shown in Figure 48. From the intensity image shown in Figure 48(a) we can observe the relationship between the sender and the receiver. A contour of the route taken by a flow is shown in 48(b). (In Figure 48(b), the locations of the sender and receiver are marked with star.)

2. Motion privacy

To detect the motion of the wireless nodes in a field, we periodically compare the K selected components using cross correlation. Suppose the attacker finds out that component R_i^t at some time t is highly correlated to component $R_j^{t+\delta}$ at some later time $t+\delta$, and the location of component R_i^t and $R_j^{t+\delta}$ is estimated to be $area_i^t$ and $area_j^{t+\delta}$ respectively. The attacker can infer that a node has moved from $area_i^t$ to $area_j^{t+\delta}$. From the analysis of node location privacy, we can get the location of k nodes at time t and $t+\delta$. Under the assumption that routing does not change dramatically from time t to $t+\delta$, we can find the correspondence between k signals at time t and k signals at time $t+\delta$ through correlation. When routes do change from time t to time $t+\delta$, the timing behavior of flows can change as well, due to new contention situations or different path lengths. If routing does change, we can take advantage of correlation in the space domain: since the speed of nodes is limited, for small values of δ the location of a node can not vary indeterminately.

3. Identity privacy

If *a priori* information is available about a wireless user, such as a model for communication or motion, the identity can be derived by correlating the K separated center components with the available models.

In this chapter, we apply blind source separation algorithm to degrade a number of anonymity measures in networks. The proposed attack methods can apply to not only wireless ad-hoc networks but also infrastructure-based wireless network since the methods do not make use of any information of base stations. Generally, our algorithm can be used to locate objects in a field by using signals from different type of sensors. For example, acoustic sensors can used to locate snipers in a field. More

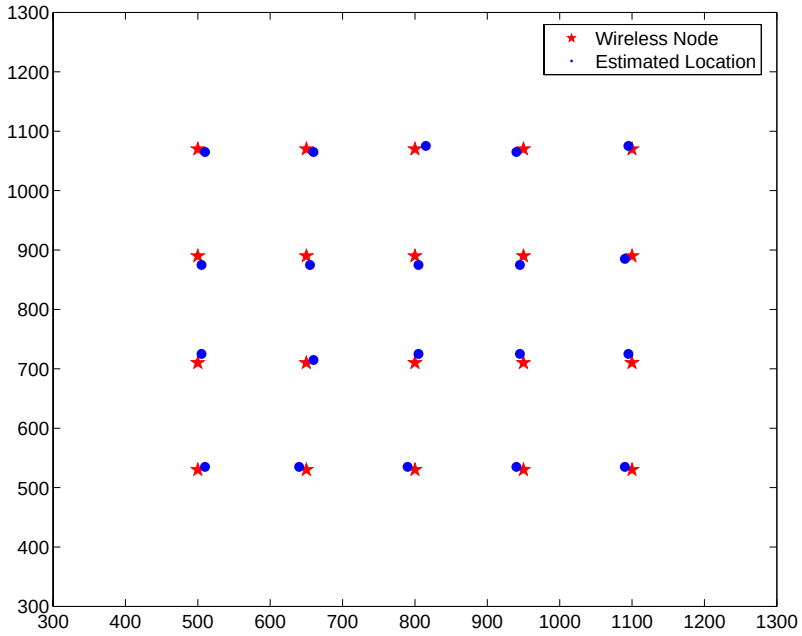


Fig. 47. Location Estimation Result (Constant Rate Traffic)

(a) Intensity Image IMG_k (b) Edge Detection

Fig. 48. Sender/Receiver/Route Anonymity Attack

generally, the algorithm can be used to analyze any observed mixture data that are linear combinations of the underlying signals.

I. Summary

In this chapter, we focus on a number of anonymity issues in wireless networks. We propose algorithms for the estimation of node density and of node location. The approach is based on principal component analysis for the estimation of nodes in the network (density estimation) and on independent component analysis (blind source separation) for the de-aggregation of the presumably fully anonymized packet trace information. Our experiments show that the attacks are very effective. Two new metrics to evaluate location privacy attack are proposed. They can capture accuracy and precision of the location privacy algorithm and differentiate different shapes of estimated areas. We applied our location privacy attack algorithm to different node arrangements. The result of location privacy attacks can be used to attack traditional sender/receiver anonymity and motion privacy as well.

The fact that the proposed schemes require from the sensors only the capability to receive and count 802.11 packets indicates that one should be able to deploy similar

schemes on nodes in ad-hoc networks, for example, for intrusion detection purposes: The ad-hoc nodes could easily collect the data necessary to identify active intruders and to pin-point their location.

CHAPTER VII

ENGINEERING OF ANONYMITY NETWORKS

A. Motivation

Researchers proposed various definitions to quantify anonymity, such as *anonymity set size* [36], *effective anonymity set size* [5] and entropy-based *anonymity degree* [4]. While the metrics led to an increasingly better understanding of anonymity, they tend to focus on the anonymity of a *single* message under a *single* anonymity attack. In practice however, metrics are needed that take into account realities of today's use of networks. a.) Communication settings in real systems range from single messages, to message groups, to streams and FTP transfers. b.) Sophisticated attacks can resort to a variety of techniques to break anonymity: flow correlation attacks, intersection attacks [99], trickle attacks [12], and so on.

A measure for the anonymity degree should satisfy a number of requirements: First, the anonymity degree should capture the *quality* of an anonymity system. It has been shown for example that information theoretical means, such as entropy, are more accurate for comparing anonymity systems than, say, anonymity sets. Second, the anonymity degree should take into account the topology of the network or that of any overlay defined by the anonymity system. The topology influences how much information can be gathered by an attacker, and thus has an impact on the system anonymity degree. For example, a system of fully-connected nodes will have a different anonymity degree from a chain of nodes. Third, the anonymity degree, as measure of the effectiveness of the anonymity system. While a large number of users clearly contributes anonymity, this not necessary reflects on the quality of the anonymity system. should be independent of the number of users. Finally, the anonymity

measure must be independent of the threat model, as attackers may use a variety of attack techniques, or combinations thereof, to break the anonymity.

Since the goal of anonymity attacks is to infer the communication relations in a system despite countermeasures, it is natural to model such attacks as covert channels. Increased interest has focused on the interdependence of anonymity and covert channels [40, 100]. The imperfectness of an anonymity system will result in the information leaking from the system. This information leakage can be evaluated in form of a covert channel. The designer of an anonymity system generally faces the question of how much information may leak from the anonymity network given the unavoidable imperfectness of the anonymity network and how this may affect the anonymity degree. The imperfectness of an anonymity system will result in the information leaking from the system. This information leakage can be evaluated in form of a covert channel.

The work presented in this chapter takes a system-level view of covert channels and anonymity, and differs from previous work, such as [38, 39, 40], in two ways: First, we assume that the existence of various sources of information leakage in the elements (mixes, batchers, padders, $\cdots$) of an anonymity system are a reality that system designers and operators have to deal with. Some of the resulting covert channels can be identified and either measured or analyzed using techniques described in [38, 39][1] In addition, any cautious anonymity system designer or operator must assume that even mixes presumed to be perfect are not so, even if the particular weakness is not know *a priori*. In this chapter, we use covert channel capacity as a generic measure to model weaknesses (known or unknown) in the anonymity system infrastructure. This gives a tool for designers or operators to uniformly describe both known weaknesses

[1]Statistical techniques can be used as well, as we describe in Section B.

(i.e. results of attacks), or merely suspected ones, and to analyze their effect on the anonymity provided by the system. Second, the anonymity degree of the mix network is a result of system-level effects: changes in the user population or application mix affect the anonymity provided. So do topology of the anonymity system and routing preferences within the system. As a result, there is no one-to-one mapping from the anonymity degree to covert channel capacities of elements in a mix network and *vice versa*. In this chapter, we investigate the relationship between anonymity degree and covert channel capacity in terms of what effect one has on the other.

B. Anonymity Degree

A number of attacks have been described recently that give raise to reasonably high capacity channels on mixes. Several attacks to simple mixes lend themselves to an accurate analysis of the exploited covert channels, such as in [38, 39, 40]. For other attacks the covert channel capacity can be merely estimated, using statistical means. Examples are intersection attacks [99], timing attacks [34], Danezis's attack on continuous mixes [35], and the flow correlation attack. The timing attack [34] uses cross-correlation to match flows given the packet timestamps of the flow. Danezis's attack on continuous mix [35] uses likelihood ratios to detect a flow in aggregate traffic. The flow correlation attack employs statistical methods to detect TCP flows in aggregate traffic. The flow correlation attack can achieve high detection rates for all the mixes described in [12] and for continuous mixes.

1. Attack Model

We model a single mix (Figure 49) as a communication node that connects m senders $S = (s_1, s_2, s_3, \cdots, s_m)$ to n receivers $R = (r_1, r_2, r_3, \cdots, r_n)$. Every Sender s_i may

communicate to every Receiver r_j. We say that a *communication* exists between s_i and r_j whenever s_i communicates to r_j. A communication between s_i and r_j is denoted by the term $[s_i, r_j]$. It can consist of a single packet being sent, or of an established flow.

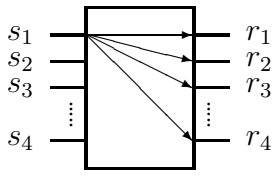


Fig. 49. Model of a Mix

We model an *attack* to such a node in terms of its effectiveness in determining who is talking to whom: the set of probabilities $p([s_u, r_v]_s|[s_i, r_j]_a)$ denotes the probability that Communication $[s_u, r_v]_s$ is suspected, given that communication $[s_i, r_j]$ is actually taking place. In other words, a probability $p([s_u, r_v]_s|[s_i, r_j]_a)$ denotes the probability of erroneously suspecting s_u sending to r_v when in actuality s_i is sending to r_j. This model allows for an accurate description of many different attacks, as the probability $p([\cdot, \cdot]_s|[\cdot, \cdot]_a)$ can be defined based on the observation of single packets, a number of packets, a flow or a session, depending on the particular attack method used. For example, the passive attack described in [33] determines a flow successfully when the flow is alone on a link. So the probability $p([s_i, r_j]_s|[s_i, r_j]_a)$ of correctly identifying communication $[s_i, r_j]$ is equal to the chance that the flow is alone on the output link from the mix to Receiver r_j. Alternatively, Danezis's attack on the continuous mix, the probability $p([s_i, r_j]_s|[s_i, r_j]_a)$ is the probability that the likelihood of the hypothesis assuming that the flow of interest is going through the link between the mix and Receiver r_j is greater than any other hypothesis assuming that the flow of interest

is going to any other receiver. Finally, for the flow correlation attack, the probability of $p([s_i, r_j]_s|[s_i, r_j]_a)$ is equal to the probability that the mutual information between the flow of interest and the aggregate traffic on the link between the mix and Receiver r_j is larger than the mutual information between the flow of interest and the aggregate traffic on any other outgoing link.

We note that the attacker may use different attack methods to estimate the probability $p([s_u, r_v]_s|[s_i, r_j]_a)$ for different communications on different mixes, or even on the same mix.

The model above describes attacks on sender-receiver anonymity, where both sender and receiver are anonymous. It can be easily extended to sender anonymity or receiver anonymity, that is, cases where the sender only or the receiver only are anonymous, respectively. For example, we can describe the results of a sender-anonymity attack in terms of $p([s_u, *]_s|[s_i, *]_a)$ or just $p([s_u]_s|[s_i]_a)$. To keep the following discussion simple and general, we will focus on sender-receiver anonymity, with the understanding that sender anonymity or receiver anonymity can be modeled just as well.

2. Proposed Anonymity Degree

We define a new measure, D, for the anonymity degree based on the following rationale: Let the random variable $[S, R]_a$ indicate the *actual* sender and receiver pair, and the random variable $[S, R]_s$ in turn indicate the *suspected* sender and receiver pair. If the attack identifies the communicating pairs with high accuracy, then the dependence between the two random variables $[S, R]_a$ and $[S, R]_s$ will be high.

In general, the dependence of two random variables can be measured using the *mutual information* of the two random variables. The mutual information $I(X; Y)$ of

two random variables X and Y is a function of the entropies of X and Y as follows:

$$I(X;Y) = H(X) - H(X|Y). \tag{7.1}$$

Therefore, the effectiveness of the attack can be described in terms of the mutual information $I([S,R]_a;[S,R]_s)$.

To give a more figurative interpretation of mutual information as measure of the attack effectiveness, we use an analogy to communication channels: Mutual information is typically used to describe the amount of information sent across a channel from a sender X to a receiver Y where $H(X)$ is the information at the input of the channel and $H(X|Y)$ describes the information attenuation caused by noise on the channel. (See [101] for an easy-to-read introduction to the information theory used in this context.) This gives an intuition of why mutual information describes the effectiveness of an anonymity attack: Let $[S,R]_a$ be the random variable that describes the actual sender and receiver pair. Let the attacker's estimate of $[S,R]_a$ through observation of the system, i.e. the attack, be $[S,R]_s$. The information carried through the observation channel provided by the attack is therefore $I([S,R]_a;(S,R]_s)$. The higher this carried information, the more accurate the anonymity attack. Using the textbook definition for entropy, the effectiveness of an anonymity attack can be described as follows:

$$\begin{aligned} I([S,R]_a;[S,R]_s) &= H([S,R]_a) - H([S,R]_a|[S,R]_s) \\ &= \sum_{[s,r]_a,[s,r]_s} p([s,r]_a,[s,r]_s) \log \frac{p([s,r]_s|[s,r]_a)}{p([s,r]_s)} \quad . \end{aligned} \tag{7.2}$$

In Equation (7.2), we let $p([s,r]_a,[s,r]_s) = p([s,r]_a)p([s,r]_s|[s,r]_a)$ and $p([s,r]_s) = \sum_{[s,r]_a} p([s,r]_a,[s,r]_s)$. We let $p([s,r]_a)$ denote the *a priori* probability of s communicating to r, typically derived from the expected traffic from s to r.

We can now formulate the *Anonymity Degree* D as a function of the attack effectiveness as follows:

$$D = 1 - \frac{I([S,R]_a;[S,R]_s)}{\log(m \cdot n)} \quad . \tag{7.3}$$

Since $I([S,R]_a;[S,R]_s) \leq H([S,R]_a) \leq \log(m \cdot n)$, we use $\log(m \cdot n)$ to normalize the anonymity degree into the range of $[0,1]$ in Equation (7.3). Alternatively, one could choose $H([S,R]_a)$ as normalization factor. However the latter depends on *a priori* probability of communication between each pair of sender and receiver. The impact of this *a priori* probability been taken into account by the term $p([s,r]_a)$ in Equation (7.2).

The equality $I([S,R]_a;[S,R]_s) = H([S,R]_a)$ holds when perfect identification is achieved, that is,
$p([s_i,r_j]_s|[s_i,r_j]_a) = 1$ for each pair of sender and receiver. This corresponds to the situation where anonymity is totally broken, in which case the anonymity degree measure D is zero

3. Relationship to Previous Anonymity Degree Definitions

The anonymity degree definition D is a generalization of the entropy-based definitions proposed in [4, 5]. In fact, we can rewrite the attack effectiveness $I([S,R]_a;[S,R]_s)$ as

$$\begin{aligned} I([S,R]_a;[S,R]_s) &= H([S,R]_s) - H([S,R]_s|[S,R]_a) \\ &= H([S,R]_s) \\ &- \sum_{[s,r]_a} p([s,r]_a) H([S,R]_s|[S,R]_a = [s,r]_a) \end{aligned} \tag{7.4}$$

In Equation (7.4), the term $H([S,R]_s|[S,R]_a = [s,r]_a)$ represents the conditional

entropy of the suspected sender-receiver pair distribution given the communication $[s, r]$. This corresponds to the anonymity degree definition described in [5] and also to the core of the anonymity degree defined in [4].

In our mutual-information based anonymity degree, the entropy-based degree is included by averaging according to $p([s, r]_a)$, the *a priori* probability of traffic between each pair. In comparison with previous entropy-based definitions (for example [4, 5]), our proposed definition describes the anonymity provided by a network of mixes.

C. Anonymity-based Covert Channels

Less-than-perfect anonymity systems give raise to a form of covert channel that is exploited by anonymity attacks. We call this form of covert channel *anonymity-based* covert channel. The input symbols of this type of covert channel are the *actual* sender-receiver pairs $[s, r]_a$, and the channel output symbols are the *suspected* sender-receiver pairs $[s, r]_s$. The channel transition probability $p([s, r]_s|[s, r]_a)$ (i.e. the probability that $[s, r]_s$ is suspected as communication given that $[s, r]_a$ is the actual communication) describes the result of the anonymity attack.

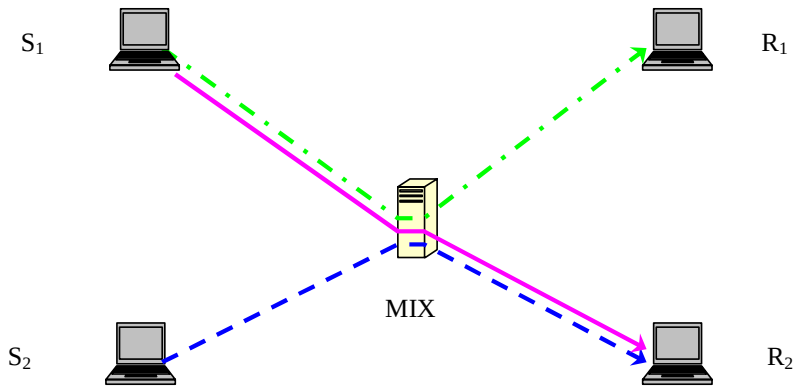


Fig. 50. Single-Mix Scenario

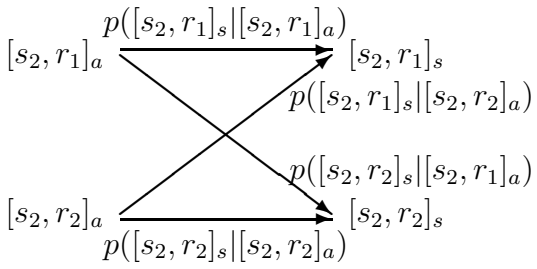


Fig. 51. Anonymity-based Covert Channel Model

We use the simple scenario shown in Figure 50 as an example. We assume that the attacker can collect data at the output ports of the mix as well as some additional information about incoming traffic from the senders. The details on how this information is collected and evaluated depend on the particular attack. See Section 1 for examples. Given sufficient collected data, the attacker can detect individual communications, such as $[s_2, r_2]$, with some non-negligible probability, despite the anonymity preserving count-measures in the mix.

The fact that the attacker is able to gain information about communications indicates that a covert channel of the following form exists: A covert channel sender can send a symbol by establishing a communication from some Sender s_2 to Receiver r_1 and send another symbol by establishing a communication from Sender s_2 to another Receiver, r_2. The covert channel receiver can use the anonymity attack to detect the flow's direction and then make the decision. The channel model is as shown in Figure 51. For sake of simplicity, in this example we limit the covert channel sender to establishing communications from Sender s_2. Allowing communications from Sender s_1 increases the set of input symbols accordingly.

We compute the capacity of the (anonymity-based) covert channel in textbook

fashion by maximizing the mutual information over all input symbol distributions:

$$\begin{aligned} C &= \max_{p([s_2,r]_a)} I([s_2,R]_a;[s_2,R]_s) \\ &= \max_{p([s_2,r]_a)} \sum_{i=1}^{2}\sum_{j=1}^{2} (p([s_2,r_i]_a,[s_2,r_j]_s)\cdot \log \frac{p([s_2,r_j]_a,[s_2,r_i]_s)}{p([s_2,r_i]_a)p([s_2,r_j]_s)}). \end{aligned} \tag{7.5}$$

The covert channels previously proposed in the context of mix networks [38, 39, 40] are not anonymity-based in the sense described above, as the signal is not received across the channel as the result of an anonymity attack. Rather, they describe information leakage in low-level mechanisms that are used to realize mixes, such as batching mechanisms in [38, 40]. These covert channels are then exploited by the anonymity attacks, which in turn can be used to establish the type of anonymity-based covert channels described in this chapter.

D. Single-mix Case

In a mix with a single Sender s_1, a covert-channel sender can establish a covert channel by having s_1 communicate with any combination of j among the n receivers. For this covert channel, the set of input symbols is $\{[s_1,r_k]_a : 1 \le k \le n\}$ and the set of output symbols is $\{[s_u,r_v]_s : 1 \le u \le m,\ 1 \le v \le n\}$. We can include all communications into the set of output symbols because the improbability of any particular communication being declared as suspected by a particular attack can be appropriately reflected by a zero transition probability.

Therefore $\sum_{j=1}^{n} \binom{n}{j}$ different covert channels can be established. Similarly, if the covert channel sender has control over multiple senders, there are at least $\sum_{i=1}^{m} \binom{m}{i} \sum_{j=1}^{n} \binom{n}{j}$ different covert channels that can be established. Which of these $\sum_{i=1}^{m} \binom{m}{i} \sum_{j=1}^{n} \binom{n}{j}$ covert channels has the maximum capacity?

Lemma 3 *For a single sender s_i on a single mix, maximum covert channel capacity*

is achieved when s_i communicates to all receivers.

Proof: By having s_i communicate to all receivers, the covert channel sender can send all the possible symbols $[s_i, r_j]_a$, $1 \leq j \leq n$. We call this covert channel x. Without loss of generality, we assume another covert channel y is established by communicating only to a subset of receivers, r_1, r_2, $\cdots$, r_l, $1 \leq l < n$.

By definition, the capacity of channel x is the maximal mutual information over the distributions $p([s_i, r_1]_a)$, $p([s_i, r_2]_a), \cdots, p([s_i, r_n]_a)$, where $\sum_{j=1}^{n} p([s_i, r_j]_a) = 1$, that is:

$$C_x = \max_{\substack{p([s_i,r_1]_a),p([s_i,r_2]_a),\\ \cdots,p([s_i,r_n]_a)}} I([S,R]_a;[S,R]_s) \quad . \tag{7.6}$$

If Sender s_i does not send to Receiver r_j, the probability $p([s_i, r_j]_a)$ is zero. By constraining some of the probabilities to zero, the maximum value of the capacity does not increase.

$$\begin{aligned} C_x &\geq \max_{\substack{p([s_i,r_1]_a),p([s_i,r_2]_a),\cdots,\\ p([s_i,r_l]_a),\underbrace{0,\cdots,0}_{n-l}}} I([S,R]_a;[S,R]_s) \\ &= \max_{\substack{p([s_i,r_1]_a),p([s_i,r_2]_a),\\ \cdots,p([s_i,r_l]_a)}} I([S,R]_a;[S,R]_s) = C_y \end{aligned}$$

Hence, the capacity of Channel x communicating to all receivers is larger or equal to the capacity of all other covert channels that communicating to only a subset of receivers.

■

Theorem 3 *For a single mix, the maximum covert channel capacity is achieved when the covert channel sender controls all the Senders s_1,s_2, $\cdots$,s_m, and the input symbols of the corresponding channel include all the possible pairs $[s_i, r_j]_a$.*

The proof of Theorem 3 follows the same approach as the proof of Lemma 4.

From Theorem 3, we can get the following corollary.

Corollary 2 *For the single-mix model shown in Figure 49, the maximum covert-channel capacity is*

$C = \max_{p([s,r]_a)} I([S,R]_a; [S,R]_s)$.

From Corollary 2 and Equation (7.3), we get the relationship between the quality of a single mix (i.e. the capacity of any covert channel that allows information to leak from the mix) and the anonymity degree. (Note that this relationship is trivial for the single-mix case. However, we make use of this result in the analysis of networks of mixes.)

Lemma 4 *Given a single mix with a possible maximum information leakage that is upper-bounded by C_{upper}, the anonymity degree of the single mix is lower-bounded by $1 - \frac{C_{upper}}{\log(m \cdot n)}$. Similarly, given that the anonymity degree provided by a single mix is upper-bounded by D_{upper}, the maximum information leakage of the mix is lower-bounded by $(1 - D_{upper}) \log(m \cdot n)$.*

Proof: If the covert channel capacity is upper-bounded by C_{upper},

$$\begin{aligned} D &= 1 - \frac{I([S,R]_a; [S,R]_s)}{\log(m \cdot n)} \\ &\geq 1 - \frac{C}{\log(m \cdot n)} \\ &\geq 1 - \frac{C_{upper}}{\log(m \cdot n)} \quad . \end{aligned}$$

If the anonymity degree is upper-bounded by D_{upper},

$$\begin{aligned} C &= max(I([S,R]_a; [S,R]_s)) \\ &\geq I([S,R]_a; [S,R]_s) \end{aligned}$$

$$
\begin{aligned}
&= (1 - D) \log (m \cdot n) \\
&\geq (1 - D_{upper}) \log (m \cdot n) \quad .
\end{aligned}
$$

■

Lemma 4 describes how the design and implementation quality of a mix affects effectiveness. In the following sections, we will describe this relation for the case of mix networks.

E. Mix Network Case

1. Anonymity Degree of a Mix Network

We generalize the anonymity degree for a single mix defined in Equation (7.3) to the network case by observing that the effectiveness of a mix network can be represented similarly to that of a "super mix". Let R_M and S_M represent the set of senders and receivers of the super mix, respectively. The anonymity degree of the super mix (and of the mix network) is

$$D = 1 - \frac{I([S_M, R_M]_a; [S_M, R_M]_s)}{\log(m \cdot n)} \tag{7.7}$$

where, similarly to the single-mix case,

$$
\begin{aligned}
&I([S_M, R_M]_a; [S_M, R_M]_s) = \\
&\sum_{[s_i, r_j]_a, [s_u, r_v]_s} (p([s_i, r_j]_a, [s_u, r_v]_s) \cdot \log \frac{p([s_u, r_v]_s | [s_i, r_j]_a)}{p([s_u, r_v]_s)}).
\end{aligned} \tag{7.8}
$$

$I([S_M, R_M]_a; [S_M, R_M]_s)$ is determined by $p([s_i, r_j]_a)$ and $p([s_u, r_v]_s | [s_i, r_j]_a)$, where probability $p([s_i, r_j]_a)$ is the proportion of traffic between s_i and r_j, and the probability $p([s_u, r_v]_s | [s_i, r_j]_a)$ is determined by the results of the anonymity attack at one or more mixes in the mix network. In the following sections, we describe how to make

use of the single-mix attack result to describe the effectiveness of a mix network.

2. Effectiveness of Single-Mix vs. Super Mix

In the following, we use the term $p_h([s_u, r_v]_s|[s_i, r_j]_a)$ to represent the transition probabilities that are the result of some anonymity attack on Mix M_h, and $p([s_u, r_v]_s|[s_i, r_j]_a)$ to represent the end-to-end transition probability for the super mix. Without loss of generality, we assume in the following that the super mix transition probability we are interested in is $p([s_u, r_v]_s|[s_i, r_j]_a)$. The process to determine the relationship between $p_h([s_u, r_v]_s|[s_i, r_j]_a)$ and $p([s_u, r_v]_s|[s_i, r_j]_a)$ can be divided into three steps.

a. Step 1

Find the set P_{uv} of all the possible paths between s_u and r_v. Clearly

$$p([s_u, r_v]_s|[s_i, r_j]_a) = \sum_{P_a \in P_{uv}} p([s_u, r_v]_{s,P_a}|[s_i, r_j]_a) \tag{7.9}$$

where $p([s_u, r_v]_{s,P_a}|[s_i, r_j]_a)$ denotes the probability of suspecting communication $[s_i, r_j]_a$ to be communication $[s_u, r_v]_s$ *over Path* P_a. Note that the actual communication between s_i and r_j takes only one path, which we call Path P_0.

b. Step 2

Determine the probability of suspecting an actual communication over Path P_0 to be the communication over another path P_a. Depending on how Path P_a and Path P_0 overlap, we distinguish three situations: (i) There is only one segment where the two paths overlap. (ii) The two paths share multiple segments. (iii) There is no overlap between the two paths. Since there is no overlap in Situation (iii), the probability of suspecting a communication over path P_0 to be the communication over path P_a is

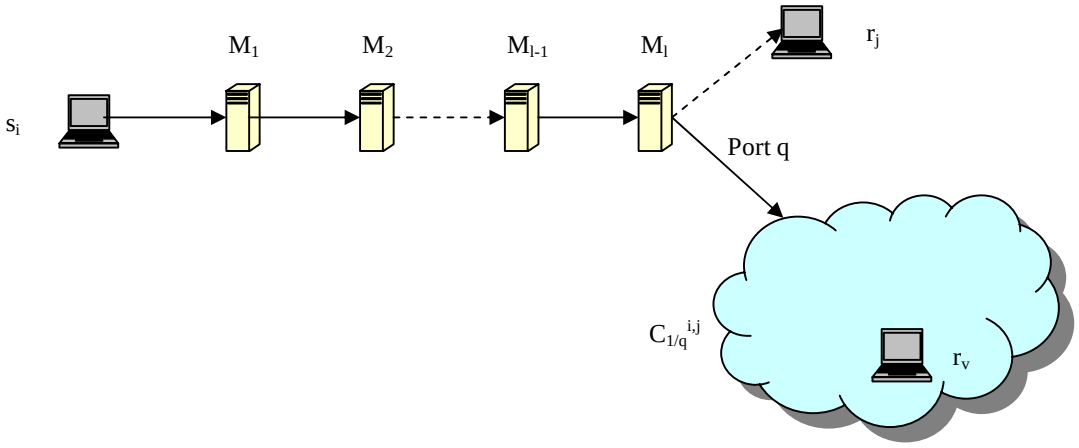


Fig. 52. Case (2)

zero. Hence, we only need to further pursue Situation (i) and Situation (ii).

Situation (i) can be divided into four sub-cases:

Case (1): P_0 and P_a are identical. This implies that $s_u = s_i$ and $r_v = r_j$. In this case, the probability of suspecting correctly is the product of the probabilities of locally suspecting correctly at all mixes along Path P_0. If we denote the mixes on Path P_0 to be M_1, M_2, $\cdots$, M_l, then

$$p([s_i, r_j]_{s,P_0}|[s_i, r_j]) = p_1([s_i, M_2]_s|[s_i, M_2]_a)$$
$$\cdot(\prod_{d=2}^{l-1} p_d([M_{d-1}, M_{d+1}]_s|[M_{d-1}, M_{d+1}]_a)) \cdot p_l([M_{l-1}, r_j]_s|[M_{l-1}, r_j]_a). \quad (7.10)$$

This follows directly from the fact that correct guesses at each mix on the path cause the attacker to correctly suspect the actual path.

Case (2): P_0 and P_a share the same path from s_i through the first Mix M_1 to some Mix M_l, and then diverge due to an error at Mix M_l. This is illustrated in Figure 52 where, in order to emphasize the path P_0 and P_a, other possible connections among the mixes and other possible mixes are ignored. The fact that P_0 and P_a share the same path from s_i means that s_i is correctly suspected, i.e. $s_u = s_i$.

In this subcase, the probability of erroneously suspecting some receiver r_v other than r_j is the result of correctly identifying the path up to some Mix M_{l-1}, and then making a mistake at Mix M_l. Once an error has been made, the remaining mixes on the path to any erroneously suspected Receiver r_v are not on Path P_0. According to the attack model described in Section B, no differentiation can be made between r_v and any other receiver that can be reached after making an error at Mix M_l. We therefore aggregate all receivers that can be reached after an error at Mix M_l into what we call a *cloud* of receivers. We denote by $C_{l/q}^{ij}$ the cloud that is a result of an error at Mix M_l, where communication $[s_i, r_j]_a$ is incorrectly identified because Port q was erroneously selected instead of the port taken by $[s_i, r_j]_a$. For the example in Figure 52, the probability of suspecting receiver to be inside Cloud $C_{l/q}^{ij}$ is

$$p([s_i, C_{l/q}^{ij}]_s|[s_i, r_j]) = p_1([s_i, M_2]_s|[s_i, M_2]_a) \cdot (\prod_{d=2}^{l-1} p_d([M_{d-1}, M_{d+1}]_s|[M_{d-1}, M_{d+1}]_a)) \cdot p_l([M_{l-1}, C_{l/q}^{ij}]_s|[M_{l-1}, r_j]_a). \quad (7.11)$$

Since we are only interested in *receivers* in the cloud, we call $C_{l/q}^{ij}$ a *receiver cloud* in this case. Whenever the context requires, we distinguish between sender clouds and receiver clouds, denoted SC and RC, respectively. We aggregate receiver into clouds because, without additional evidence about the actual flow, it is impossible to differentiate suspects in a cloud by assigning different probabilities. More sophisticated anonymity attacks may make it possible to better differentiate receivers and senders in local attacks on mixes. In such a case we would modify our detector model and extend Equation (7.11) accordingly. In some cases, a cloud can consist of a single receiver or sender.

The dashed line between Mix M_l and Receiver r_j in Figure 52 is to emphasize that the existence of intermediate mixes after M_l will not further contribute to suspecting

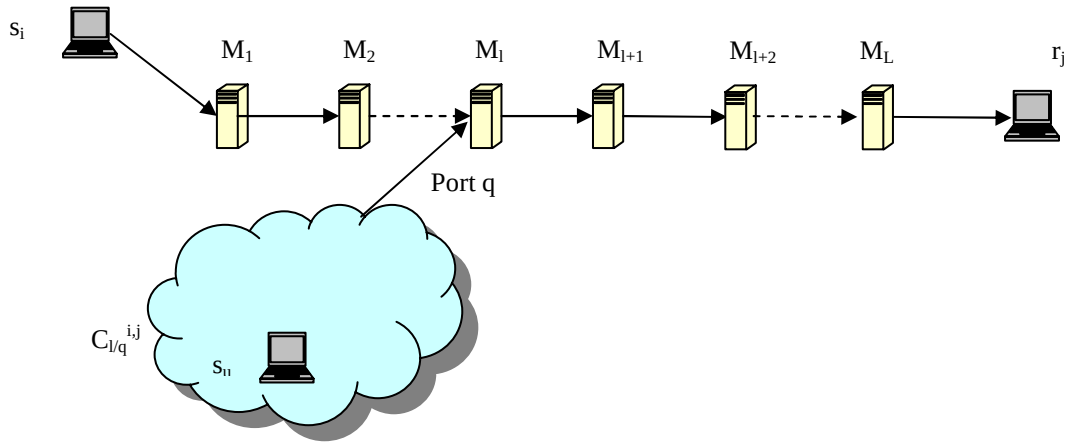


Fig. 53. Case (3)

communication $[s_i, r_j]_a$ as communication $[s_i, C^{ij}_{l/q}]_s$.

Case (3): P_0 and P_a share the same path from some Mix M_l to the receiver. Similarly to Case (2), we introduce a *sender cloud* $C^{ij}_{l/q}$, which is connected to the (input) Port q of Mix M_l. Since the anonymity attacks from Mix M_1 to Mix M_{l-1} may make wrong decision to suspect communication $[s_i, r_j]_a$ as communications from senders attached to the Mixes M_1 to M_{l-1}, the probability of suspecting communication $[s_i, r_j]_a$ as communications from senders attached to the Mixes after M_{l-1} will be $p_1([s_i, M_2]_s|[s_i, M_2]_a)$
$\cdot(\prod_{d=2}^{l-1} p_d([M_{d-1}, M_{d+1}]_s|[M_{d-1}, M_{d+1}]_a))$. Then a wrong guess at Mix M_l and correct guesses till the end of path will result in the suspected communication $[SC^{ij}_{l/q}, r_j]_s$. For the situation in Figure 53, the probability of suspecting communication $[C^{ij}_{l/q}, r_j]_s$ is

$$\begin{aligned} p([C^{ij}_{l/q}, r_j]_s|[s_i, r_j]) &= p_1([s_i, M_2]_s|[s_i, M_2]_a) \\ &\cdot(\prod_{d=2}^{l-1} p_d([M_{d-1}, M_{d+1}]_s|[M_{d-1}, M_{d+1}]_a)) \\ &\cdot p_l([C^{ij}_{l/q}, M_{l+1}]_s|[M_{l-1}, M_{l+1}]_a) \end{aligned}$$

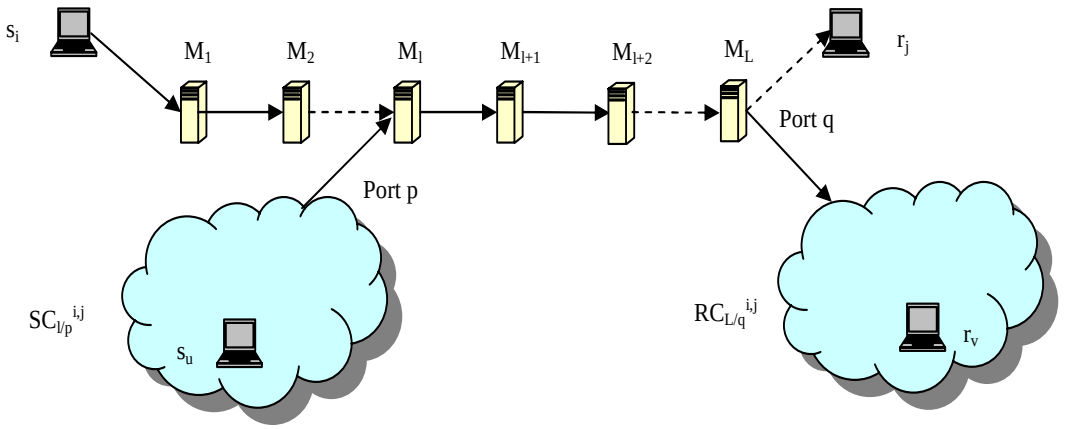


Fig. 54. Case (4)

$$\cdot(\prod_{d=l+1}^{L-1} p_d([M_{d-1}, M_{d+1}]_s|[M_{d-1}, M_{d+1}]_a))$$
$$\cdot p_L([M_{L-1}, r_j]_s|[M_{L-1}, r_j]_a). \tag{7.12}$$

Case (4): P_0 and P_a only share their path in middle of each path, as shown in Figure 54.

In this case, we combine Case (2) and Case (3) as follows:

$$p([SC_{l/p}^{ij}, RC_{L/q}^{ij}]_s|[s_i, r_j]) = p_1([s_i, M_2]_s|[s_i, M_2]_a)$$
$$\cdot(\prod_{d=2}^{l-1} p_d([M_{d-1}, M_{d+1}]_s|[M_{d-1}, M_{d+1}]_a))$$
$$\cdot p_l([SC_{l/p}^{ij}, M_{l+1}]_s|[M_{l-1}, M_{l+1}]_a)$$
$$\cdot(\prod_{d=l+1}^{L-1} p_d([M_{d-1}, M_{d+1}]_s|[M_{d-1}, M_{d+1}]_a))$$
$$\cdot p_L([M_{L-1}, RC_{L/q}^{ij}]_s|[M_{L-1}, r_j]_a), \tag{7.13}$$

We point out that Case (1), Case (2), and Case (3) can all be regarded as special cases of Case (4). In Case (1), both sender cloud and receiver cloud have only one sender and one receiver respectively. In Case (2), the sender cloud has only one sender, while in Case (3) the receiver cloud has only one receiver.

Situation (ii) can have two or more overlaps between path P_0 and P_a. However, the attacker loses the ability to infer anything about communication $[s_i, r_j]_a$ after the first mistake, where the two paths split. All the nodes reachable after the first mistake have to be aggregated in a receiver cloud. This situation is therefore no different than the single-overlap situation described above.

The result of Step 2 is the probability $p([SC_{l/p}^{ij}, RC_{L/q}^{ij}]_s|[s_i, r_j])$ of suspecting communication $[s_i, r_j]_a$ as communication $[SC_{l/p}^{ij}, RC_{L/q}^{ij}]_s$.

c. Step 3

In Step 1 and Step 2 we determined path-dependent end-to-end transition probabilities of the form $p([SC_{l/p}^{ij}, RC_{L/q}^{ij}]_s|[s_i, r_j]_a)$ from the local transition probabilities at the mixes. This allows us to determine the end-to-end transition probabilities of the super-mix (and – as a side result – the anonymity degree of the mix network) by solving the following optimization problem:

Given:

- Local transition probabilities $p_h([\cdot]_s|[\cdot]_a)$ at each mix M_h in the network
- Path-dependent transition probabilities $p([SC_{l/p}^{ij}, RC_{L/q}^{ij}]_s|[s_i, r_j]_a)$.
- Traffic volume in form of *a priori* probability $p([s_i, r_j]_a)$.

Objective Function: Minimize the Anonymity Degree D in Equation (7.3). This is equivalent to maximizing the mutual information $I([S, R]_a; [S, R]_s)$ in Equation (7.2).

Constraints: The optimization problem is subject to the following three sets of constraints:

[Constraint Set 1:] The sum of all path-independent transition probabilities to all the end nodes in a group of clouds is identical to the sum of path-dependent end-to-end transition probabilities to the clouds in the group. For simplicity of notation, we

formulate this for the special case of a correctly suspected Sender s_i. The extension to the general case is cumbersome, but straightforward. Let $GR_v^{i,j}$ be the smallest set of receiver clouds that contain r_v and all receivers in $GR_v^{i,j}$.

$$\forall r_v : \sum_{r_w \in GR_v^{i,j}} p([s_i, r_w]_s | [s_i, r_j]_a) = \sum_{RC_{l/q}^{i,j} \in GR_v^{i,j}} p([s_i, RC_{l/q}^{i,j}]_{s,P_b} | [s_i, r_j]_a). \quad (7.14)$$

[Constraint Set 2:] The sum of all path-independent transition probabilities to a sub-group of receivers is larger than the sum of the path-dependent end-to-end transition probabilities to the clouds which only contain the receivers in the sub-group. It is true because one receiver in the sub-group may be contained in another cloud which contains the receivers not in the sub-group. Let R_{sub} be a subset of the set R of all receivers. Define $H_{R_{sub}}^{i,j}$ to be the set of all clouds that contain *only* receivers in R_{sub}. For the simple case of a correctly suspected Sender s_i:

$$\forall R_{sub} : \sum_{r_v \in R_{sub}} p([s_i, r_v]_s | [s_i, r_j]_a) \geq \sum_{RC_{l/q}^{i,j} \in H_{R_{sub}}^{i,j}} p([s_i, RC_{l/q}^{i,j}]_{s,P_b} | [s_i, r_j]_a). \quad (7.15)$$

[Constraint Set 3:] The sum of all path-independent transition probabilities to a sub-group of receivers is less than the sum of the path-dependent end-to-end transition probabilities to the clouds which have at least one receiver in the sub-group. It is true because these clouds may have other receivers which are not in the sub-group. Let R_{sub} be a subset of the set R of all receivers. Define $I_{R_{sub}}^{i,j}$ to be the set of all clouds that contains *at least one* of the receivers in R_{sub}. We can conclude:

$$\forall R_{sub} : \sum_{r_v \in R_{sub}} p([s_i, r_v]_s | [s_i, r_j]_a) \leq \sum_{RC_{l/q}^{i,j} \in I_{R_{sub}}^{i,j}} p([s_i, RC_{l/q}^{i,j}]_{s,P_b} | [s_i, r_j]_a). \quad (7.16)$$

[Constraint set 4:] The end-to-end transition probabilities for all suspects for all

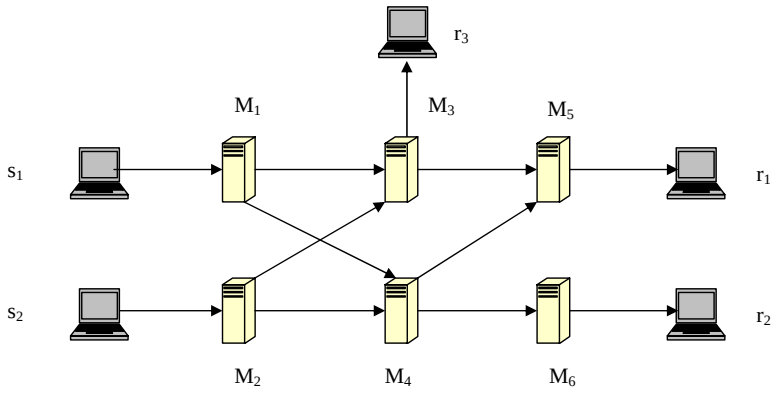


Fig. 55. A Small Example

actual communications sum up to 1:

$$\forall i, j \sum_{s_u, r_v} p([s_u, r_v]_s | [s_i, r_j]_a) = 1 \quad . \tag{7.17}$$

The solution of this optimization problem is the set of the end-to-end transition probabilities of the super mix that minimize the anonymity degree of the mix network.

3. A Small Example

We use the example mix network displayed in Figure 55 to illustrate how to compute end-to-end transition probabilities as described in Step 2 of Section 2.

We focus on communication $[s_1, r_1]$. Suppose the actual communication takes the route P_0: $s_1 \rightarrow M_1 \rightarrow M_3 \rightarrow M_5 \rightarrow r_1$. In this case, the probability of (erroneously) suspecting communications $[s_1, r_3]$ is computed as follows:

$$p([s_1, r_3]_s | [s_1, r_1]_a) = p_1([s_1, M_3]_s | [s_1, M_3]_a) \cdot p_3([M_1, r_3]_s | [M_1, M_3]_a) \quad . \tag{7.18}$$

This computation is simple, since there is only one path from s_1 to r_3.

The situation of (correctly) suspecting communication $[s_1, r_1]_a$ is more complicated, because two paths can be taken. One is $P_0 : s_1 \rightarrow M_1 \rightarrow M_3 \rightarrow M_5 \rightarrow r_1$, the

other is $P_1 : s_1 \rightarrow M_1 \rightarrow M_4 \rightarrow M_5 \rightarrow r_1$. Clearly, we have

$$p([s_1, r_1]_{s,P_0}|[s_1, r_1]) = p_1([s_1, M_3]_s|[s_1, M_3]_a)$$
$$\cdot p_3([M_1, M_5]_s|[M_1, M_5]_a) \cdot p_5([M_3, r_1]_s|[M_3, M_1]_a) \tag{7.19}$$

of suspecting $[s_1, r_1]$ over Path P_0.

For path P_1, we can not get express $p([s_1, r_1]_{s,P_1}|[s_1, r_1]_a)$ directly in terms of anonymity attack result at mixes, because the wrong guess at Mix M_1 will possibly lead to two receivers, r_1 and r_2. So we have to aggregate Receiver r_1 and r_2 in receiver cloud $C_{1/q}^{1,1}$, where q denotes the wrongly selected output port at Mix M_1. So what we can get is

$$p([s_1, C_{1/q}^{1,1}]_s|[s_1, r_1]_a) = p_1([s_1, M_4]_s|[s_1, M_1]_a) \quad , \tag{7.20}$$

where the erroneous selection of Port q on Mix M_1 leads to the suspected Path $s_1 \rightarrow M_1 \rightarrow M_4$. Clearly both Receiver r_1 and Receiver r_2 can be reached after selecting Port q on Mix M_1.

In turn, by following Equation (7.14), we can get

$$p([s_1, r_1]_s|[s_1, r_1]_a) + p([s_1, r_2]_s|[s_1, r_1]_a) =$$
$$p_1([s_1, M_4]_s|[s_1, M_1]_a) + p_1([s_1, M_3]_s|[s_1, M_3]_a)$$
$$\cdot p_3([M_1, M_5]_s|[M_1, M_5]_a) \cdot p_5([M_3, r_1]_s|[M_3, M_1]_a) \quad . \tag{7.21}$$

After repeating this for all possible sender-receiver pairs, expressions for the end-to-end transition can be formulated, and the optimization described in Step 3 of Section 2 can be used to determine the anonymity degree of the network.

F. Covert Channel Capacity *vs.* Anonymity Degree in Mix Networks

The analysis of the effectiveness of anonymity networks is rendered difficult for two reasons, among others: First, attacks on such networks are typically out-of-the-box attacks (for example none of the intersection attacks, trickle attacks, or others target measures taken by the mix network). Second, it is unknown where and how traffic information is collected. Is the attack targeting individual mixes or clusters of mixes? Is the information collected on a per-mix or a per-link basis?

In this section we describe how the anonymity in mix networks can be systematically analyzed and bounded based on estimates of either per-mix weakness (using local covert channels) or the entire mix network (using network-wide covert channels). For this purpose, we investigate the relation between the covert channel capacity of a mix network and the anonymity provided by the network.

1. Upper Bound on the Covert Channel Capacity in Mix Networks

Let the mix network have K mixes. For Mix M_h, we use S_h and R_h to represent the set of senders and receivers of Mix M_h respectively. Any anonymity attack on Mix M_h will lead to a set of probabilities of the form $p_h([s_u, r_v]_s|[s_i, r_j]_a)$ with s_u and s_i in S_h and r_v and r_j in R_h.

In a mix network, there are various ways to establish covert channels. For example, in the mix network shown in Figure 56, there are at least two ways to establish the covert channels using the two mixes M_A and M_B. One way is to establish one covert channel on M_A and M_B separately. Alternatively, one can establish a covert channel on the super mix containing both M_A and M_B. We assume each mix can only be contained in one covert channel as before. In the following, we use the notation $cc(\mathcal{M})$ to denote the covert channel that can be established over the set of the

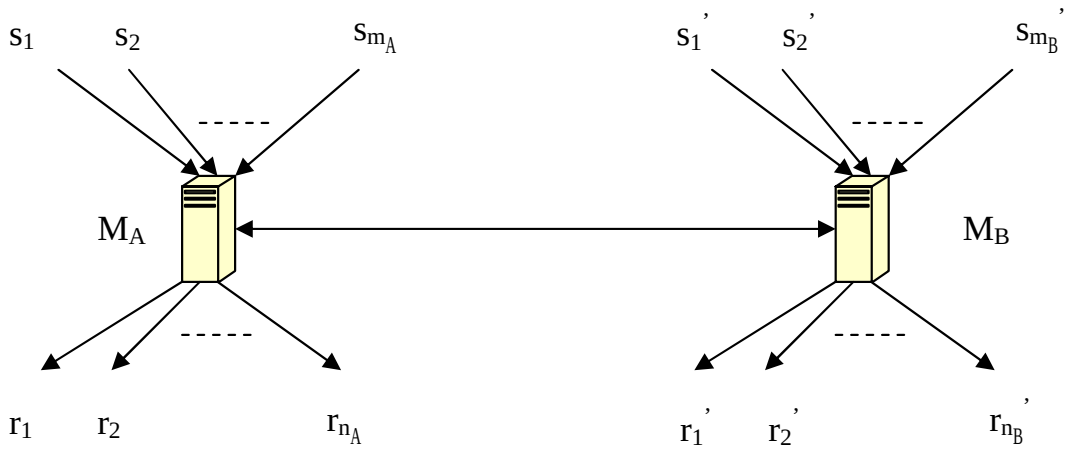


Fig. 56. Mix Network of Two Mixes

mixes $\mathcal{M}$. If we denote the capacities of $cc(\{M_A\})$ and $cc(\{M_B\})$ to be C_A and C_B, respectively, then the sum of the covert channel capacity clearly is $C_A + C_B$. We have the following lemma:

Lemma 5 *The capacity of* $cc(\{M_A, M_B\})$ *will be no greater than* $C_A + C_B$.

Proof: The input and output alphabet of $cc(\{M_A\})$ are $\{[s,r]_a : s \in S_A, r \in R_A\}$ where $S_A = \{s_1, s_2, \cdots, s_{m_A}, M_B\}$ and $R_A = \{r_1, r_2, \cdots, r_{n_A}, M_B\}$. Please note that Mix M_B can be both a sender and a receiver for Mix M_A and *vice versa*. We can construct a new channel v_1 from $cc(\{M_A\})$ with reduced set of input symbols. The input symbols of Channel v_1 are $\{[s,r]_a : s \in S_A - \{M_B\}, r \in R_A\} \bigcup \{[M_B, M_B]_a\}$. According to Theorem 3, the capacity of $cc(\{M_A\})$ will be no less than the capacity of Channel v_1.

Now we consider a covert channel v_2. The covert channel sender of channel v_2 controls all the senders $s \in S_A - \{M_B\}$ attached to Mix M_A to communicate with any Receiver r attached to both mixes, $r \in R_A \bigcup R_B - \{M_A, M_B\}$, where $R_B = \{r'_1, r'_2, \cdots, r'_{n_B}, M_A\}$. Let I_2 to denote the set $\{[s,r]_a : s \in S_A - \{M_B\}, r \in R_A \bigcup R_B - \{M_A, M_B\}\}$. Assuming the covert channel sender can also send the symbol $[M_B, M_B]_a$,

the input symbols of v_2 are $I_2 \bigcup \{[M_B, M_B]_a\}$. The receiver of the covert channel v_2 can only observe all the links connected to Mix M_A. So the channel output symbols are $\{[s, r]_s : s \in S_A, r \in R_A\}$. The transition probability for Channel v_2 is fully determined by the anonymity attack on Mix M_A. For example, for input symbol $[s_1, r_1]_a$ and output symbol $[s_1, r_1]_s$, the transition probability is $p_{M_A}([s_1, r_1]_s|[s_1, r_1]_a)$. Please note:

$$p([s_1, r_x]_s|[s_1, r'_i]_a) = p([s_1, r_x]_s|[s_1, r'_j]_a) = p_{M_A}([s_1, r_x]_s|[s_1, M_B]_a), \tag{7.22}$$

where $r_x \in R_A$, $r'_i \in R_B$ and $r'_j \in R_B$.

We can observe that because of Equation (7.22), we can get channel v_1 by aggregating Channel v_2's input symbols $[s_x, r'_1]_a, [s_x, r'_2]_a, \cdots, [s_x, r'_{n_B}]_a$ ($s_x \in S_A - \{M_B\}$) into $[s_x, M_B]_a$. It is obvious that

$$\sum_{i=1}^{n_B} p([s_x, r'_i]_a) = p([s_x, M_B]_a) \quad . \tag{7.23}$$

The mutual information $I(X;Y)$ is a concave function of $p(x)$ for fixed $p(y|x)$. From *Jensen's Inequality* [102], we can infer that the mutual information between Channel v_1's input and output will be no less than the mutual information between Channel v_2's input and output. So the capacity of Channel v_1, C_{v_1} is no less than the capacity of Channel v_2, C_{v_2}.

Furthermore, we can extend the output symbols of Channel v_2. The extension is as follows: (a) extend $[s_x, M_B]_s$ to $[s_x, r'_1]_a, [s_x, r'_2]_a, \cdots, [s_x, r'_{n_B}]_a$ (b) extend $[M_B, r_y]_s$ to $[s'_1, r_y]_s, [s'_2, r_y]_s, \cdots, [s'_{m_B}, r_y]_s$ (c) extend $[M_B, M_B]_s$ to $\{[s, r] : s \in S_B - \{M_A\}, r \in R_B - \{M_A\}\}$.

Then we can get Channels v_3. Its input symbols are the output symbols of Channel v_2 and its output symbols are the extended output symbols of the Channel v_2. Clearly the transition probabilities of Channel v_3 are determined by the anonymity

attack on Mix M_B. So the Channel v_3's output is determined by the Channel v_3's input and it is independent of Channel v_2's input given Channel v_3's input. So we have the *Markov Chain*: Channel v_2's input $\rightarrow$ Channel v_2's output, i.e. Channel v_3's input $\rightarrow$ Channel v_3's output.

According to *Data Processing Inequality* [102], the mutual information between Channel v_2's input and Channel v_2's output will be no less than the mutual information between Channel v_2's input and Channel v_3's output. We can create a Channel v_4 whose input is Channel v_2's input and whose output is Channel v_3's output. Then we have C_{v_4}, the capacity of Channel v_4 will be no less than C_{v_2}, the capacity of the Channel v_2.

So far we have

$$C_A \quad \geq \quad C_{v_1} \geq C_{v_2} \geq C_{v_4} \tag{7.24}$$

and

$$C_{v_4} = \max_{p([s_i,r_j]_a)=1} \sum_{\substack{[s_i,r_j]_a \in I_2 \\ \bigcup \{[M_B,M_B]\}}} I([s_i,r_j]_a;[s_u,r_v]_s) \tag{7.25}$$

where $s_u \in S_A \bigcup S_B - \{M_A, M_B\}$, $r_v \in R_A \bigcup R_B - \{M_A, M_B\}$, and
$S_B = \{s'_1, s'_2, \cdots, s'_{m_B}, M_A\}$.

Clearly the output symbols of Channel v_4 is the same as the output symbols of Channel $cc(\{M_A, M_B\})$ which is built on the super mix. The input symbols of Channel v_4 contains a part of the input symbols of Channel $cc(\{M_A, M_B\})$ and $[M_B, M_B]_a$.

Similarly, we can get

$$C_B \geq \max_{p([s_i,r_j]_a)=1} \sum_{\substack{[s_i,r_j]_a \in I'_2 \\ \bigcup \{[M_A,M_A]\}}} I([s_i,r_j]_a;[s_u,r_v]_s) \tag{7.26}$$

where $I'_2 = \{[s,r]_a : s \in S_B - M_A, r \in R_A \bigcup R_B - \{M_A, M_B\}\}$ and

$S_B = \{s'_1, s'_2, \cdots, s'_{m_B}, M_A\}$. The other part of the input symbols $cc(\{M_A, M_B\})$ are included in I'_2.

The capacity of Channel $cc(\{M_A, M_B\})$ is

$$C_s = \max_{p([s_l,r_j]_a)=1} \sum_{[s_l,r_j]_a \in I_2 \bigcup I'_2} I([s_i, r_j]_a; [s_u, r_v]_s) \tag{7.27}$$

$$\leq \max_{p([s_i,r_j]_a)=1} \sum_{[s_i,r_j]_a \in I_2 \bigcup \{[M_B,M_B]_a\}} I([s_i, r_j]_a; [s_u, r_v]_s)$$

$$+ \max_{p([s'_i,r_j]_a)=1} \sum_{[s'_i,r_j]_a \in I'_2 \bigcup \{[M_A,M_A]_a\}} I([s'_i, r_j]_a; [s_u, r_v]_s) \tag{7.28}$$

$$\leq C_A + C_B. \tag{7.29}$$

It is true from step 7.27 to step 7.28 because of two reasons. First, the maximization range $\sum_{[s_i,r_j]_a \in I_2 \bigcup \{[M_B,M_B]\}} p([s_i, r_j]_a) = 1$, $\sum_{[s'_i,r_j]_a \in I'_2 \bigcup \{[M_A,M_A]\}} p([s'_i, r_j]_a) = 1$ includes the maximization range $\sum_{[s_l,r_j]_a \in I_2 \bigcup I'_2} p([s_l, r_j]_a) = 1$. Second, according to *Log Sum Inequality* [102],

$$\sum_{[s_i,r_j]_s \in O_2} (p([M_B, M_B]_a, [s_i, r_j]_s) \cdot \log \frac{p([M_B, M_B]_a, [s_i, r_j]_s)}{p([M_B, M_B]_a) p([s_i, r_j]_s)})$$

$$\geq \sum_{[s_i,r_j]_s \in O_2} (p([M_B, M_B]_a, [s_i, r_j]_s) \cdot \log \frac{\sum_{[s_i,r_j]_s \in O_2} p([M_B, M_B]_a, [s_i, r_j]_s)}{\sum_{[s_i,r_j]_s \in O_2} p([M_B, M_B]_a) p([s_i, r_j]_s)}$$

$$= 0$$

where O_2 is the set of output symbols of Channel v_4 and $cc(\{M_A, M_B\})$. Adding non-negative terms will not change the direction of the inequality. From step 7.28 to step 7.29, Inequalities (7.26), (7.24) and Equation (7.25) are used. ■

Extending the two mixes case in Lemma 5, we can get the following Lemma.

Lemma 6 *For two mixes connected with more than one links, the capacity of the*

covert channel built on the super mix, $cc(\{M_A, M_B\})$ will be no greater than $C_A + C_B$.

The proof is similar to that of Lemma 5. Instead of only one path between M_A and M_B, there are more than one paths between M_A and M_B. But it will not affect the use of the inequalities employed in the proof of Lemma 5.

Theorem 4 *In a mix network of K mixes, the sum of the capacities of all the covert channels in the mix network will be no greater than $\sum_{h=1}^{K} C_h$.*

Proof: This theorem can be proved by induction on K mixes with the help of Lemma 6, as any set $K+1$ mixes can be partitioned into a supermix of K mixes and a single mix.

2. Relationship

Similarly to the single-mix case in Section D, we are interested in how bounds on the achievable anonymity degree are affected by the covert channel capacity of the system, and *vice versa*. For example, it is obvious that an upper bound on the anonymity degree will result in a lower bound on the total covert-channel capacity, following the observation that anonymity attacks are more effective in less anonymous mixes.

The upper bound D_{upper} on the anonymity gives raise to a lower bound C_{lower} on the sum of the local channel capacities:

$$C_{lower} = min(\sum_{n=1}^{K} C_h) \tag{7.30}$$

Equation (7.30) gives raise to a minimization problem over anonymity attack results $p_h([s_u, r_v]_s | [s_i, r_j]_a)$, with the following three constrains: First, the local *a priori* probabilities for communications at each Mix M_h must sum to one:

$$\sum_{i=1}^{m_h}\sum_{j=1}^{n_h} p_h([s_i, r_j]_a) = 1. \tag{7.31}$$

Second, the transition probability from each input symbol $[s_i, r_j]_a$ of each mix should sum up to one:

$$\sum_{u=1}^{m_h}\sum_{v=1}^{n_h} p_h([s_u, r_v]_s|[s_i, r_j]_a) = 1, \tag{7.32}$$

Third, the anonymity of the system, as computed in Section 1, should not exceed D_{upper}.

We can solve this constrained optimization problem analytically by using Lagrange multipliers and Kuhn-Tucker conditions. Or we can use numerical methods such as Monte-Carlo.

Similarly, given upper bound C_{upper} on the total covert channel capacity of the mix network, we would like to find out a lower bound D_{lower} for anonymity degree of the mix network.

The objective function becomes

$$D_{lower} = \min[1 - \frac{I([S_M, R_M]_a; [S_M, R_M]_s)}{\log(m \cdot n)}] \tag{7.33}$$

This optimization problem is over all possible anonymity attack result $p_h([s_u, r_v]_s|[s_i, r_j]_a)$. Constraints (7.31) and (7.32) still in this case. The new constraint is

$$C_{upper} \geq \sum_{h=1}^{K} C_h \tag{7.34}$$

G. Evaluation

We use the mix network shown in Figure (57) as an example to illustrate the relationships established in the previous section. We choose six mixes because it is not

a trivial topology, and both a mix cascade and a stratified network case [103] can be established on the six mixes.

We assume that communications between each sender-receiver pair have the same *a priori* probability (alternatively, the same share of total traffic volume). Since there are two senders and two receivers, we have four sender-receiver pairs. The actual path for Communication $[s_i, r_j]_a$ is shown in Table III if the actual path is not specified and the path is possible in the topology. We assume the anonymity attack at each mix is useful, meaning the attack can identify the actual local communication $[s_i, r_j]_a$ with a probability equal or larger than random guess. For our examples, we use adaptive simulated annealing to solve the optimization problem to establish D_{lower} from a known bound on the mix network capacity.

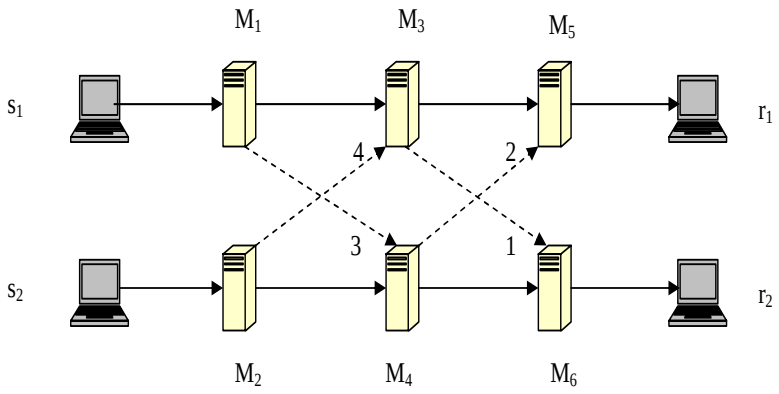


Fig. 57. An Example Mix Network

1. Impact of the Connectivity

Obviously the connectivity will affect the anonymity degree in a mix network. In our first set of examples, the base topology contains only the solid lines in Figure 57. Then edges are incrementally added to the base topology in the order of the label

Table III. Path of the Actual Communications

Communication	Actual Path
$[s1, r1]_a$	$s_1 \rightarrow M_1 \rightarrow M_3 \rightarrow M_5 \rightarrow r_1$
$[s1, r2]_a$	$s_1 \rightarrow M_1 \rightarrow M_3 \rightarrow M_6 \rightarrow r_2$
$[s2, r1]_a$	$s_2 \rightarrow M_2 \rightarrow M_4 \rightarrow M_5 \rightarrow r_1$
$[s2, r2]_a$	$s_2 \rightarrow M_2 \rightarrow M_4 \rightarrow M_6 \rightarrow r_2$

assigned to each edge. The average degree of the topologies including base topology are $2, \frac{14}{6}, \frac{16}{6}, 3, \frac{20}{6}$ respectively.

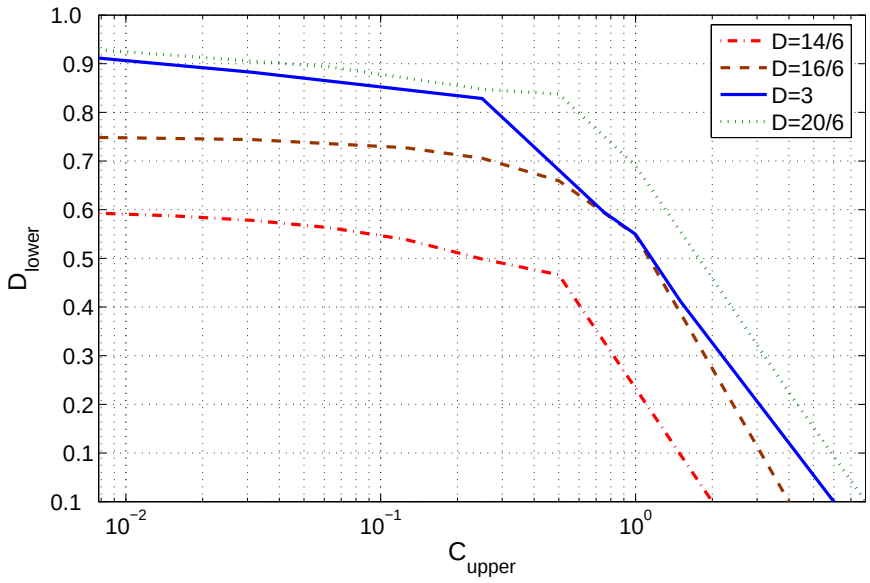


Fig. 58. Impact of the Connectivity

For every mix in the base topology, there is only one input link and one output link. So there is only one sender receiver pair for the mix in the base topology. A channel which has only one input symbol and one output symbol will have capacity zero. So the capacity C_{sum} is zero for base topology.

From Figure 58, first we can observe that the lower bound of the anonymity degree decreases with increasing bound on the capacity, just as we expect. In addition, the capacity C_{sum} increases with increasing connectivity. For a given upper bound of the capacity C_{sum}, increasing connectivity will increase the anonymity degree. Third, we can observe that there is large gap between the base topology and the topology of the next higher average degree. This is because adding the edge of label 1 will connect s_1 and r_2 and the Communication $[s_1, r_2]_a$ can be suspected as $[s_1, r_1]_s$. So the initial edge added to the topology can increase the anonymity degree significantly. In comparison, the effect of adding edge with Label 4 is marginal.

2. Effect of Adding Different Edges

In the second set of examples, we use the solid lines and edge with label 1 as base topology. Then we add one more edge 2, 3 or 4 to the base topology. We label the new topology as A, B and C respectively. Clearly these topologies are of the same average degree. From Figure 59, we can observe that the anonymity degree increase cause by adding edge with label 3 is smaller than adding the other two edges. This is because adding the other two edges can make Communication $[s_2, r_1]_a$ possible and the Communication $[s_2, r_1]_a$ can be suspected as other communications.

3. Effect of Path Selection

In this set of examples, we focus on the topology containing all the solid and dashed lines except the edge with label 3. We consider two cases. In one case, the actual path for Communication $[s_2, r_1]_a$ follows Path A as in Table III. In the other case, the actual path B for Communication $[s_2, r_1]_a$ is $s_1 \rightarrow M_2 \rightarrow M_3 \rightarrow M_5 \rightarrow r_1$.

We can observe going though Mix M_3 will slightly increase the anonymity from Figure 60. This is because Mix M_3 has more output and input links than the other

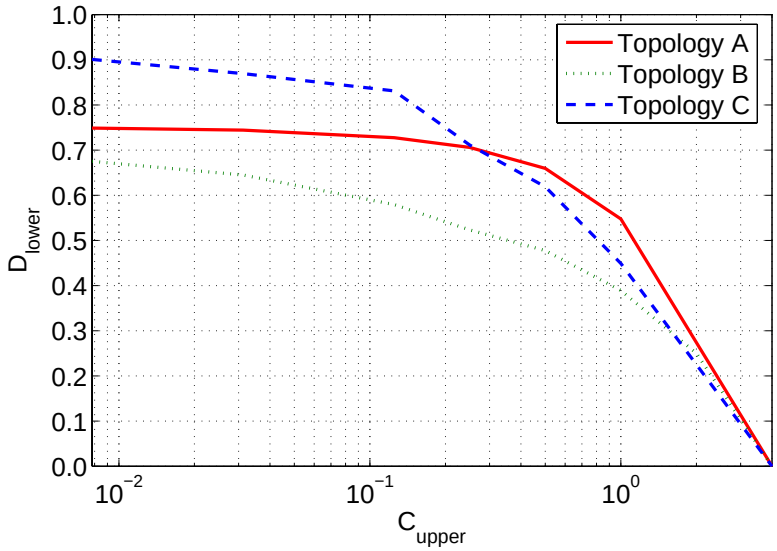


Fig. 59. Effect of Adding Different Edges

mixes. So the communication through Mix M_3 is more easy to hide.

H. Summary

In this chapter we propose a new mutual information based anonymity degree. It gives out one number which is between zero and one to indicate the overall effectiveness of a whole mix network. We also gives out a proof on how to achieve maximal covert channel capacity for a single mix based on anonymity attacks on the mix. The relationship between the anonymity degree and anonymity attack based covert channel capacity is derived for both a single mix case and mix network case.

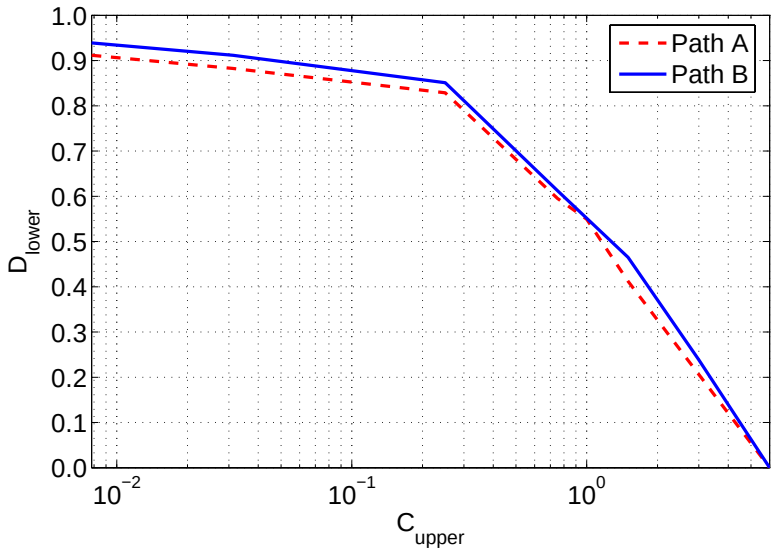


Fig. 60. Effect of Path Selection

CHAPTER VIII

CONCLUSION AND FUTURE WORK

In this dissertation, we investigated if the current anonymous communication system can achieve anonymity. We proposed several traffic analysis methods to analyze the effectiveness of current anonymity system.

Traffic analysis based on flow correlation, presented in Chapter III, can determine the dependence between individual traffic flows and aggregate traffic flows. We proposed to measure the dependency in the time domain using mutual information and frequency domain using cross-correlation. We theoretically analyze the effectiveness of flow correlation attack for different mix strategies. We formally prove that given sufficient length of trace data, flow correlation attack can achieve detection rate arbitrary close to 100%. We derive the formulae to reveal the relationship between effectiveness of flow correlation attack and parameters of different mix strategies. To counter the passive flow correlation attack, we proposed adaptive output traffic control which can be immune to flow correlation attack and achieve high throughput for payload traffic.

Traffic analysis based on flow separation, presented in Chapter V, can separate individual traffic flows by using Blind Source Separation.In a mix network, the mixture of traffic flows can be observed on the input and output links of a mix. By applying blind source separation algorithms to maximize the independence between individual traffic flows, the attacker can separate the traffic flows through the mix network. Furthermore, the passive attacker can get the traffic map of the mix network by matching frequency spectrum of flows separated at each mix. In contrast to previous research results, we experimentally and analytically showed that multicast traffic can be in some cases dangerous for anonymity network.

We propose a location estimation method based on packet timing information in Chapter VI. The location estimation method is based on Blind Source Separation. Our experiments show that the proposed method can both precisely and accurately estimate the location of wireless nodes.

In chapter VII, we propose an anonymity degree to evaluate the quality of an anonymity network. In comparison with previously proposed anonymity degrees mostly defined with respect to a message or a packet, our proposed anonymity degree can capture the quality of an anonymity network. Our proposed anonymity degree takes into account both network topology and heterogeneity in deployed anonymity measures and possible attack methods. The proposed anonymity degree is based on mutual information, and it can generalize all the previously proposed information-theoretical measures. The imperfectness of an anonymity network can result in information leakage from the anonymity network in the form of covert channel. We formally derived the relationship between our proposed anonymity degree and maximum capacity of covert channels inside the anonymity network.

Our future work will focus on building an anonymity system. To build an anonymity system, the following aspects need to be addressed: First new techniques to mix the traffic flow and defeat traffic analysis attack can be invented based on statistical signal processing and randomized algorithm. Next the fundamental trade off between anonymity and usability needs to be evaluated. Low-latency anonymity system was proposed recently for interactive traffic flows. But the study on the performance of TCP traffic flow in anonymity system is still blank. Finally our initial research in this area indicates the need for a new MAC protocol in consideration of both efficiency and privacy. I would like to continue the study in this area.

REFERENCES

[1] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Commun. ACM*, vol. 24, no. 2, pp. 84–90, 1981.

[2] D. M. Goldschlag, M. G. Reed, and P. F. Syverson, "Hiding routing information," *Information Hiding*, pp. 137–150, May 1996.

[3] J. Kong, M. Gerla, and X. Hong, "A new set of passive routing attacks in mobile ad hoc networks," in *Proc. of IEEE Military Communications Conference (MILCOM'03)*, Boston, MA, October 2003, pp. 796–801.

[4] C. Díaz, S. Seys, J. Claessens, and B. Preneel, "Towards measuring anonymity," in *Proc. of Privacy Enhancing Technologies Workshop (PET 2002)*, San Francisco, CA, April 2002, pp. 54–68.

[5] A. Serjantov and G. Danezis, "Towards an information theoretic metric for anonymity," in *Proc. of Privacy Enhancing Technologies Workshop (PET 2002)*, San Francisco, CA, April 2002, pp. 41–53.

[6] D. Chaum, A. Fiat, and M. Naor, "Untraceable electronic cash," in *Proc. on Advances in Cryptology*, Santa Barbara, CA, 1990, pp. 319–327.

[7] H. Wang and Y. Zhang, "Untraceable off-line electronic cash flow in e-commerce," in *Proc. of the 24th Australasian Conference on Computer Science*, Queensland, Australia, 2001, pp. 191–198.

[8] J. Helsingius, "Press release: Johan Helsingius closes his internet remailer," Available: http://www.penet.fi/press-english.html [Accessed June 2001].

[9] S. Parekh, "Prospects for remailers - where is anonymity heading on the internet," Available: http://www.firstmonday.dk/issues/issue2/remailers/ [Accessed June 2001].

[10] C. Gülcü and G. Tsudik, "Mixing email with Babel," in *Proc. of the Network and Distributed Security Symposium - NDSS '96*, San Diego, CA, February 1996, pp. 2–16.

[11] U. Möller, L. Cottrell, P. Palfrader, and L. Sassaman, "Mixmaster protocol — version 2," Available: http://www.eskimo.com/ rowdenw/crypt/Mix/draft-moeller-mixmaster2-proto%col-00.txt [Accessed June 2000], July 2003.

[12] A. Serjantov, R. Dingledine, and P. Syverson, "From a trickle to a flood: Active attacks on several mix types," in *Proc. of Information Hiding Workshop (IH 2002)*, Noordwijkerhout, The Netherlands, October 2002, pp. 36–52.

[13] G. Danezis, R. Dingledine, and N. Mathewson, "Mixminion: Design of a type iii anonymous remailer protocol," in *Proc. of the 2003 IEEE Symposium on Security and Privacy*, Oakland, CA, May 2003, pp. 2–15.

[14] A. Back, U. Möller, and A. Stiglic, "Traffic analysis attacks and trade-offs in anonymity providing systems," in *Proc. of Information Hiding Workshop (IH 2001)*, Pittsburgh, PA, April 2001, pp. 245–257.

[15] O. Berthold and H. Langos, "Dummy traffic against long term intersection attacks," in *Proc. of Privacy Enhancing Technologies Workshop (PET 2002)*, San Francisco, CA, April 2002, pp. 110–128.

[16] O. Berthold, A. Pfitzmann, and R. Standtke, "The disadvantages of free MIX routes and how to overcome them," in *Proc. of Designing Privacy Enhancing*

Technologies: Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, July 2000, pp. 30–45.

[17] M. Mitomo and K. Kurosawa, "Attack for flash mix," in *Proc. of ASIACRYPT 2000*, Kyoto, Japan, December 2000, pp. 192–204.

[18] J.-F. Raymond, "Traffic analysis: Protocols, attacks, design issues, and open problems," in *Proc. of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, Berkeley, CA, July 2000, pp. 10–29.

[19] P. Boucher, A. Shostack, and I. Goldberg, "Freedom systems 2.0 architecture," Zero Knowledge Systems, Inc.," White Paper, December 2000.

[20] M. K. Reiter and A. D. Rubin, "Crowds: Anonymity for web transactions," *ACM Trans. Inf. Syst. Secur.*, vol. 1, no. 1, pp. 66–92, 1998.

[21] M. J. Freedman and R. Morris, "Tarzan: A peer-to-peer anonymizing network layer," in *Proc. of the 9th ACM conference on Computer and communications security*, Washington, DC, 2002, pp. 193–206.

[22] R. Sherwood, B. Bhattacharjee, and A. Srinivasan, "p^5: A protocol for scalable anonymous communication," in *Proc. of the 2002 IEEE Symposium on Security and Privacy*, Oakland, CA, May 2002, pp. 58–70.

[23] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proc. of the 13th USENIX Security Symposium*, San Diego, CA, August 2004, pp. 303–320.

[24] The Free Haven Project Team, "Tor: An anonymous internet communication system," Available: http://tor.eff.org/ [Accessed May 2006].

[25] J. R. Cuellar, J. John B. Morris, D. K. Mulligan, J. Peterson, and J. M. Polk, "RFC 3693: Geopriv requirements," Available: http://www.ietf.org/rfc/rfc3693.txt [Accessed July 2005].

[26] J. Kong and X. Hong, "ANODR: Anonymous on demand routing with untraceable routes for mobile ad-hoc networks," in *Proc. of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC-03)*, Annapolis, MD, June 2003, pp. 291–302.

[27] K. El-Khatib, L. Korba, R. Song, and G. Yee, "Secure dynamic distributed routing algorithm for ad hoc wireless networks," in *Proc. of ICPP Workshops*, Los Alamitos, CA, 2003, pp. 359–366.

[28] B. Zhu, Z. Wan, M. S. Kankanhalli, F. Bao, and R. H. Deng, "Anonymous secure routing in mobile ad-hoc networks," in *Proc. of the 29th Annual IEEE International Conference on Local Computer Networks (LCN'04)*, Tampa, FL, 2004, pp. 102–108.

[29] M. Gruteser and D. Grunwald, "Enhancing location privacy in wireless lan through disposable interface identifiers: A quantitative analysis," in *Proc. of the 1st ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots*, San Diego, CA, September 2003, pp. 46–55.

[30] D. Chaum, "The dining cryptographers problem: Unconditional sender and recipient untraceability," *Journal of Cryptology*, vol. 1, pp. 65–75, 1988.

[31] Q. Sun, D. R. Simon, Y.-M. Wang, W. Russell, V. N. Padmanabhan, and L. Qiu, "Statistical identification of encrypted web browsing traffic," in *Proc. of the 2002 IEEE Symposium on Security and Privacy*, Oakland, CA, 2002, pp. 19–30.

[32] A. Hintz, "Fingerprinting websites using traffic analysis," in *Proc. of Privacy Enhancing Technologies workshop (PET 2002)*, San Francisco, CA, April 2002, pp. 171–178.

[33] A. Serjantov and P. Sewell, "Passive attack analysis for connection-based anonymity systems," in *Proc. of ESORICS 2003*, Gjøvik, Norway, October 2003, pp. 116–131.

[34] B. N. Levine, M. K. Reiter, C. Wang, and M. K. Wright, "Timing attacks in low-latency mix-based systems," in *Proc. of Financial Cryptography (FC '04)*, Key West, FL, February 2004, pp. 251–265.

[35] G. Danezis, "The traffic analysis of continuous-time mixes," in *Proc. of Privacy Enhancing Technologies Workshop (PET 2004)*, Toronto, Canada, May 2004, pp. 35–50.

[36] D. Kesdogan, J. Egner, and R. Büschkes, "Stop-and-go MIXes: Providing probabilistic anonymity in an open system," in *Proc. of Information Hiding Workshop (IH 1998)*, Portland, OR, April 1998, pp. 83–98.

[37] Y. Guan, X. Fu, R. Bettati, and W. Zhao, "An optimal strategy for anonymous communication protocols," in *Proc. of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)*, Vienna, Austria, July 2002, pp. 257–266.

[38] I. S. Moskowitz, R. E. Newman, D. P. Crepeau, and A. R. Miller, "Covert channels and anonymizing networks," in *Proc. of the Workshop on Privacy in the Electronic Society (WPES 2003)*, Washington, DC, October 2003, pp. 79–88.

[39] R. E. Newman, V. R. Nalla, and I. S. Moskowitz, "Anonymity and covert channels in simple timed mix-firewalls," in *Proc. of Privacy Enhancing Technologies Workshop (PET 2004)*, Toronto, Canada, May 2004, pp. 1–16.

[40] I. S. Moskowitz, R. E. Newman, and P. F. Syverson, "Quasi-anonymous channels," in *Proc. of the IASTED International Conference on Communication, Network, and Information Security*, New York, NY, December 2003, pp. 126–131.

[41] A. M. Ladd, K. E. Bekris, A. Rudys, G. Marceau, L. E. Kavraki, and D. S. Wallach, "Robotics-based location sensing using wireless Ethernet," in *Proc. of the Eighth ACM International Conference on Mobile Computing and Networking (MOBICOM)*, Atlanta, GA, September 2002, pp. 227–238.

[42] P. Bahl and V. N. Padmanabhan, "Radar: An in-building rf-based user location and tracking system," in *INFOCOM*, Tel Aviv, Israel, March 2000, pp. 775–784.

[43] D. Niculescu and B. Nath, "Vor base stations for indoor 802.11 positioning," in *Proc. of the 10th Annual International Conference on Mobile Computing and Networking*, Philadelphia, PA, September 2004, pp. 58–69.

[44] D. Niculescu and B. R. Badrinath, "Ad hoc positioning system (aps) using aoa," in *INFOCOM*, San Francisco, CA, March 2003, pp. 1734–1743.

[45] R. J. Ismail Guvenc, Chaouki T. Abdallah and O. Dedeoglu, "Enhancements to rss based indoor tracking systems using kalman filters," in *GSPx & International Signal Processing Conference*, Dallas, TX, April 2003. [Online]. Available: http://www.ece.unm.edu/controls/papers/Guv_CTA_Jor_Ded.pdf

[46] K. Pahlavan, X. Li, and J.-P. Mäkelä, "Indoor geolocation science and technology," *Communications Magazine, IEEE*, vol. 40, no. 2, pp. 112–118, 2002.

[47] E. Elnahrawy, X. Li, and R. P. Martin, "The limits of localization using signal strength: A comparative study," in *IEEE SECON 2004*, Santa Clara, California, 2004, pp. 406–414.

[48] J. Cai, H. You, B. Lu, U. Pooch, and L. Mi, "Whisper - a lightweight anonymous communication mechanism in wireless ad-hoc networks," in *Proc. of Internation Conference on Wireless Networks (ICWN 05)*, Las Vegas, NV, June 2005, pp. 482–488.

[49] C. Tang and C. S. Raghavendra, "Compression techniques for wireless sensor networks," *Wireless Sensor Networks*, pp. 207–231, 2004.

[50] D. X. Song, D. Wagner, and X. Tian, "Timing analysis of keystrokes and timing attacks on ssh," in *Proc. of the 10th USENIX Security Symposium*, Washington, DC, 2001, pp. 337–352.

[51] X. Fu, B. Graham, R. Bettati, W. Zhao, and D. Xuan, "Analytical and empirical analysis of countermeasures to traffic analysis attacks." in *32nd International Conference on Parallel Processing (ICPP 2003)*, vol. 00, Kaohsiung, Taiwan, October 2003, pp. 483–492.

[52] J. D. Howard, "An analysis of security incidents on the internet 1989-1995," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, USA, 1998.

[53] F.B.I, "Carnivore diagnostic tool," Available: http://www.fbi.gov/hq/lab/carnivore/carnivore2.htm [Accessed May 2003].

[54] Onion Routing Development Achives, "Link padding and the intersection attack," Available: http://archives.seul.org/or/dev [Accessed June 2002].

[55] Tcpdump.org, "tcpdump," Available: http://www.tcpdump.org/ [Accessed May 2003].

[56] Cisco Inc., "Netflow services solutions guide," Available: http://www.cisco.com/univercd/cc/td/doc/cisintwk/intsolns/netflsol/nfwhite.htm [Accessed May 2003].

[57] R. Moddemeijer, "On estimation of entropy and mutual information of continuous distributions," *Signal Processing*, vol. 16, no. 3, pp. 233–246, 1989.

[58] MathWorks, "Documentation for mathworks products (release 13)," Available: http://www.mathworks.com/access/helpdesk/help/helpdesk.shtml [Accessed May 2003].

[59] TimeSys, "Timesys linux docs," Available: http://www.timesys.com/ [Accessed June 2003].

[60] Netfilter.org, "Netfilter," Available: http://netfilter.samba.org/ [Accessed June 2003].

[61] A. V. Oppenheim, A. S. Willsky, and S. H. Nawab, *Signals and Systems*, 2nd ed. Upper Saddle River, NJ 07458, USA: Prentice-Hall, 1997.

[62] M. Rennhard, S. Rafaeli, L. Mathy, B. Plattner, and D. Hutchison, "Analysis of an anonymity network for web browsing," in *Proc. of the IEEE 7th Intl. Workshop on Enterprise Security (WET ICE 2002)*, Pittsburgh, PA, June 2002, pp. 49–54.

[63] Y. Guan, X. Fu, D. Xuan, P. U. Shenoy, R. Bettati, and W. Zhao, "Netcamo: Camouflaging network traffic for qos-guaranteed mission critical applications." *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, vol. 31, no. 4, pp. 253–265, 2001.

[64] B. W. Silverman, *Density Estimation for Statistics and Data Analysis.* New York: Chapman and Hall, London, 1986.

[65] S. McCanne and S. Floyd, "'The network simulator - ns-2'," Available: http://www.isi.edu/nsnam/ns/ [Accessed May 2004].

[66] V. Paxson and S. Floyd, "Wide area traffic: The failure of poisson modeling," *IEEE/ACM Trans. Netw.*, vol. 3, no. 3, pp. 226–244, 1995.

[67] IEPM group, "Ttcp," Available: http://www-iepm.slac.stanford.edu/monitoring/voip/ttcp.html [Accessed June 2005].

[68] C. Jutten and J. Herault, "Blind separation of sources, part 1: An adaptive algorithm based on neuromimetic architecture," *Signal Process.*, vol. 24, no. 1, pp. 1–10, 1991.

[69] J.-F. Cardoso, "Blind signal separation: Statistical principles," vol. 9, no. 10, pp. 2009–2025, 1998, special issue on blind identification and estimation.

[70] P. Comon, "Independent component analysis, a new concept?" *Signal Process.*, vol. 36, no. 3, pp. 287–314, 1994.

[71] Z. He, L. Yang, J. Liu, Z. Lu, C. He, and Y. Shi, "Blind source separation using clustering-based multivariate density estimation algorithm," *IEEE Trans. on Signal Processing*, vol. 48, no. 2, pp. 575–579, 2000.

[72] A. Hyvärinen, "Fast and robust fixed-point algorithms for independent component analysis," *IEEE Transactions on Neural Networks*, vol. 10, no. 3, pp. 626–634, 1999.

[73] A. Hyvärinen and E. Oja, "A fast fixed-point algorithm for independent component analysis," *Neural Comput.*, vol. 9, no. 7, pp. 1483–1492, 1997.

[74] M. Gaeta and J.-L. Lacoume, "Source separation without prior knowledge: The maximum likelihood solution," in *Proc. of EUSIPCO'90*, Barcelona, Spain, September 1990, pp. 621–624.

[75] D.-T. Pham, P. Garrat, and C. Jutten, "Separation of a mixture of independent sources through a maximum likelihood approach," in *Proc. of EUSIPCO'92*, Brussels, Belgium, August 1992, pp. 771–774.

[76] A. Hyvärinen and M. Inki, "Estimating overcomplete independent component bases for image windows," *J. Math. Imaging Vis.*, vol. 17, no. 2, pp. 139–152, 2002.

[77] A. Hyvärinen, R. Cristescu, and E. Oja, "A fast algorithm for estimating overcomplete ICA bases for image windows," in *Proc. Int. Joint Conf. on Neural Networks*, Washington, DC, 1999, pp. 894–899.

[78] P. Huang, A. Feldmann, and W. Willinger, "A non-instrusive, wavelet-based approach to detecting network performance problems." in *Proc. of Internet Measurement Workshop*, San Francisco, CA, 2001, pp. 213–227.

[79] S. Cruces and A. Cichocki, "Combining blind source extraction with joint approximate diagonalization: Thin algorithms for ICA," in *Proc. of the Fourth*

Symposium on Independent Component Analysis and Blind Signal Separation, Nara, Japan, April 2003, pp. 463–468.

[80] K. Park and W. Willinger, *Self-Similar Network Traffic and Performance Evaluation*. New York, NY, USA: John Wiley & Sons, Inc., 2000.

[81] L. Tong, R. wen Liu, V. C. Soon, and Y.-F. Huang, "Indeterminacy and identifiability of blind identification," *Circuits and Systems, IEEE Transactions on*, vol. 38, no. 5, pp. 499–509, 1991.

[82] L. Molgedey and H. G. Schuster, "Separation of a mixture of independent signals using time delayed correlations," *Physical Review Letters*, vol. 72, no. 23, pp. 3634–3637, June 1994.

[83] P. Castro, P. Chiu, T. Kremenek, and R. R. Muntz, "A probabilistic room location service for wireless networked environments," in *Proc. of the 3rd International Conference on Ubiquitous Computing*, Atlanta, GA, September 2001, pp. 18–34.

[84] A. R. Beresford and F. Stajano, "Location privacy in pervasive computing," *IEEE Pervasive Computing*, vol. 2, no. 1, pp. 46–55, 2003.

[85] B. Hoh and M. Gruteser, "Protecting location privacy through path confusion," in *IEEE/CreateNet Intl. Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm)*, Athens, Greece, 2005, pp. 194–205.

[86] C. Savarese, J. M. Rabaey, and K. Langendoen, "Robust positioning algorithms for distributed ad-hoc wireless sensor networks," in *Proc. of the General Track:*

2002 USENIX Annual Technical Conference, Salt Lake City, UT, June 2002, pp. 317–327.

[87] C. Savarese, J. M. Rabaey, and J. Beutel, "Locationing in distributed ad hoc wireless sensor networks," in *Proc. of 2001 Int'l Conf. Acoustics, Speech, and Signal Processing (ICASSP 2001)*, vol. 4, Salt Lake City, UT, May 2001, pp. 2037–2040.

[88] P. von Rickenbach and R. Wattenhofer, "Gathering correlated data in sensor networks," in *Proc. of the 2004 Joint Workshop on Foundations of Mobile Computing*, Philadelphia, PA, October 2004, pp. 60–66.

[89] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[90] H. F. Kaiser, "The application of electronic-computers to factor-analysis," *Education and Psychological Measurement,*, vol. 20, no. 1, pp. 141–151, 1960.

[91] R. B. Cattell, "The scree test for the number of factors," *Multivariate Behavioral Research*, vol. 1, pp. 245–276, 1966.

[92] H. Akaike, "A new look at the statistical model identification," *IEEE Transactions on Automatic Control*, vol. AC-19, pp. 716–723, 1974.

[93] G. Schwarz, "Estimating the dimension of a model," *The Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[94] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, pp. 465–471, 1978.

[95] T. P. Minka, "Automatic choice of dimensionality for PCA," in *NIPS*, Denver, CO, December 2000, pp. 598–604.

[96] J.-F. Cardoso and A. Souloumiac, "Blind beamforming for non Gaussian signals," *IEE Proceedings-F*, vol. 140, no. 6, pp. 362–370, Dec. 1993.

[97] A. Belouchrani, K. Abed-Meraim, J.-F. Cardoso, and E. Moulines, "A blind source separation technique using second order statistics," *IEEE Transactions on Signal Processing*, vol. 45, no. 2, pp. 434–444, 1997.

[98] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679–698, 1986.

[99] G. Danezis and A. Serjantov, "Statistical disclosure or intersection attacks on anonymity systems," in *Proc. of 6th Information Hiding Workshop (IH 2004)*, Toronto, Canada, May 2004, pp. 293–308.

[100] B. Graham, Y. Zhu, X. Fu, and R. Bettati, "Using covert channels to evaluate the effectiveness of flow confidentiality measures," in *Proc. of the 11th International Conference on Parallel and Distributed Systems (ICPADS'05)*, Fukuoka, Japan, July 2005, pp. 57–63.

[101] I. S. Moskowitz and M. H. Kang, "Covert channels - here to stay?" in *Compass'94: 9th Annual Conference on Computer Assurance*, Gaithersburg, MD, June 1994, pp. 235–243.

[102] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Wiley-Interscience, 1991.

[103] R. Dingledine, V. Shmatikov, and P. Syverson, "Synchronous batching: From cascades to free routes," in *Proc. of Privacy Enhancing Technologies Workshop (PET 2004)*, Toronto, Canada, May 2004, pp. 186–206.

[104] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the internet," *IEEE/ACM Trans. Netw.*, vol. 7, no. 4, pp. 458–472, 1999.

[105] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, "Modeling TCP throughput: A simple model and its empirical validation." in *Proc. of the ACM SIGCOMM '98 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, Vancouver, B.C., Canada, 1998, pp. 303–314.

[106] K. Fall and S. Floyd, "Simulation-based comparisons of Tahoe, Reno and SACK TCP," *SIGCOMM Comput. Commun. Rev.*, vol. 26, no. 3, pp. 5–21, August 1996.

APPENDIX A

MAXIMUM FREQUENCY COMPONENT OF A TCP FLOW

In this appendix we prove Corollary 1 that a TCP flow has a feature frequency component with the maximum power density at $1/RTT$.

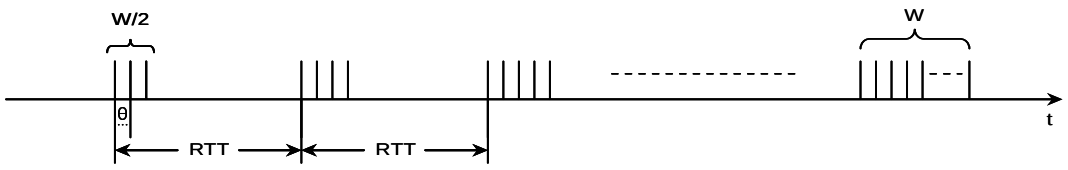


Fig. 61. TCP Congestion Window in Congestion Avoidance Phase

Based on [104, 105], a stable TCP flow's rate changing trend can approximately be illustrated as in Figure 61 if TCP-reno [106] version of congestion control algorithm is used. When a TCP flow is in additive increase phase, one more packet is sent each round trip time. While in multiplicative decrease phase, the packet number in one round trip time decreases by half from W to $\frac{W}{2}$. The inter-departure time θ of two adjacent packets is determined by the smallest bandwidth along the flow path and jitter of queueing delay. Usually, θ is much smaller than RTT.

So we can model the TCP packet train in congestion control phase as

$$x(t) = \sum_{k=0}^{\frac{W}{2}} \sum_{l=0}^{\frac{W}{2}+k-1} \delta(t - l \cdot \theta - k \cdot RTT) \tag{A.1}$$

where $\delta(t)$ is the *unit impulse function.*

Its Fourier transformation is

$$X(\omega) = \sum_{k=0}^{\frac{W}{2}} \sum_{l=0}^{\frac{W}{2}+k-1} e^{-j\omega(k \cdot RTT + l \cdot \theta)} \tag{A.2}$$

.

Its energy-density spectrum is

$$
\begin{aligned}
|X(w)|^2 &= [\sum_{k=0}^{\frac{W}{2}} \sum_{l=0}^{\frac{W}{2}+k-1} \cos(k \cdot RTT \cdot \omega + l \cdot \theta \cdot \omega)]^2 \\
&\quad + [\sum_{k=0}^{\frac{W}{2}} \sum_{l=0}^{\frac{W}{2}+k-1} \sin(k \cdot RTT \cdot \omega + l \cdot \theta \cdot \omega)]^2 \\
&= \sum_{\substack{0 \le k \le \frac{W}{2}, 0 \le l \le \frac{W}{2}+k-1 \\ 0 \le m \le \frac{W}{2}, 0 \le n \le \frac{W}{2}+m-1}} \cos(k \cdot RTT \cdot \omega + l \cdot \theta \cdot \omega) \cos(m \cdot RTT \cdot \omega + n \cdot \theta \cdot \omega) \\
&\quad + \sum_{\substack{0 \le k \le \frac{W}{2}, 0 \le l \le \frac{W}{2}+k-1 \\ 0 \le m \le \frac{W}{2}, 0 \le n \le \frac{W}{2}+m-1}} \sin(k \cdot RTT \cdot \omega + l \cdot \theta \cdot \omega) \sin(m \cdot RTT \cdot \omega + n \cdot \theta \cdot \omega) \\
&= \sum_{\substack{0 \le k \le \frac{W}{2}, 0 \le l \le \frac{W}{2}+k-1 \\ 0 \le m \le \frac{W}{2}, 0 \le n \le \frac{W}{2}+m-1}} \cos(((k-m)RTT + (l-n)\theta)\omega) \qquad \text{(A.3)}
\end{aligned}
$$

Since $\theta \ll RTT$, $|l-n| \le W$ and $(l-n)\theta \to 0$, Equation (A.3) can be approximated as follows,

$$
|X(w)|^2 \approx \sum_{\substack{0 \le k \le \frac{W}{2}, 0 \le l \le \frac{W}{2}+k-1 \\ 0 \le m \le \frac{W}{2}, 0 \le n \le \frac{W}{2}+m-1}} \cos((k-m) \cdot RTT \cdot \omega) \qquad \text{(A.4)}
$$

This corresponds to the case that all packets in one RTT are sent out at roughly the same time at the beginning of the RTT. When $\omega = 2\pi \frac{h}{RTT}$ (h is an integer), we get the maximum $|X(w)|^2$ since $\cos((k-m) \cdot RTT \cdot \omega) = 1$.

Thus, the maximum frequency component of a FTP flow is around frequency $\frac{1}{RTT}$.

APPENDIX B

PROOF OF LEMMAS ON MEAN AND VARIANCE OF MUTUAL INFORMATION ESTIMATION

In this appendix, we provide the major steps for proving Lemma 1 and Lemma 2.

Lemma 1: The mean of mutual information estimation $\hat{I}(f,l)$ is given by

$$E(\hat{I}(f,l)) \approx I(f,l) + \frac{(r-1)(s-1)}{N}$$

where $I(f,l)$ is the original mutual information, and r and s are defined in (4.1) and (4.2) respectively.

Proof:

We estimate mutual information $I(f,l)$ in (4.4) as follows,

$$\begin{aligned}\hat{I}(f,l) &= \sum_{a=0}^{r}\sum_{b=0}^{s}\hat{p}(a,b)\log\frac{\hat{p}(a,b)}{\hat{p}(a)\hat{p}(b)} \\ &= \sum_{a,b}\hat{p}(a,b)\log\hat{p}(a,b) - \sum_{a}\hat{p}(a)\log\hat{p}(a) - \sum_{b}\hat{p}(b)\log\hat{p}(b). \qquad \text{(B.1)}\end{aligned}$$

If we apply a second order Taylor expansion[1] to the three items in (B.1) at $p(a,b)$, $p(a)$, and $p(b)$[2], respectively, after a series of rearrangements, we have

$$\begin{aligned}\hat{I}(f,l) \approx\ & \sum_{a,b}\hat{p}(a,b)\log\frac{p(a,b)}{p(a)p(b)} + \frac{1}{2}\sum_{a,b}\frac{1}{p(a,b)}[\hat{p}(a,b)-p(a,b)]^2 \\ & -\frac{1}{2}\sum_{a}\frac{1}{p(a)}[\hat{p}(a)-p(a)]^2 - \frac{1}{2}\sum_{b}\frac{1}{p(b)}[\hat{p}(b)-p(b)]^2 \qquad \text{(B.2)}\end{aligned}$$

[1]Since the functions to be expanded are smooth functions, it is appropriate to ignore remainder terms. Same is true for Step B.6 in proof of Lemma 2.

[2]The reason for the expansion at the points of the underlying distribution is that the original mutual information is based on the underlying distribution.

Now we are ready to compute the mean of $\hat{I}(f,l)$:

$$E[\hat{I}(f,l)] = \sum_{\substack{n_{0,0},\cdots,n_{r,s} \\ n_{0,0}+\cdots+n_{r,s}=N}} p(n_{0,0},\cdots,n_{r,s})\hat{I}(f,l) \tag{B.3}$$

where $n_{a,b}$ is the the number of occurrences of (a,b). One sample in (4.3) corresponds to a $(n_{0,0},\cdots,n_{r,s})$, which gives one possible mutual information estimation. $p(n_{0,0},\cdots,n_{r,s})$ satisfies a multinomial distribution[3].

Substituting (B.2) into (B.3) and using properties of the multinomial distribution, we have, after rearrangements,

$$\begin{aligned} E[\hat{I}(a,b)] &= \sum_{a,b} p(a,b)\log\frac{p(a,b)}{p(a)p(b)} + \frac{1}{2N}\sum_{a,b}(1-p(a,b)) - \frac{1}{2N}\sum_{a}(1-p(a)) \\ &\quad -\frac{1}{2N}\sum_{b}(1-p(b)) \qquad \text{(B.4)} \\ &= I(f,l) + \frac{rs-1}{2N} - \frac{r-1}{2N} - \frac{s-1}{2N} \\ &= I(f,l) + \frac{(r-1)(s-1)}{2N}. \qquad \text{(B.5)} \end{aligned}$$

Q.E.D.

■

Lemma 2 The variance of the mutual information estimation $\hat{I}(f,l)$ is given by

$$var(\hat{I}(f,l)) \approx \frac{C_{f,l}}{N}$$

[3]In the simplified case where there are only two possible outcomes of (a,b), the distribution will be binomial distribution. For the case where there are more than two outcomes, the distribution will be multinomial.

where $C_{f,l}$ is a constant and is defined as follows

$$C_{f,l} = \sum_{a,b} p(a,b) \left(\log \frac{p(a,b)}{p(a)p(b)} \right)^2 - \left(\sum_{a,b} p(a,b) \log \frac{p(a,b)}{p(a)p(b)} \right)^2$$

where $p(a,b)$ is the original probability distribution of (a,b).

Proof:

To obtain the variance of $\hat{I}(f,l)$, we perform an approximation by only keeping the first item in the Taylor expansion (B.2). Thus,

$$\hat{I}(f,l) \approx \sum_{a,b} \hat{p}(a,b) \log \frac{p(a,b)}{p(a)p(b)} \tag{B.6}$$

According to the definition, we know

$$\hat{p}(a,b) = \frac{n_{a,b}}{N} \tag{B.7}$$

Substituting (B.7) into (B.6), we have

$$\hat{I}(f,l) = \frac{1}{N} \sum_{a,b} n_{a,b} \log \frac{p(a,b)}{p(a)p(b)} \tag{B.8}$$

The multinomial distribution has the following property

$$\sum_{a,b} s_{a,b} n_{a,b} = N \left(\sum_{a,b} p(a,b) s_{a,b}^2 - (\sum_{a,b} p(a,b) s_{a,b})^2 \right) \tag{B.9}$$

where $s_{a,b}$ is a constant. Applying this property to (B.8) with

$$s_{a,b} = \log \frac{p(a,b)}{p(a)p(b)},$$

we have

$$\begin{aligned} Var[\hat{I}(a,b)] &\approx \frac{1}{N^2} Var \left[\sum_{a,b} n_{a,b} \log \frac{p(a,b)}{p(a)p(b)} \right] \\ &= \frac{1}{N} \sum_{a,b} p(a,b) \left(\log \frac{p(a,b)}{p(a)p(b)} \right)^2 - \frac{1}{N} \left(\sum_{a,b} p(a,b) \log \frac{p(a,b)}{p(a)p(b)} \right)^2 \end{aligned}$$

$$= \frac{C_{f,l}}{N} \tag{B.10}$$

Q.E.D.

■

APPENDIX C

PROOF OF DETECTION RATE THEOREM

In this appendix, we describe the major steps for proving Theorem 2.

Theorem 1 For a mix with any number of output links, the detection rate, v, is given by

$$v \approx 1 - \sqrt{\frac{C_{f,l_{M\to R_1}}}{N}} \times \int_{-\infty}^{-I(f,l_{M\to R_1})\sqrt{\frac{N}{C_{f,l_{M\to R_1}}}}} N(0,1)dx$$

where N is the sample size, $I(f, l_{M\to R_1})$ is the mutual information of the input flow f and its corresponding output link aggregated flow $l_{M\to R_1}$, $N(0,1)$ is the density function of the standard normal distribution, and $C_{f,l_{M\to R_1}}$ is defined in (4.8).

Proof:

We know that $\hat{I}(f,l)$ satisfies a normal distribution. Its mean and variance can be derived from Lemma 1 and Lemma 2, respectively. Thus, the detection rate can be obtained by (4.5).

Now, let us examine the distribution of the mutual information estimation. The mutual information estimation $\hat{I}(f, l_{M\to R_1})$ between the input flow f and its corresponding output link aggregated flow $l_{M\to R_1}$ has the following normal distribution:

$$\hat{I}(f, l_{M\to R_1}) \sim N\left(I(f, l_{M\to R_1}) + \frac{(r-1)(s-1)}{N},\ \frac{C_{f,l_{M\to R_1}}}{N}\right). \tag{C.1}$$

Since the input flow goes through $link_{M\to R_1}$, it is easy to see that

$$\begin{aligned} C_{f,l_{M\to R_1}} &= \sum_{a,b} p(a,b)\left(\log\frac{p(a,b)}{p(a)p(b)}\right)^2 - \left(\sum_{a,b} p(a,b)\log\frac{p(a,b)}{p(a)p(b)}\right)^2 \\ &\neq 0, \end{aligned} \tag{C.2}$$

where $p(a,b)$ refers to the joint distribution of input flow f and its corresponding

output link aggregated flow $l_{M\to R_1}$[4].

The mutual information $\hat{I}(f, l_{M\to R_i})$ $(i \neq 1)$ between the input flow f and aggregated flow $l_{M\to R_i}$ has the following normal distribution:

$$\hat{I}(f, l_{M\to R_i}) \sim N\left(I(f, l_{M\to R_i}) + \frac{(r-1)(s-1)}{N},\ \frac{C_{f,l_{M\to R_i}}}{N}\right).$$

where

$$C_{f,l_{M\to R_i}} = \sum_{a,b} p(a,b)(\log\frac{p(a,b)}{p(a)p(b)})^2 - \left(\sum_{a,b} p(a,b)\log\frac{p(a,b)}{p(a)p(b)}\right)^2$$

If we assume that the input flow f is approximately independent of the output link aggregated flow $l_{M\to R_i}$ $(i \neq 1)$, it is easy to see

$$C_{f,l_{M\to R_i}} = 0,$$

and

$$I(f, l_{M\to R_i}) = 0.$$

That is, the mutual information estimation $\hat{I}(f, l_{M\to R_i})$ $(i \neq 1)$ degenerates into a constant $\frac{(r-1)(s-1)}{N}$.

Now, we assume the same size N is sufficiently large and the mix's links have the same bandwidth, the detection rate formula (4.5) becomes

$$\begin{aligned} v &= Pr\left(I(f, l_{M\to R_1}) > \frac{(r-1)(s-1)}{N}, \cdots, I(f, l_{M\to R_1}) > \frac{(r-1)(s-1)}{N}\right) \\ &= Pr\left(I(f, l_{M\to R_1}) > \frac{(r-1)(s-1)}{N}\right) \end{aligned}$$

Since $I(f, l_{M\to R_1})$ has a normal distribution as in (C.1), we can easily obtain

[4]Rigorously, $p(a,b)$ should be $p_{f,l_{M\to R_1}}(a,b)$. The subscript is removed for the sake of brevity.

detection rate v

$$\begin{aligned} v &= \int_{\frac{(r-1)(s-1)}{N}}^{+\infty} N\left(I(f, l_{M\to R_1}) + \frac{(r-1)(s-1)}{N},\ \frac{C_{f,l_{M\to R_1}}}{N}\right) dx \\ &= 1 - \int_{-\infty}^{\frac{(r-1)(s-1)}{N}} N\left(I(f, l_{M\to R_1}) + \frac{(r-1)(s-1)}{N},\ \frac{C_{f,l_{M\to R_1}}}{N}\right) dx \end{aligned} \quad \text{(C.3)}$$

After some transformations, (C.3) becomes

$$v \approx 1 - \sqrt{\frac{C_{f,l_{M\to R_1}}}{N}} \times \int_{-\infty}^{-I(f,l_{M\to R_1})\sqrt{\frac{N}{C_{f,l_{M\to R_1}}}}} N(0,1)dx$$

Q.E.D. ■

APPENDIX D

JOINT DISTRIBUTION DERIVATION FOR SIMPLE MIX

In this appendix, we derive the joint distribution of the input flow packet count a and the output flow packet count b for a simple proxy. Denote Alice's flow as f. There are two cases: (1) an output link carries the flow from Alice to Bob, $l_{M\to R_1}$; (2) an output link *does not* carry the flow from Alice to Bob, $l_{M\to R_i}$, $i > 1$.

Joint Distribution for flows f and $l_{M\to R_1}$

Denote $p(a, b)$ as the joint distribution

$$p(a, b) = p(a)p(b|a) \tag{D.1}$$

Since each packet is padded to the same size in a mix network, the service time for each packet is constant. Because the arrival is Poisson distributed, we can model the simple proxy as a M/D/1 queuing system.

The conditional probability $p(b|a)$ in (D.1) is determined by the queue length Q and noise traffic to Bob. Denote the packet count of noise packets as N_1 and the maximum number of output packets during a sampling interval as C_1. We need to consider two cases:

(1) $b < C_1$: Since the simple proxy is modeled as $M/D/1$ queue, this case means that the output link bandwidth is not fully used and the number of output packets is greater than Alice's packets into the mix during the sampling interval, thus

$$\begin{aligned} p(b|a) &= p(Q + N_1 = b - a) \\ &= \sum_{q=0}^{b-a} p(Q = q)p(N_1 = b - a - q) \end{aligned} \tag{D.2}$$

(2) $b = C_1$: This case means that the output link bandwidth is fully used, thus

$$\begin{aligned} p(C_1|a) &= p(Q + N_1 \geq C_1 - a) \\ &= \sum_{v=C_1-a}^{\infty} \sum_{q=0}^{v} p(Q = q)p(N_1 = v - q) \end{aligned} \tag{D.3}$$

Now we determine the queue length distribution $p(Q = q)$. Denote the noise packet arrival rate as λ_{N_1}, Alice's packet arrival rate as λ_f and output link bandwidth as c_1 ($C_1 = c_1 T$, T is the sampling interval). From basic queuing theory, the equilibrium state queue length distribution of M/D/1 queue is:

$$p(Q = 0) = 1 - \rho \tag{D.4}$$

$$p(Q = 1) = (1 - \rho)(e^{\rho} - 1) \tag{D.5}$$

$$p(Q = q) = (1 - \rho) \sum_{j=1}^{q} (-1)^{q-j} [\frac{(j\rho)^{q-j}}{(q-j)!} + (1 - \delta_{qj}) \frac{(j\rho)^{q-j-1}}{(q-j-1)!}] e^{j\rho}, q \geq 2 \tag{D.6}$$

where

$$\rho = \frac{\lambda_{N_1} + \lambda_f}{c_1}$$

and

$$\delta_{qj} = \begin{cases} 0, & q \neq j \\ 1, & q = j \end{cases}$$

Recalling that noise traffic packet count ($P(N_1)$) and Alice's packet count ($P(a)$) is Poisson distributed, we can get the joint distribution by substituting (F.19), (F.20), (F.21), (D.2) and (D.3) into (D.1).

Joint Distribution for flows f and $l_{M \to R_i}$, $i \geq 2$

Here we assume that Alice's flow f and the mix's output flows into receivers other than Bob are independent, thus Therefore

$$p(a, b) = p(a)p(b) \tag{D.7}$$

Clearly, $p(b)$ can be easily got from the $M/D/1$ queue model if we assume that all traffic is Poisson and the average rate of traffic to receiver R_i is λ_{N_i}. Denoting the maximum number of output packets to R_i as C_i and the corresponding link bandwidth as c_i, we have two cases as above,

(1) $b < C_i$:

$$p(b) = p(N_i + Q = b) = \sum_{q=0}^{b} p(Q = q)p(N_i = b - q) \tag{D.8}$$

(1) $b = C_i$:

$$p(C_i) = p(N_i + Q \geq C_i) = \sum_{v=C_i}^{\infty} \sum_{q=0}^{v} p(Q = q)p(N_i = v - q) \tag{D.9}$$

Noting that the noise traffic is Poisson distributed, the probability of queue length and the joint distribution can be easily got as above.

APPENDIX E

JOINT DISTRIBUTION DERIVATION FOR TIMED MIX

In this appendix, we derive the joint distribution of the input flow packet count a and the output flow packet count b for a timed mix. For a timed mix queue, our model is a little different from that of a simple proxy. In the deduction, we use a sampling interval equal to the period of the timed mix. Thus, packets queued in the current sampling interval will be served by the output link in the next sampling interval. In Figure 62, we can see that the output flow packet count b and the input flow packet count a have a shift of one sampling interval and we denote the queue length Q as the number of packets queued exactly before the output link begins to process the packets. Thus,

$$p(a, b) = p(a)p(b|a) \tag{E.1}$$

$p(a)$ follows a Poisson distribution from the assumption of Poisson traffic. Below we discuss the derivation of $p(b|a)$.

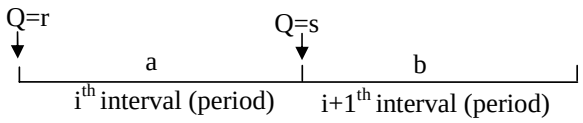


Fig. 62. Timed Mix Queue

Denote Alice's flow as f. There are two cases: (1) an output link carries the flow from Alice to Bob, $l_{M\to R_1}$; (2) an output link does not carry the flow from Alice to Bob, $l_{M\to R_i}$, $i > 1$.

Joint Distribution for flows f and $l_{M\to R_1}$

Similarly the discussion of the simple proxy, we denote C_1 as the maximum number of packets that can be sent out by the output link and L as the maximum queue length. We consider two cases in terms of b.

(1) $b < C_1$:

$$\begin{aligned} p(b|a) &= p(b, Q \leq C_1|a) + p(b, Q > C_1|a) \\ &= \sum_{q=0}^{C_1} p(Q=q)p(b|a, Q=q) + \sum_{q=C_1+1}^{L} p(Q=q)p(b|a, Q=q) \end{aligned}$$

Denoting N_1 as the number of noise packets,

$$\begin{aligned} p(b|a) &= \sum_{q=0}^{C_1} p(Q=q)P(N_1 = b-a) \\ &\quad + \sum_{q=C_1+1}^{L} p(Q=q)p(N_1 = b-a-(q-C_1)) \end{aligned} \tag{E.2}$$

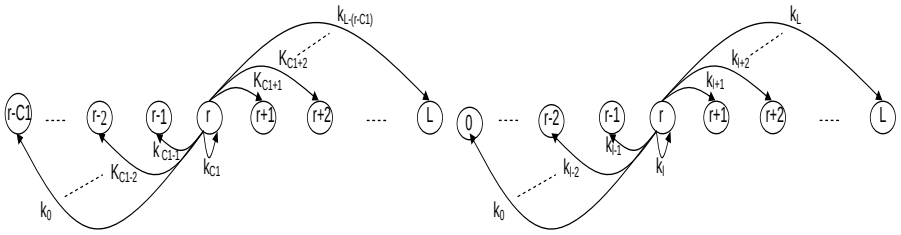
(1) $b = C_1$: For this case, clearly

$$p(C_1|a) = 1 - \sum_{b=0}^{C_1-1} p(b|a) \tag{E.3}$$

In equations (E.2) and (E.3), the terms related with noise traffic is easy to get since noise traffic is Poisson. Now we focus on the derivation of the queue-length probability.

Queue Model

We model the queue using an embedded Markov chain. Denote P_{rs} as the state transition probability matrix, where r is the current queue length exactly before the i^{th} interval) and s the queue length exactly before the $(i+1)^{th}$ interval. We consider two cases, (1) When $r > C_1$, to move the state from $Q = r$ to $Q = s$, there must be $s - (r - C_1)$ packets coming during the i^{th} interval as shown in Figure 62. Then (2) When $r \leq C_1$, to move the state from $Q = r$ to $Q = s$, there must be s incoming



(a) $r > C_1$ (b) $r \leq C_1$

Fig. 63. Embedded Markov Chain

packets during the ith interval. Denoting k_n as the probability that n packets coming in the i^{th} interval, Figure 63 shows the state transition probability from the current state r to the next state s.

Thus, we have the following $(L+1) \times (L+1)$ transmission matrix,

$$[P_{rs}] = \begin{pmatrix} k_0 & k_1 & k_2 & k_3 & k_4 & k_5 & \cdots & k_L \\ k_0 & k_1 & k_2 & k_3 & k_4 & k_5 & \cdots & k_L \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ k_0 & k_1 & k_2 & k_3 & k_4 & k_5 & \cdots & k_L \\ 0 & k_0 & k_1 & k_2 & k_3 & k_4 & \cdots & k_{L-1} \\ 0 & 0 & k_0 & k_1 & k_2 & k_3 & \cdots & k_{L-2} \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & k_0 & k_1 & \cdots & k_{C_1} \end{pmatrix}. \tag{E.4}$$

Thus, we can easily use the following equilibrium equations (E.6) and (E.6) to get the state probability.

$$\vec{\pi} = \vec{\pi}\mathbf{P_{rs}}, \tag{E.5}$$

$$\sum_{i=0}^{L} \pi_i = 1. \tag{E.6}$$

where $\vec{\pi} = (p(Q=0)\ p(Q=1)\ \cdots\ p(Q=L))^T$ is the state probabilities. The final result is

$$\vec{\pi} = (\mathbf{P_m^T} - \mathbf{I_{L+1}} + \begin{pmatrix} 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & 0 \\ 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix})^{-1} \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}, \tag{E.7}$$

where I_{L+1} is a $(L+1) \times (L+1)$ identity matrix and

$$\mathbf{P_m} = \begin{pmatrix} k_0 & k_1 & k_2 & k_3 & k_4 & k_5 & k_6 & \cdots & k_L \\ k_0 & k_1 & k_2 & k_3 & k_4 & k_5 & k_6 & \cdots & k_L \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ k_0 & k_1 & k_2 & k_3 & k_4 & k_5 & k_6 & \cdots & k_L \\ 0 & k_0 & k_1 & k_2 & k_3 & k_4 & k_5 & \cdots & k_{L-1} \\ 0 & 0 & k_0 & k_1 & k_2 & k_3 & k_4 & \cdots & k_{L-2} \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & k_0 & k_1 & k_2 & \cdots & k_{C_1+1} \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}. \tag{E.8}$$

Joint Distribution for flows f and $l_{M \to R_i}$, $i > 1$

We assume that Alice's traffic is independent of noise traffic, thus (E.1) becomes

$$p(a, b) = p(a)p(b)$$

Similar to the discussion above, denoting C_i as the maximum number of packets to the output link to receiver i in one sampling interval and N_i as the corresponding packet count, we have

(1) $b \leq C_i$

$$p(b) = \sum_{q=0}^{C_i} P(Q = q)p(N_i = b) + \sum_{q=C_1+1}^{L} P(Q = q)p(N_i = b + C_i - q) \tag{E.9}$$

where N_i is the noise traffic packet count to Receiver i.

(2) $b = C_i$

$$p(C_i) = 1 - \sum_{b=0}^{C_i-1} p(b) \tag{E.10}$$

The queue length distribution in (E.9) and (E.10) can be derived similarly as in (E.4) and (E.8).

APPENDIX F

JOINT DISTRIBUTION DERIVATION FOR EXPONENTIAL MIX

Overview

The formation of (4.9) in Theorem 2 is generic in terms of traffic and mix characteristics. However, both constant C_i $(1 \leq i \leq M)$ and the original mutual information I_i depend on the joint distribution function $p(u, v_i)$, which in turn depends on the traffic and the mix characteristics. For the case of a continuous-time mix, the effect of the mix on the traffic can be modeled as a two-queue model shown in Figure 64.

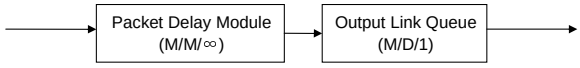


Fig. 64. Model of a Continuous-time Mix

The first queue of the continuous-time mix is introduced by the packet delay module, while the second queue represents the fixed capacity output link of the mix. Based on the definition of continuous-time mix, the delay module can be modeled as a $M/M/\infty$ queue. Since the output traffic of this queue is still a Poisson process and since packets in a mix network are typically padded to a fixed size, the output link queue can be modeled as $M/D/1$ queue.

Based on this model of a continuous-time mix, it is straightforward to derive the joint distribution of (X, Y_i) if we can model the incoming traffic into the mix. In the following, to demonstrate the modeling framework, we assume that the incoming traffic is a Poisson process.

Denote Y_i' as the packet count of the delay module's output flow, and Y_i' is also the packet count of the input flow of the i^{th} output link queuing module where this link may correspond to a possible receiver. Thus $X \rightarrow Y_i' \rightarrow Y_i$ forms a Markov chain. So the joint probability of (X, Y_i) is

$$\begin{aligned} & p(X = u, Y_i = v_i) \\ & \quad = \sum_{v_i'=0}^{\infty} p(X = u, Y_i' = v_i', Y_i = v_i) \qquad \text{(F.1)} \\ & \quad = \sum_{v_i'=0}^{\infty} p(X = u) \\ & \qquad\qquad \times p(Y_i' = v_i' | X = u) \\ & \qquad\qquad \times p(Y_i = v_i | Y_i' = v_i') \qquad \text{(F.2)} \end{aligned}$$

According to our assumption about traffic arrival, the first term $p(X = u)$ in (F.2) follows a Poisson distribution. The second term $p(Y_i' = v_i' | X = u)$ is determined by the packet delay module and the third term $p(Y_i = v_i | Y_i' = v_i')$ is determined by the output link queuing module.

Derivation of $p(Y_i' = v_i' | X = u)$ based on $M/M/\infty$ queuing

Without loss of generality, we assume that Y_1' represents the packet count of flow to receiver Bob. Below we first derive $p(Y_1' = v_1' | X = u)$ for Bob and then $p(Y_i' = v_i' | X = u)$ $(2 \leq i \leq M)$ for receivers other than Bob.

Derivation of $p(Y_1' = v_1' | X = u)$

Three sources of packets contribute to Y_1', the number of packet leaving the packet delay module during the sampling interval: (i) packets left over from the previous sampling interval, denoted as n_q, (ii) Alice's packets arriving in the current sampling

interval, denoted as n_f, and (iii) noise packets arriving during the current sampling interval, denoted as n_z. Thus,

$$p(Y_1' = v_1'|X = u) = \sum_{n_q+n_f+n_z=v_1'} p(N_q = n_q)p(N_f = n_f|X = u)p(N_z = n_z) \quad \text{(F.3)}$$

The derivation of the three terms in (F.3) is as following:

1. $p(N_q = n_q)$

Obviously,

$$p(N_q = n_q) = \sum_{q=n_q}^{\infty} p(Q = q) \cdot \binom{q}{n_q} p_{qdep}^{n_q}(1 - p_{qdep})^{q-n_q}, \quad \text{(F.4)}$$

where p_{qdep} denotes the probability of a packet delayed from a previous interval by delay module of the continuous mix being released during the sample interval, and $p(Q = q)$ denotes the probability of q packets held by the delay module.

Due to the memoryless property of the exponential distribution employed by the delay module, the distribution of remaining delay time after the beginning of a sample interval still follows an exponential distribution with the same parameter. If we assume that the delay module uses an exponential distribution with parameter λ_d,

$$p_{qdep} = \int_0^T \lambda_d e^{-\lambda_d t} dt \quad \text{(F.5)}$$

Since the system can be modeled as M/M/∞ queue, the distribution of queue size Q at the beginning of a sample interval is:

$$p(Q = q) = \frac{r^q e^{-r}}{q!} \quad \text{(F.6)}$$

where $r = \frac{\lambda_f+\lambda_z}{C_1}$, λ_f and λ_z are the Poisson arrival rate for the flow from Alice and noise traffic coming in through the same port. Equation (F.6) holds because of the fact that the flow from Alice is independent of the other traffic through the same port

and the sum of the two Poisson process is also a Poisson process with arrival rate $\lambda_f + \lambda_z$.

So from Equation (F.4), (F.5) and (F.6), we can compute the probability $p(N_q = n_q)$.

2. $p(N_f = n_f | X = u)$

Clearly, when $u < n_f$, the probability $p(N_f = n_f | X = u)$ is zero because the number of packet departures from the flow from Alice in one sampling interval should be no greater than u, the packet arrivals of the flow. There are $\binom{u}{n_f}$ combinations of n_f departures from the u arrivals.

We first label the u incoming packets with sequence number from 1 to u. Suppose the n_f departures contain the packets with sequence number $d_1, d_2, \cdots, d_{n_f}$. We use S_d to denote the set of the sequence number. So $S_d = \{d_1, d_2, \cdots, d_{n_f}\}$.

Since the packet count arrival is Poisson distributed, the probability of exactly u arrivals in a sample interval T is

$$\begin{aligned} P(u) = & \int_{t_1=0}^{T} \lambda_f e^{-\lambda_f t_1} \cdot \int_{t_2=0}^{T-t_1} \lambda_f e^{-\lambda_f t_2} \cdots \\ & \int_{t_u=0}^{T-\sum_{i=1}^{u-1} t_i} \lambda_f e^{-\lambda_f t_u} \cdot \\ & (1 - \int_{t_{u+1}=0}^{T-\sum_{i=1}^{u} t_i} \lambda_f e^{-\lambda_f t_{u+1}} dt_{u+1}) dt_u \cdots dt_1 \end{aligned} \tag{F.7}$$

Let $\Delta_i(t, t_H)$ be defined as follows:

$$\Delta_i(t, t_H) = \begin{cases} \lambda_f e^{-\lambda_f t} \cdot (1 - \int_{t'=0}^{t_H - t} \lambda_d e^{-\lambda_d t'} dt'), & \text{if i} \notin S_d \\ \lambda_f e^{-\lambda_f t} \cdot \int_{t'=0}^{t_H - t} \lambda_d e^{-\lambda_d t'} dt', & \text{if i} \in S_d \end{cases} \tag{F.8}$$

The probability that the n_f packets in S_d are released by the delay module of the continuous-time mix is then

$$p_u(\lambda_f, S_d) = \int_{t1=0}^{T} \Delta_1(t_1, T) \int_{t2=0}^{T-t_1} \Delta_2(t_2, T-t_1) \cdots$$
$$\int_{t_u=0}^{T-\sum_{i=1}^{u-1} t_i} \Delta_u(t_u, T - \sum_{i=1}^{u-1} t_i)$$
$$\cdot(1 - \int_{t_{u+1}=0}^{T-\sum_{i=1}^{u} t_i} \lambda_f e^{-\lambda_f t_{u+1}} dt_{u+1}) dt_u dt_{u-1} \cdots dt_1 \tag{F.9}$$

For example, when $u = 4$ and $S_d = \{2, 4\}$, we can get

$$p_4(\lambda_f, \{2,4\}) = \int_{t_1=0}^{T} \lambda_f e^{-\lambda_f t_1} \cdot (1 - \int_{t'_1=0}^{T-t_1} \lambda_d e^{-\lambda_d t'_1} dt'_1)$$
$$\cdot \int_{t_2=0}^{T-t_1} \lambda_f e^{-\lambda_f t_2} \cdot \int_{t'_2=0}^{T-t_1-t_2} \lambda_d e^{-\lambda_d t'_2} dt'_2$$
$$\cdot \int_{t_3=0}^{T-t_1-t_2} \lambda_f e^{-\lambda_f t_3} \cdot (1 - \int_{t'_3=0}^{T-t_1-t_2-t_3} \lambda_d e^{-\lambda_d t'_3} dt'_3)$$
$$\cdot \int_{t_4=0}^{T-t_1-t_2-t_3} \lambda_f e^{-\lambda_f t_4} \cdot \int_{t'_4=0}^{T-t_1-t_2-t_3-t_4} \lambda_d e^{-\lambda_d t'_4} dt'_4$$
$$\cdot(1 - \int_{t_5=0}^{T-t_1-t_2-t_3-t_4} \lambda_d e^{-\lambda_d t_5} dt_5) dt_4 dt_3 dt_2 dt_1 \tag{F.10}$$

By summing up all the probabilities for the set of the same size, we can get

$$p(N_f = n_f | X = u) = \sum_{|S_d|=n_f} p_u(\lambda_f, S_d) \tag{F.11}$$

3. $p(N_z = n_z)$

The probability $p(N_z = n_z)$ can be calculated in a similar way as the probability

$p(N_f = n_f|X = u)$. For the same port noise traffic, we can get $p_z(\lambda_z, S_d)$ in a similar way deriving Equation (F.9), where λ_z denotes the traffic rate of the same port noise traffic.

Thus we can get

$$p(N_z = n_z) = \sum_{z=n_z}^{\infty} \sum_{|S_d|=n_z} p_z(\lambda_z, S_d) \tag{F.12}$$

B. Derivation of $p(Y_i' = v_i'|X = u)$ *where* $i > 1$

Since Alice's traffic is independent from traffic of receivers other than Bob, easily we have

$$p(Y_i' = v_i'|X = u) = p(Y_i' = v_i') \quad . \tag{F.13}$$

We can derive the probability $p(Y_i' = v)$ in the same way of deriving $p(N_z = n_z)$. We use λ_{Y_i} to denote the average rate of the traffic through Port 2.

$$p(Y_i' = v_i') = \sum_{z=v_i'}^{\infty} \sum_{|S_d|=v_i'} p_z(\lambda_{Y_i'}, S_d) \tag{F.14}$$

Derivation of $p(Y_i' = v_i'|X = u)$ for $i > 1$

Since Alice's traffic is independent from traffic of receivers other than Bob, easily we have

$$p(Y_i' = v_i'|X = u) = p(Y_i' = v_i') \quad . \tag{F.15}$$

We can derive the probability $p(Y_i' = v)$ in the same way of deriving $p(N_z = n_z)$. We use $\lambda_{Y_i'}$ to denote the average rate of the traffic to the possible (link) receiver i $(i > 1)$.

$$p(Y_i' = v_i') = \sum_{z=v_i'}^{\infty} \sum_{|S_d|=v_i'} p_z(\lambda_{Y_i'}, S_d) \tag{F.16}$$

Derivation of $p(Y_i = v_i|Y_i' = v_i')$ based on $M/D/1$ queuing

Similar to the above, we differentiate the case of $p(Y_1 = v_1|Y_1' = v_1')$ and $p(Y_i = v_i|Y_i' = v_i')$ where $i > 1$.

Derivation of $p(Y_1 = v_1|Y_1' = v_1')$

The probability $p(Y_1 = v_1|Y_1' = v_1')$ is determined by the $M/D/1$ queue. We use Q_1 to denote the size of the queue at output Port 1. So the probability $p(Y_1 = v_1|Y_1' = v_1')$ can be expressed as follows:

$$p(Y_1 = v_1|Y_1' = v_1') = p(Q_1 = v_1 - v_1') \tag{F.17}$$

when $v_1 < C_1T$, where in this subsection, C_1 is the bandwidth of the link to Bob. Obviously, when $v_1 < C_1T$, the probability $p(Y_1 = v_1|Y_1' = v_1')$ is zero if $v_1' > v_1$. Because $v_1 < C_1T$ means the link bandwidth is not fully utilized, the queue size will be zero. So all the v_1' incoming packets should depart in the sample interval. When $v_1 = C_1T$, we have

$$\begin{aligned} p(Y_1 = v_1|Y_1' = v_1') &= p(Q_1 > C_1T - v_1') \\ &= \sum_{q=C_1T-v_1'}^{\infty} p(Q_1 = q) \end{aligned} \tag{F.18}$$

According to queuing theory results, the equilibrium state queue length distribution of M/D/1 queue will be:

$$p(Q_1 = 0) = 1 - \rho \tag{F.19}$$

where $\rho = \frac{\lambda_z + \lambda_f}{C_1}$, λ_z is the average rate of noise traffic to Bob and λ_f is the average rate of Alice's traffic to Bob.

$$p(Q_1 = 1) = (1 - \rho)(e^{\rho} - 1) \tag{F.20}$$

$$\begin{aligned} p(Q_1 = q) &= (1 - \rho) \sum_{j=1}^{q} (-1)^{q-j} [\frac{(j\rho)^{q-j}}{(q-j)!} \\ &\quad + (1 - \delta_{qj}) \frac{(j\rho)^{q-j-1}}{(q-j-1)!}] e^{j\rho} \end{aligned} \tag{F.21}$$

where $q \geq 2$ and $\delta_{qj} = \{^{1, n=j}_{0, n \neq j}$.

Derivation of $p(Y_i = v_i | Y_i' = v_i')$, $i > 1$

The probability $p(Y_i = v_i | Y_i' = v_i')$ can be derived in the same way as in Equation (F.17) and (F.18).

VITA

Ye Zhu received his B.E. in electronics engineering in 1994 from Shanghai Jiao Tong University and his M.Sc. in electrical and computer engineering in 2002 from Texas A&M University. He received his Ph.D. degree in electrical and computer Engineering from Texas A&M University in 2006. From 2000 to 2006 he was a research assistant in the NetCamo project at TAMU. His research interests include anonymous communication, network security, peer to peer networking, and traffic engineering. He can be reached at: Texas A&M University, Dept. of Electrical and Computer Engineering, TAMU 3128, College Station, TX 77843-3128.