Towards Securing Interdomain Routing on the Internet

Sharon Goldberg

A Dissertation
Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Doctor of Philosophy

Recommended for Acceptance
by the Department of
Electrical Engineering

Advisors: Jennifer Rexford, Boaz Barak

September, 2009

Abstract

The Internet consists of multiple autonomous systems (ASes), each consisting of networks of devices that are prone to malfunction, misconfiguration, or attack by malicious parties, and each controlled by profit-seeking businesses with different economic goals. Despite these complex relationships, the interdomain routing system (that allows ASes to communicate over the global Internet) currently operates under the assumption that all nodes in the network can trust each other. The thesis contributes to the body of works that seeks to remedy this, by considering network protocols that operate correctly even in the presence of adversarial or selfish behavior.

We take a principled approach to analyze the types of security guarantees that are possible within the engineering and economic constraints of the Internet's interdomain routing system. We focus exclusively on protocols that can be used to improve availability in the Internet, *i.e.,* to increase the likelihood that packets arrive uncorrupted at their correct destination, and analyze two broad themes:

1. Which part of the system should be secured?
2. What is the right tradeoff between security and efficiency?

To address these questions, we consider securing the following two parts of the system: the routing protocols, used to set up paths through the Internet, and the data-plane mechanisms, used to forward packets along the paths set up by the routing protocols.

1. We start with a *game-theoretic* analysis that shows that even the strongest known secure routing protocol is not sufficient to prevent *selfish* ASes from lying about the paths that data packets take through the network. We then find sufficient conditions that ensure that ASes will not lie. Unfortunately, these conditions are highly unrealistic, and so we conclude that ASes may have an incentive to lie about paths, and thus potentially forward their customer's traffic via paths that drop or corrupt packets.
2. We next consider secure data-plane mechanisms. We use novel *cryptographic and data-streaming* approaches to design lightweight protocols that *detect* packet loss and corruption on a path through the network, even when some nodes on the path are *adversarial*. Our protocols allow a sender and receiver to securely monitor billions of packets using only a few hundred bytes of storage and a pair of comparably sized control packets.
3. Finally, we take the security guarantees above even further, by considering protocols that also *localize* an adversarial node that drops or corrupts packets. We use cryptographic proof techniques to design new protocols and argue that *any* secure localization protocol requires the participation of *every* node on the path. This requirement is considered severe in the setting of interdomain routing, where each node is owned by independent economic entity, with little incentive to participate in the localization protocol.

Our results have implications on the design of high-performance network architectures that can withstand selfish and adversarial behavior.

Acknowledgements

I could not have asked for better advisors than Jennifer Rexford and Boaz Barak. I thank them both for giving me complete freedom throughout my PhD. Jen's sharp intellect, excellent taste in problems, and ability to find exactly the right course of action in any situation has been a continual inspiration to me over the years. My career as a researcher has largely been driven by her unwavering support, mentorship and high standards. I thank Boaz for teaching me to think like a theorist, and for always being ready to give me the technical tools and the support I needed to do my research. Despite of his bent for "hardcore" theory, Boaz was always excited about whatever applied problem I happened to throw in his direction, and his ability to work through complex lines of thought in a matter of minutes never fails to amaze me.

I thank Shai Halevi for his unwavering support and the countless hours he spent reading my papers, working with me on proofs, and teaching me to tackle problems in a systematic way. Shai, Tal Rabin, and the rest of the cryptography group at IBM Research (Ran Canetti, Nelly Fazio, Rosario Gennaro, Craig Gentry, Charanjit Jutla, Jonathan Katz, Hugo Krawczyk, and Vinod Vaikuntanathan) have been invaluable mentors to me. I especially thank Tal for her wise advice and for letting me "squat" in the crypto lab at IBM for so many years.

I thank Maria Klawe for supporting my switch in research areas during my second year at Princeton. Maria kept me in grad school, and I cannot thank her enough for giving me the opportunity to work with fantastic researchers in computer science.

This thesis is based on work done jointly with several fantastic computer science researchers: Boaz Barak, Shai Halevi, Jennifer Rexford, Eran Tromer and David Xiao. I especially thank Dave Xiao for hours spent in the library teaching me to think like a cryptographer, and Michael Schapira, with whom I've never coauthored a paper (yet!), for hours spent on the phone teaching me to think like a game theorist.

As part of the Cabernet group, I was fortunate to be in an environment where there was always someone to question my assumptions, challenge my conclusions, or claim that my adversary models were too strong. I especially thank Yi Wang, Changhoon Kim, Haakon Ringberg, Rui Zhang-Shen and Elliot Karpilovsky for their thoughtful comments on many iterations of papers and practice talks. I spent a fantastic summer at Cisco Research in California, where Fabio Manio, Flavio Bonomi, Syam Appala and David McGrew went above and beyond to teach me about the practical side of network security and software engineering. I benefitted from conversations about life and research with Nadia Heninger, Alexandra Kolla, Eugene Brevdo, Alex Fabrikant, Guy Rothblum, Yaron Singer, Carmit Hazay, Alex Halderman, Ari Feldman, Hoeteck Wee, Bin Li and many others. I'd also like to thank various faculty members for sharing their expertise with me, including Robert Calderbank, Moses Charikar, Nick Feamster, Mike Freedman, Claire Gmachl, Piotr Indyk, Li-Shuan Peh, Leo Reyzin, Adam Smith, Dan Wallach and Jo Kelly. I especially thank Tal Malkin and Erich Nahum for being part of my committee.

I'd like to thank Professor Paul Prucnal for giving me my first taste of research during my early years at Princeton. I am also indebted to Andrea Civelli for his unwavering support during those years.

Finally, I'd like to thank various friends scattered in New York, Princeton, Boston, Toronto, San Francisco and Montreal for supporting me, in various ways, over the past five years. I am especially grateful to my family, my parents Sara and Doron, my brother Danny, and my "sister" Lorie, for being there for me through the ups and downs of graduate school. This thesis is dedicated to my Saba, who probably would have earned a PhD himself if he had had the time.

Bibliographical Notes

The material in Chapter 2 and Appendix A is from a paper that was originally coauthored with Shai Halevi [45], then merged with the work of Aaron Jaggard, Vijay Ramachadran and Rebecca Wright [65], and finally published as [46] at SIGCOMM'08.

Chapter 3 and Appendix B expands and clarifies material that originally appeared in a paper [49] that was coauthored with David Xiao, Eran Tromer, Boaz Barak, and Jennifer Rexford and published at SIGMETRICS'08. Sections 3.3, 3.5, and Appendix B present a number of new ideas that did not appear in [49], and Theorem 3.5.2 and Theorem B.5.4 are corrected versions of Theorem 5 and Theorem 6 from [49].

Chapter 4 and Appendix C is a clearer exposition of ideas that appeared in a paper [16] that was coauthored with David Xiao and Boaz Barak and published at EUROCRYPT'08. While the results of Section 4.3.2 were mentioned in [16], they appear in their entirety for the first time as Theorem 4.3.5.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Today's Internet is a collection of *autonomous systems* (ASes) (*e.g.,* Princeton's campus network, AT&T's global backbone network), each controlled by different profit-seeking businesses, each consisting of a complex network of *routers* and other devices. Connectivity on the Internet requires these competing economic entities to cooperate; communication from a source to a destination can traverse multiple devices inside multiple ASes. Despite these complex relationships, the Internet was originally designed under the assumption that devices inside the network could be trusted; security threats were perceived to come from outside the network. Furthermore, the system is notoriously resistant to change; because the Internet is not controlled by single centralized entity, it is extremely difficult to convince multiple independently-operated ASes to upgrade to a new protocol. As such, many protocols used on the Internet today were designed at a time when it still made sense to assume that all devices in the network can trust each other.

Because the Internet functions in a complex economic environment, its operation is challenged by the presence of *adversarial* or *selfish* parties that choose to deviate from correct operation of network protocols. For example, a profit-seeking Internet service provider (ISP) might misrepresent network performance in order to attract more of traffic from its paying customers. As another example, a router hacked by a malicious outsider may selectively modify traffic from a website like cnn.com, perhaps in order to drive up stock prices. Unfortunately, many of the network protocols used on today's Internet were not designed to deal with these types of malicious or strategic behavior. The thesis contributes to the body of works that seeks to remedy this, by considering the *design and analysis of network protocols that operate correctly even in the presence of adversarial or selfish behavior.*

1.1 The interdomain routing system on the Internet

When we purchase an item from Amazon.com, traditional cryptography prevents attackers from seeing our credit card numbers or impersonating the Amazon website. But how can we ensure that our request actually *arrives* at the Amazon.com server, without being dropped or corrupted along the way? This is exactly the challenge we address in this thesis – improving network *availability*, or improving the chances that packets arrive correctly at their destination.

We focus specifically on availability in the *interdomain* routing system, that enables communication between ASes in the global Internet. We separate the interdomain routing system into two parts: the control plane, *i.e.,* the routing protocols used to establish paths through the Internet, and the data plane, *i.e.,* the mechanisms used to forward packets over the paths set up by the routing protocols. Network protocols and devices handle control-plane (routing) and data-plane (forwarding) mechanisms in different ways; data-plane mechanisms are designed

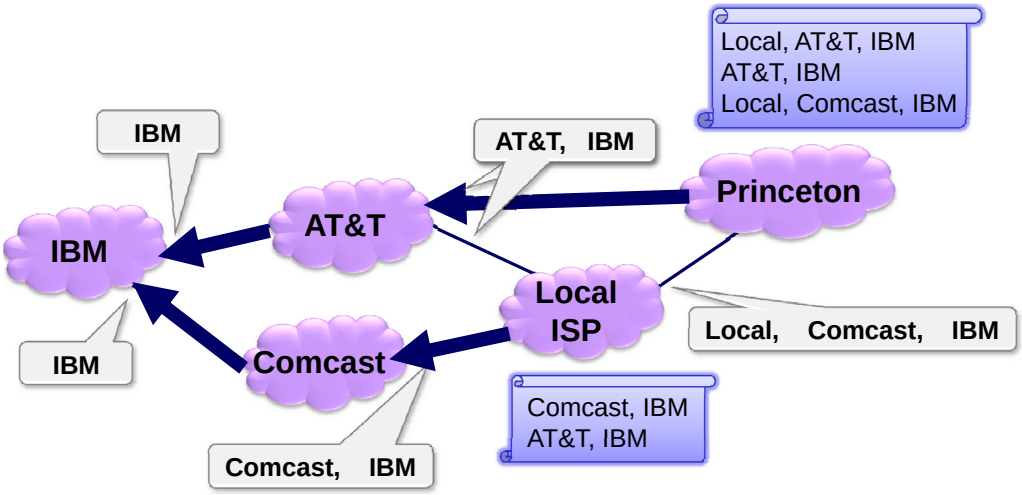


Figure 1.1: A stylized view of the interdomain routing system running BGP.

to be simple and fast, while control-plane mechanisms may be more complex and potentially slower. This separation exists because paths in the Internet typically change as a result of link or node failure, which happens on a much smaller timescale then the timescale used for packet forwarding (*c.f.,* a sensor network, where wireless inference causes paths to change on the same timescale as packet forwarding).

The control plane. The control-plane protocol used in the Internet today is the Border Gateway Protocol (BGP) [92]. BGP allows ASes to discover paths to each destination in the Internet. In BGP, an AS discovers a path to a destination via an *announcement* message that it receives from each of its neighboring ASes. Each announcement contains the AS-level path that the neighbor AS uses to reach that destination. In this thesis, we make the simplifying assumption that each AS selects a *single* path for *all* its traffic to each destination. (In fact, each individual *router* inside the AS selects a single path for each destination, but we ignore this complication in our work.) Path selection is guided by the AS's *routing policies*; these routing policies may depend arbitrarily on commercial, performance, or even security considerations [22].

(In Figure 1.1, we show how BGP announcements allow Comcast, AT&T, Local ISP, and Princeton to discover paths to destination IBM. Routing policies for Local ISP and Princeton are shown inside scrolls. Here, Local ISP prefers the path through Comcast over the path through AT&T, perhaps because Comcast provides service to Local ISP at a lower cost than AT&T. As such, Local ISP routes all traffic destined for IBM over the path through Comcast. As a result, Princeton's most preferred path (Local, AT&T, IBM) is not available, and so Princeton chooses to send traffic over its second favorite path through AT&T.)

The data plane. Once an AS establishes a path to a destination using BGP, the routers inside the AS forward packets along these paths. Because each AS uses BGP to choose a *single* AS-level path to each destination, it follows that packet forwarding from a source to a destination on the Internet typically occurs on a single AS-level path. Even so, packet forwarding can be a non-trivial task; at the core of the Internet, packets must be processed at extremely high speeds (about 2 nsec per packet). To ensure that packet-processing is extremely fast, the data-plane was designed to be quite simple; for instance, it was *not* designed to guarantee that packets will arrive unmodified at their correct destination. As packets travel through the network, congestion at links or nodes can cause packets to be dropped before they arrive at their destination; there

is no mechanism that detects or prevents packet loss[1] . Furthermore, packet modification may occur as a result of device malfunction, link failure, or even malicious attack; because packets in the Internet are usually not authenticated cryptographically, the data-plane does *not* guarantee that packet modification is always detected and/or prevented.

1.1.1 Protocols for improving availability

This thesis studies protocols that can be used to improve availability on the Internet; namely, to improve the chances that the network delivers packets correctly. In this work, we will use the term "secure" to mean that a protocol operates correctly even in the presence of certain misbehaviors by parties on the network. We emphasize that our focus is exclusively on protocols that can be used to improve availability; we do not concern ourselves with protecting confidentiality, privacy, or any other issues traditionally associated with "security".

Ultimately, one of our goals will be to understand whether improved guarantees on availability should be architected into the control-plane, the data-plane, or both. As such, we start by surveying a small sampling of security research proposals that deal with availability on the control-plane and the data-plane.

Securing the control plane. BGP was designed under the assumption that all nodes in the network can trust each other. As such, BGP does not have any mechanisms to validate that a path announced by an AS in BGP is actually used for forwarding traffic, or even *exists* in the Internet topology! The networking research community has put together a number of research proposals to remedy this (see [21] for a comprehensive survey).

The most important of these research proposals is "Secure BGP" (S-BGP) [66]. S-BGP guarantees that ASes can only announce paths that actually exist in the Internet by using digital signatures to cryptographically authenticate each BGP announcement message. This ensures that no AS can announce a path to its neighbors unless that path was announced to it by one of its own neighbors. While S-BGP provides the strongest control-plane security guarantees known to date [21], there are still many hurdles that must be overcome before the protocol can be deployed in the Internet. The most significant of these is probably the fact the security properties of the protocol only take effect *after* it has been adopted by a large number of autonomous systems; however, independently operated ASes will only undertake a costly upgrade to S-BGP once its security benefits have taken effect. In spite of this, practitioners are currently working towards a large-scale deployment of S-BGP [2].

Even though S-BGP defends against announcement of paths that do not exist in the Internet topology, S-BGP does *not* guarantee that a path that appears in a BGP announcement message (*i.e.,* in the control plane) is actually being used for forwarding traffic (in the data plane)! To see how, consider Local ISP in Figure 1.1. Because Local ISP learns two different paths from its two neighbors, AT&T and Comcast, Local ISP can easily send an S-BGP announcement to Princeton containing the AT&T path , while actually forwarding all its traffic over the Comcast path!

Securing the data plane. While most of the security efforts of the networking community have focused on the control plane, earlier studies of routing security focused instead on the data-plane mechanisms. These early works (*e.g.,* Radia Perlman's thesis [89] and the work on Secure Message Transmission [32]) focused on designing protocols that *prevent* packet loss and corruption, even in the presence of adversarial nodes in a network. To do this, these protocols encode and transmit message over multiple paths, such that only a some subset of these paths is

[1]Detecting and preventing packet loss is handled by the transport and application layers; this thesis focuses on the network layer.

controlled by the adversary. However, because these protocols require a source and destination to communicate over *multiple* paths, they are unsuitable for today's interdomain routing system where the source and destination communicate over only a *single* path.

When a source and destination may communicate over only a single path, data-plane mechanisms alone *cannot* guarantee that packets arrive correctly at their destination. (To see why, suppose that an adversary on the path decides to drop all traffic from the source. Then, the source has no way of guaranteeing that his traffic arrives at the destination, unless the source switches to a different path. However, here we shall consider path-switching mechanisms to be part of the control-plane, not the data-plane. We make this distinction because we think of data-plane mechanisms as operating at the level of individual packets; path-switching mechanisms typically operate on an aggregate stream of packets, rather than on individual packets themselves.) For this reason, instead of attempting to *prevent* packet loss, many recent works [28,12,33,59,60,95,96,82,33,76,99,86,13,11,81,10] have focused on developing techniques for *detecting* when packet loss occurs on a path. Some works [11,13,86,109,81,10] take this one step further by also localizing the link that is responsible for packet loss. These protocols can then be used to mitigate packet loss if they are used in conjunction with modern control-plane protocols [55,105] that react to packet loss (and other performance issues) by switching to better paths through the network.

1.2 Our Goals

In this thesis, we take a principled approach to analyze the types of security guarantees that are possible within the engineering and economic constraints of the Internet's interdomain routing system. Our ultimate goal is to inform and advance practitioners' efforts to deploy new security protocols in the system. We do this by analyzing two broad themes:

1. **Which part of the system should be secured?** Should we be designing secure protocols for the control plane, the data plane or both?

2. **What is the right tradeoff between security and efficiency?** Ideally, we would like to design protocols that operate correctly even in the presence of very strong adversarial behavior. However, protocols with strong security guarantees can sometimes come with a cost that makes them impractical for real deployment in the interdomain routing system. As in most traditional settings, one important cost that we consider in this work is *system overhead*; namely, the increase in computation, storage and communication incurred by a network device running the security protocol. A less traditional issue that is extremely important in our setting, is the cost of *participation*; namely, the number of parties in the system that must deploy and participate in a protocol before its security guarantee can take effect. Because the Internet lacks a centralized authority that can force ASes on the Internet to adopt a new security protocol, deploying new protocols in the network requires each AS to independently decide upgrade to the protocol. Thus, we a protocol that requires participation from multiple parties comes at a higher cost than one that requires participation from only a small number parties.

1.2.1 Security guarantees and threat models

We formally characterize the types of security guarantees that can be achieved by different parts of the interdomain routing system. We focus exclusively on protocols that can be used to improve availability in the Internet. We shall consider control-plane protocols separately from data-plane

Misbehavior	Description	Technique
Honest	Follows the protocol	-
Benign	Average-case failure	-
Rational	Strategically deviates from protocol to maximize utility	Game Theory
Adversarial	Worst-case failure	Cryptography

Figure 1.2: Misbehaviors considered in this thesis.

protocols, in order to understand the types of security guarantees that can be built into each part of the system. For a given security guarantee, we shall study the conditions (*e.g.,* system overhead, participation, etc.) that are necessary in order to achieve that security guarantee. In many cases, we shall also design new protocols that achieve the security guarantee.

When we say that a protocol provides a certain "security guarantee", we really mean that protocol should function *correctly* in the face of certain *behaviors* or *threats* in the system. We consider well-defined security guarantees: for each, we will specify the notion of correctness (*e.g.,* "detect if more than 1% of all traffic on a path is being dropped"), and clearly define the behaviors of the parties that participate in the protocol (*e.g.,* "the sender and receiver are honest, and there is a single adversarial party on the path between them that can add/drop/modify packets at will"). At this point, we defer explicit statements of each of the security guarantee to individual chapters in this thesis. Instead, we overview, below and in Table 1.2, the general "threat models" or misbehaviors considered in this thesis.

Because the inventors of the Internet assumed that devices inside the network can be trusted, Internet protocols are typically designed to deal with for honest parties and benign failures:

> **Honest behavior.** An honest party always correctly follows the protocol.
>
> **Benign failure.** We will use benign failure as an umbrella terms for "average-case" deviation from the correct behavior of a protocol. Benign failures can include random link cuts or node failures that case parties to stop responding to protocols. Another example of benign failure is when a router randomly drops packets as a result of congestion. Benign failures are caused by parties that are not strategic or malicious.

Because the Internet is now a federated system consisting of multiple ASes owned by independent profit-seeking businesses, there is a high potential for parties to act selfishly/strategically in order to maximize profits or derive benefits for themselves:

> **Rational (selfish) behavior.** A rational party will strategically deviate from a network protocol in order to derive some well-defined benefit for itself. When we think of rational parties, we first define their utility function. Then, we assume that these parties will attempt to maximize their utility, potentially at the expense of deviating for the correct behavior prescribed by a network protocol. In this thesis, we will use emerging game-theoretic techniques, namely, distributed algorithmic mechanism design, to analyze protocol correctness in the presence of rational behavior (see Chapter 2).

Devices on the Internet are also subject to misconfiguration, or malfunction; they can be commandeered by malicious outsiders or be subverted by disgruntled network operators. The most general way to model these types of misbehaviors is to assume that the device is controlled by a malicious adversary.

> **Adversarial (malicious) behavior.** Unlike the rational party, the adversarial party is not characterized by a utility function. This is models of worst-case behavior;

adversarial parties do anything in their power to break the correct operation of the protocol.[2] In this thesis, we will leverage techniques from cryptography to analyze protocol correctness in the presence of adversarial behavior.

The reader might wonder why we bother with the rational model of behavior, when the more general adversarial models are available. We do this for two reasons:

1. While we always prefer protocols that operate correctly even in the presence of very strong adversaries, these protocols often incur unacceptably high costs (*e.g.,* system overhead, participation). Thus, it sometimes makes sense to design protocols that operate correctly in the presence of realistic models of rational behavior, even if we know that these protocols fail in the presence of adversarial behavior.

2. In this work, we shall prove statements of the form: "Security guarantee X is impossible without (system overhead or participation) cost Y". These statements are actually more convincing if we prove them under the assumption that parties in the network are rational, rather than adversarial! To see why, notice that arbitrarily malicious behavior is a superset of rational behavior. As such, if a security guarantee X requires some (system overhead or participation) cost Y even when parties are rational, then cost Y is also required when parties are arbitrarily malicious.

1.3 Our Contributions

Each chapter of this thesis is completely self-contained, with its own introduction, motivation, notation, and conclusion. We now overview the contents and connections between these chapters, and discuss how they relate to the goals of this thesis, as discussed in Section 1.2.

1.3.1 Securing the control plane (Chapter 2)

Our goal is to study network security protocols that can be used improve availability on the Internet's interdomain routing system. With this goal in mind, there are many reasons why it is natural to consider the security of the control-plane protocols (*i.e.,* BGP) that are used to establish paths through the network. Firstly, recall that in BGP, ASes announce the (AS-level) paths that they use to reach each destination in the Internet. Thus, the design of BGP seems to encourage ASes to rely on path announcements as an accurate indication of the paths that packets take through the network. If BGP announcements did indeed accurately reflect the paths that packets take in the data plane, then an AS could rely on BGP announcements to choose a high-performance AS path for its traffic, or to avoid ASes that it perceives to be unreliable or adversarial. Secondly, as we discussed in Section 1.1.1, control-plane protocols operate at a much smaller timescale than data-plane protocols. As such, the system overhead (*i.e.,* communication, computation, storage) incurred by control-plane protocols is typically less costly than that incurred by data-plane protocols.

Thus, in Chapter 2 we explicitly focus on control-plane protocols, and consider the security requirement of ensuring that the paths announced in the control plane protocol (*i.e.,* BGP, S-BGP, etc.) match the AS-level forwarding paths that are used in the data plane. Because this security requirement is quite strong, we investigate whether it can be met in a weaker, but still realistic, 'threat model' where all ASes in the network are assumed to be rational, rather

[2] Of course, in order to formally model adversarial parties, we must define their adversarial powers. See Section 3.2 for one example.

than arbitrarily malicious (see Section 1.2.1). Assuming that ASes are rational allows us to use game-theoretic tools to reason about when ASes have an incentive to send BGP messages that *deliberately misrepresent* the AS-level paths that their traffic takes through the Internet. We use tools from distributed algorithmic mechanism design (DAMD) to look for conditions under which we could prove that ASes have no incentive to send BGP announcements that misrepresent the forwarding paths they use in the data-plane. Earlier attempts within the DAMD framework [39, 73, 35, 36, 37, 38, 84, 87]assumed that the utility of an AS is completely determined by the *outgoing path* its traffic takes to the destination. However, this model of utility fails to capture the fact that many ASes are paid by their customers to carry incoming traffic (*e.g.,* In Figure 1.1, Princeton pays AT&T to carry its traffic.) Thus, for the first time, our work considers ASes with utility functions that also depend on the *incoming traffic* that they attract to their networks.

Our analysis yields some surprising results. We first show that even if we assume that ASes are rational, and even if they all use S-BGP, the strongest known secure routing protocol (Section 1.1.1), then some ASes may still benefit from sending BGP messages that misrepresent the paths that they use for forwarding traffic. We then prove that there do exist certain conditions under which ASes have no incentive to misrepresent their about forwarding paths; however, these conditions require unrealistically strong assumptions on the routing policies of every AS in the Internet.

Thus, the results in Chapter 2 suggest that ASes should not rely on traditional secure routing protocols (like S-BGP [66]) to improve availability by choosing high-performance/trusted paths for their traffic.

1.3.2 Data-plane path-quality monitoring (PQM) (Chapters 3-4)

In Chapters 3-4, we move away from control-plane mechanisms, and focus instead on data-plane mechanisms that can be used to improve availability. Here, instead of taking the more traditional approach of *preventing* packet loss by sending traffic over *multiple* paths, we instead focus on the more realistic *single* path setting. (Recall that with BGP, routers chose a single path for all their traffic to a destination.) We study *path-quality monitoring (PQM)* protocols that run in the data-plane and *monitor* packet loss and corruption on a single path through the Internet. Then, packet loss and corruption can be *prevented* by combining these PQM protocols with control-plane techniques (*e.g.,* intelligent route control, source routing, overlay routing [55]) that give source networks greater flexibility when selecting a path to a destination; if the monitoring protocol indicates that packet loss or corruption on a path is too high, the source can switch to another (better) path. Because we want path-quality monitoring protocols that can be used to inform routing decisions, our goal is to design protocols that can run in high-speed routers. Furthermore, we require these protocols to return correct information, even when adversarial nodes on the path interfere with the monitoring process.

Because our goal is to design PQM protocols that run in the data-plane of high-speed Internet routers, our protocols need to be able to keep up with the high packet-processing speeds and traffic volumes at the core of the Internet. Thus, the question of security *v.s.,* efficiency becomes paramount. Indeed, we argue (informally) that if data-plane monitoring protocols are required to return correct information even when adversarial nodes on the path try to bias monitoring results, then these protocols incur high overheads, related to the amount of traffic sent in the data-plane. To see why, notice that if traffic pertaining to the monitoring protocol can be distinguished from regular data-plane traffic, then the adversary can bias the outcome of monitoring protocol by selectively dropping the regular traffic, while providing good performance for the monitoring traffic. Thus, ensuring that the outputs of the protocol cannot be biased

requires us to make PQM-related traffic indistinguishable from regular traffic send on the path. Providing this indistinguishablity introduces system overheads that are roughly proportional to the amount of traffic sent in the data plane.

In Chapter 3 we consider protocols that can *detect* high rates of packet loss/corruption, even in the presence of adversaries. In Chapter 4, we take this security requirement one step further by considering protocols that can also *localize* the (possibly adversarial) link responsible for dropped/corrupted packets. Our major objectives in each of these chapters is to understand the cost (in terms of system overhead and participation, see Section 1.2) of each type of security requirement. Along the way, we also design some interesting detection and localization protocols.

Detecting the adversary.

In Chapter 3 we consider protocols that allow a source to detect high rates of packet loss and corruption on data-plane path.

We start by using simple cryptographic proof techniques (*i.e.,* reductions [50]) to prove that *any* protocol that robustly detects high rates of packet loss and corruption in the presence of adversaries requires that the sender and receiver share secret keys and perform cryptographic operations. We then use cryptographic and data-streaming approaches to design a number of highly-efficient detection protocols. One of our protocols, the "secure sketch", can monitor up to a billion packets without marking normal data-plane traffic, and using only two control messages and 200-600 bytes of storage at the source and destination only. (Asymptotically, monitoring T packets requires $O(\log T)$-storage, and two control messages.) We prove that all our protocols satisfy a precise definition of security, and derive analytic expressions for the tradeoff between statistical (measurement) accuracy and storage overhead for each protocol.

The results of Chapter 3 are encouraging; by focusing on the modest security requirement of detecting packet loss/corruption, we are able to design highly-efficient protocols that can withstand very strong adversaries. Furthermore, all of our protocols require the participation of the source and destination only; no other node on the path is required to participate.

Localizing the adversary.

While it is useful to enable sources to detect packet loss and corruption on path, it is even more useful to be able to localize the adversarial node responsible for tampering with packets. Thus, in Chapter 4, we use a similar adversarial model to study a *stronger* security requirement; namely that a source can localize the link that is responsible for high packet loss or corruption.

We start by developing a formal cryptographic model of security for the localization problem, and use this formal model to find security vulnerabilities in previously published works [86,13,10]. We then present a number of localization protocols. One of our protocols can monitor T packets using $O(\log T)$-storage per node, two additional control messages, and shared keys between the source node and every other node on the path. While the detection protocols of Chapter 3 require participation from the source and destination only, all known localization protocols *e.g.,* [11,13,86,109,81,10], including the ones we design in Chapter 4, require participation from *every* node on the path. It is natural to ask if these high levels of participation are necessary. We answer this question in the affirmative by leveraging cryptographic proof techniques (black box separations [62]) to argue that *any* protocol that correctly localizes links responsible for packet loss and corruption in the presence of adversaries, requires *every node on the path* to share secret keys with the source, and perform cryptographic operations.

Thus, the results of Chapter 4 suggest that security requirement of localizing links responsible for packet loss/corruption might be too ambitious for the interdomain routing system; we may be

better off with the more efficient protocols that only *detect* packet loss/corruption, as designed in Chapter 3.

1.4 Conclusions, Implications and Future Directions

In Section 1.2, we mentioned that the goals of this work are to understand which parts of the interdomain routing system should be secured, and to study the tradeoffs between security and efficiency. We now discuss how our results and several new research directions can begin to address these goals. We also overview the implications of our work on the design of network architectures that guarantee availability in the presence of selfish or adversarial behavior.

1.4.1 Which part of the system should be secured?

Should we be designing secure protocols for the control plane, the data plane, or both?

Securing the control plane is not a panacea. Our results in Chapter 2 suggest that availability will still be a challenge even if the strongest known secure routing protocol (S-BGP) is fully deployed in the Internet. We showed that, even if we assume that all ASes in the network use S-BGP, and are *rational* (rather than *adversarial*), ASes still have an incentive to announce AS-level paths in the control plane that do not match the paths actually used in the data-plane. Thus, our analysis shows that it is unreasonable to assume that an AS can rely on BGP messages to choose paths that circumvent routing traffic through untrusted or adversarial ASes.

It is interesting to note that our analysis in Chapter 2 is a *worst-case* analysis. We show that if ASes are rational and use S-BGP, then *there exist network topologies* where at least one AS has an incentive to send a BGP announcement that misrepresents the path he uses in the data plane. To better understand the practical relevance of the results in Chapter 2, we would also like to answer the following questions: Firstly, how often do such network topologies (where ASes have an incentive to lie) appear in practice – do they only exist in the obscure corners of the Internet, or are they extremely prevalent? Secondly, how effective is S-BGP in reducing the number of ASes with an incentive to misrepresent their paths – how many more ASes can get away with lying if we assume that ASes use plain BGP, as compared to S-BGP or some other secure routing protocol [21]?

We are in the process of conducting an empirical study of the Internet's topology that seeks to answer some of these questions. Preliminary results suggest that even if all ASes in the Internet use S-BGP, many ASes will still have an incentive to lie in their BGP announcements.

Strong security guarantees are possible in the data-plane. Our results suggest that it is feasible to design secure protocols that are efficient enough to run in the data-plane of high speed routers, especially if we consider protocols that only require the participation of a source and destination. Indeed, we were able to design highly efficient path-quality monitoring (PQM) protocols that operate correctly even in the presence of a very strong adversary that knows the details of the monitoring protocol and can add/drop/modify traffic at will.

While this thesis focused on PQM protocols that monitor packet loss and corruption, there are many other metrics that can determine path quality, including traffic latency (delay), jitter (delay variance), and packet lag (the number of packets that arrive out-of-order at the destination). We believe that designing efficient and secure PQM protocols for these metrics is a worthwhile direction for future work.

Hop-by-hop protocols vs. end-to-end protocols. On one hand, we can design 'end-to-end' security protocols do not require knowledge of the *identities* of the nodes on the path between the source and destination, like the detection protocols of Chapter 3. On the other

hand, we can design 'hop-by-hop' protocols that require the sender to know the identities of the nodes on a path, like the localization protocols of Chapter 4. However, our results indicate that (a) control-plane protocols like BGP and S-BGP do not always accurately return information about the identities of the ASes on a data-plane path, and (b) data-plane protocols that localize an adversary are expensive, because each node on the path has to participate. Taken together, these results suggest that hop-by-hop protocols are impractical; indeed, such protocols are likely useful only in limited settings where the need for security is so strong that it overwhelms such practical concerns.

We believe that promising direction for future research is to analyze other security functionalities that can be realized in an end-to-end manner. For instance, certain control-plane path-switching protocols (*e.g.,* multipath routing, overlay routing [55]) can be realized in an end-to-end manner. However, more work is required to characterize the security guarantees that can be achieved by these path-switching protocols, especially when they are combined with end-to-end PQM protocols as discussed in Chapter 3.

1.4.2 Security versus efficiency

As we discussed in Section 1.2.1, our notion of a "security guarantee" for a protocol has two parts: a notion of "correctness", and a "threat model". We would like our protocol to operate correctly even in the presence of parties that behave (and misbehave) in the ways specified by the threat model. Ideally, we would like to design protocols that provide strong security guarantees; however, these protocols often come at high cost, either in the form of system overhead (*e.g.,* computation, storage, communication resources) or participation (*i.e.,* many nodes in the network must deploy the protocol, so that deploying these protocols in the Internet becomes a challenge, see Section 1.2). In order to understand which security guarantees are feasible within the engineering and economic constraints of the Internet's routing system, we studied ways to tradeoff between strong security guarantees and protocol cost. One way to do this is to consider weaker notions of protocol correctness; another is to consider weaker threat models. Indeed, this thesis takes both of these approaches:

Weaker notions of correctness. Our study of path-quality monitoring in both Chapters 3-4 considered the following 'threat model': a source and destination trust each other, while an adversary that drops and corrupts traffic occupies any subset of the nodes on the path between them. The ideal notion of correctness in this setting would be to empower the source to *localize* the adversarial nodes; however, in Chapter 4 we show that achieving this notion of correctness comes at the unacceptably high cost of requiring all the nodes on the path to participate in the protocol. Thus, in Chapter 3 we show that a weaker notion of correctness, *i.e.,* empowering the source to *detect* when packets are lost/corrupted, comes at a much more reasonable cost, *i.e.,* participation by the source and destination only.

Indeed, we believe that analyzing a spectrum of notions of correctness is a useful exercise, especially when architecting networks with practical and useful security guarantees. We believe that a number of other problems in network security could benefit from this approach.

Weaker threat models. Now consider the following notion of correctness: ensuring that ASes send BGP messages that accurately reflect that AS-level paths that they use in the data plane. Viewing our results in Chapter 2 in the broader context of the work on distributed algorithmic mechanism design and BGP, we see that this notion of correctness has been studied for a variety of different 'threat models'. For instance, Levin, Schapira and Zohar [73] show that full deployment of S-BGP is a sufficient condition for this notion of correctness, as long as ASes are modeled as rational with utility that depends only on the *outgoing path* that they use for their traffic. However, once we consider a stronger threat model, where ASes' utility

also depends the *incoming traffic* routed through their network, our results in Chapter 2 show that S-BGP alone is no longer sufficient; we also need to (unrealistically) constrain the set of allowable routing policies. Finally, if we assume ASes are adversarial, even constraining the set of allowed routing protocols and requiring nodes to use S-BGP is insufficient for this notion of correctness.

While it is not surprising that this notion of correctness becomes increasingly difficult to achieve as the threat model becomes stronger, it is interesting to note that these results are extremely *sensitive* to the strength of threat model. This observation suggests that we must be very careful in extrapolating from positive results obtained in a weak threat model (*i.e.,* statements of the form: condition X guarantees Y notion of correctness for threat model Z) to the real world. Indeed, this is likely one of the reasons for the success of strong cryptographic threat models, in which parties are assumed to be arbitrarily malicious. On the other hand, we do view weak threat models as a useful tool for proving very convincing negative results (*i.e.,* statements of the form: notion of correctness Y for threat model Z cannot be achieved without condition X). For instance, our results in Chapter 2 show that S-BGP is not sufficient for matching the control- and data-plane *even if ASes obey a realistic, well-defined notion of rationality*; it immediately follows that S-BGP will not guarantee that the control- and data-plane match when ASes are adversarial.

1.4.3 Implications on network architecture

To summarize our discussion, we discuss the implications of our work on the design of networks that can withstand selfish or adversarial behavior, and present a number of other open questions.

Firstly, we believe that any solution that purports to improve availability must include some data-plane security component; indeed, our results in Chapter 2 suggest that even if we assume ASes are rational, control-plane security protocols are not sufficient to ensure ASes do not misbehave in the data plane.

Secondly, we believe that the most promising direction for improving availability in the setting of interdomain routing is focus on protocols that take an end-to-end view of the network; in particular, we advocate for combining intelligent route control protocols [55,105] with the end-to-end PQM protocols proposed in this thesis.

Thirdly, while this thesis suggests that securing the control plane is not a panacea, we do believe that control-plane security protocols have an important role to play in making the interdomain routing system more predictable and robust. However, it is unclear which of the many of proposed control-plane security protocols [21] are 'right' for the interdomain. As such, we believe that it would be valuable to have further studies *comparing* the deployability and security guarantees provided by each of these protocols.

Finally, another interesting direction (that we did not investigate here) is the question of accountability and contracts in the Internet. Because ASes are controlled by profit-seeking businesses, it may be possible to enforce 'good behavior' in the interdomain routing system by designing a system of contracts that penalizes ASes that perform poorly, *e.g.,* by dropping or corrupting packets. While there have been a number of interesting works in this direction [10, 69, 74, 40, 26], many of these results assume that the existence of hop-by-hop secure PQM protocols that we showed to be impractical (Chapter 4). As such, we believe that the question of designing a practical accountability system for the Internet, that uses only end-to-end security protocols, remains open for future research.

Chapter 2

Incentives for Honest Path Announcements in BGP

2.1 Introduction

Interdomain routing on the Internet consists of a *control plane*, where Autonomous Systems (ASes) discover and establish paths, and a *data plane*, where they actually forward packets along these paths. The control-plane protocol used in the Internet today is the Border Gateway Protocol (BGP) [92]. BGP is a path-vector protocol in which ASes discover paths through the Internet via announcements from neighboring ASes. In BGP, each AS has routing policies that may depend arbitrarily on commercial, performance, or other considerations. These policies guide the AS's behavior as it learns paths from its neighbors, chooses which (if any) neighbor it will forward traffic to in the data plane, and announces path information to its neighbors. The design of BGP seems to encourage ASes to rely on path announcement as an accurate indication for the paths that data-plane traffic follows. However, BGP does not include any mechanism to enforce that these announcements match actual forwarding paths in the data plane.

Traditional work on securing interdomain routing (*e.g.,* Secure BGP (S-BGP) [66] and the like [106, 52, 21]) has focused on the control plane, with the loosely-stated goal of ensuring "correct operation of BGP" [66]. However, addressing the control plane in isolation ignores the important issue of how packets are actually forwarded in the data plane. Here, we explicitly focus on the security goal of ensuring that the paths announced in the control plane match the AS-level forwarding paths that are used in the data plane; this has been implicit in many previous works (on securing BGP [66,106,52] and incentives and BGP [36,38,35,37,39,87,73]). This way, an AS can rely on BGP messages, *e.g.,* to choose a high-performance AS path for its traffic or to avoid ASes that it perceives to be unreliable or adversarial [14,91,57].

This goal has recently received some attention by works [99,107,74,10] that suggest auxiliary enforcement protocols that operate in the data plane. However, because such solutions typically incur a high overhead (see Section 2.1.1), here we consider solutions that operate in the control plane alone. Furthermore, most works on BGP security assume ASes can be arbitrarily malicious. Here, we instead follow a different line of research where ASes are modeled as *rational*, *i.e.,* act in a self-interested manner. In our work, we define this to mean that ASes both (1) try to obtain the best possible outgoing path for their traffic, while (2) also attracting incoming traffic (see Section 2.1.3). We look for conditions under which rational ASes have *no incentive to lie* about their forwarding paths in their BGP path announcements. We find that protocols like S-BGP [66] are generally *not* sufficient to prove that ASes have no incentive to lie about forwarding paths; we also require unrealistically strong assumptions on the routing policies of

every AS in the network. Our results emphasize the high cost of ensuring that control- and data-plane paths match, even if we assume that ASes are rational (self-interested), rather than arbitrarily malicious.[1]

In the rest of this section, we motivate our approach, discuss related work, outline our results and discuss their implications. The model we use is defined in Sections 2.2–2.3, and our results are detailed in Sections 2.4–2.6. Related work is discussed further in Section 2.7. Proofs and additional discussion can be found in the appendices.

2.1.1 Matching the control and data planes.

One way to enforce honest path announcements in BGP is to deploy AS-path measurement and enforcement protocols that run in the data plane. However, determining AS-level paths in the data plane is a nontrivial task even in the absence of adversarial behavior (*e.g.,* [77] discusses the difficulty of determining AS-level paths from traceroute data). When dealing with ASes that may have incentives to announce misleading paths in the control plane, we need AS-path enforcement protocols that cannot be "gamed" (*e.g.,* by ASes that send measurement packets over the path advertised in the control plane, while sending regular traffic over a different path). Thus, data-plane enforcement protocols [74, 107, 86, 10] must ensure that measurement packets are indistinguishable from regular traffic, resulting in high overheads that are usually proportional to the amount of traffic sent in the data plane. Also, while secure *end-to-end* data-plane protocols can robustly monitor performance and reachability, *e.g.,* [12,49], these protocols do *not* trace the identities of the ASes on a data-plane path; securely tracing AS paths requires participation of every AS on the path [74, 10, 86, 107].

Alternatively, one could hope to ensure that control- and data-plane paths match by ubiquitously deploying S-BGP [66] and the like [21]. This provides a property called path verification [73], which ensures that no AS can announce a path to its neighbors unless that path was announced to it by one of its neighbors. While path verification defends against announcement of paths that do not exist in the Internet topology [66], it does not, by itself, ensure that control- and data-plane paths match. For example, an AS a with two different paths announced by two different neighbors can easily lie in its path announcements—announcing one path in the control plane, while sending traffic over the other path in the data plane.

While it is tempting to argue that ASes are unlikely to lie about their forwarding paths because they either fear getting caught or creating routing loops, this argument fails in many situations. The hierarchy in the Internet topology itself often prevents routing loops from forming, *e.g.,* if the lie is told to a stub AS, or see also [15]. Furthermore, empirical results indicate that catching lies can be difficult, because even tracing AS-level paths that packets traverse in the data plane is prone to error [77]. Finally, to minimize the likelihood of getting caught, an AS could lie only when it has a good idea about where its announcements will propagate.

2.1.2 The game-theoretic approach.

In this work we explore the extent to which we can use *only control-plane mechanisms*, in conjunction with assumptions on AS policies, to motivate ASes to honestly announce data-plane paths in their BGP messages. Our exploration is carried out within the context of distributed

[1] We do not consider situations when the control and data plane do not match due to malfunction or misconfiguration; we consider this *irrational* behavior. We also do not consider control- and data-plane mismatches caused by path aggregation [77], since typically only last hop of the (data-plane) AS-path is omitted from the BGP path announcement.

Control-plane verification	**Model of AS utility**			
	No traffic attraction	Increase volume of incoming traffic (Section 2.4)	Attract customer traffic via direct link (Section 2.6)	Generic traffic attraction (Section 2.5)
None	Policy consistency Consistent export [39, 37]	Next-hop policy All-or-nothing export	No known restrictions suffice	
Loop			Policy consistency Gao-Rexford conditions Next-hop at attractees Consistent export	Next-hop policy All-or-nothing export
Path	Arbitrary [73]	Policy consistency Consistent export		

Table 2.1: For each utility model and type of control-plane verification, the additional restrictions that ensure that ASes in a network with no dispute wheel have no incentive to dishonestly announce paths.

algorithmic mechanism design [84, 36], which is rooted in game theory. This paradigm asserts that ASes are *rational players* that they participate in interdomain routing because they derive utility from establishing paths and forwarding packets; ASes will do whatever they can to maximize their own utility. The task of mechanism design is to ensure that the incentives of rational players are aligned with accomplishing the task at hand, so players have no incentive to deviate from the prescribed behavior.

The paradigm of algorithmic mechanism design in the context of routing was first suggested by Nisan and Ronen [84]. Feigenbaum *et al.* [36] brought *distributed* algorithmic mechanism design to the study of incentives in routing and shifted the focus to *interdomain* routing and BGP in particular. Rather than a centralized mechanism that sets up paths, the model in [36] postulates that paths are set up in a distributed fashion by the economically interested ASes themselves. The model was further developed in a sequence of works [36, 87, 35, 39, 37, 38, 73, 30]. Our model builds upon the work of Levin, Schapira, and Zohar [73], who brought a fully formal game-theoretic and distributed-computational model to this line of research (Section 2.2 and Appendix A.1). When the prescribed behavior includes the requirement that ASes honestly announcing forwarding paths to their neighbors (as is the case in all prior work), and when every AS follows this behavior, then the control plane and the data plane will match. In this sense, all work within this paradigm implicitly addressed matching the control and data planes. In this work, we highlight this matching (which is strictly weaker than the goal in prior work) as a stand-alone security property that should be addressed on its own.

2.1.3 Modeling utility with traffic attraction.

Recent work of Levin *et al.* [73] shows that if ASes are rational, then path verification (*e.g.,* S-BGP) is sufficient for honest path announcements, even when ASes have arbitrary routing policies. This encouraging result improved on earlier work [36, 37, 35, 38, 39] that explored restricted classes of routing policies. For example, Feigenbaum *et al.* [37, 39] found that it is sufficient to require policy consistency, a generalization of shortest-path routing and next-hop policy that requires that the preferences of neighboring ASes regarding different paths always agree. However, these results [36, 87, 35, 39, 37, 38, 73] were obtained under the assumption that the utility an AS derives from interdomain routing *is entirely determined by the outgoing path that traffic takes to the destination.* In reality, however, the utility of an AS is likely to be influenced by many other factors. For example, the utility of a commercial ISP may increase when it carries more traffic from its customers [58], or a nefarious AS might want to attract traffic so it can eavesdrop, degrade performance, or tamper with packets [91, 57, 14].

Here, we use a more realistic utility model (see Section 2.2.3), focusing in particular on the effect of traffic attraction, where the utility of one AS increases when it transits *incoming* traffic

from another AS. We consider three models of traffic attraction. In our first model, traffic-volume attractions, utility depends only the origin of the incoming traffic, but not on the path that it takes. This captures the notion that an AS may be interested in increasing the volume of its incoming traffic or that a nefarious AS might want to attract traffic from a victim AS, in order to, say, perform traffic analysis. Our second model, generic attractions, encompasses all forms of traffic attraction; the utility of an AS may depend on the path incoming traffic takes. Our third model, customer attractions, is more restrictive. This model assumes that utility increases only if an AS attracts traffic from a neighboring customer AS that *routes on the direct link* between them; this models the fact that service contracts in the Internet are typically made between pairs of neighboring ASes [58] (Section 2.3.3).

2.1.4 Overview of our results.

In this work, we want to argue that under some set of conditions, any utility that an AS can obtain by lying in BGP announcements could also be obtained with honest announcements. Unfortunately, we find that conditions from previous work do not suffice when we consider traffic attraction: neither path verification [73] nor policy consistency [39, 37] alone is sufficient. (See Figures 2.2, 2.3, and 2.5 for examples.) These disappointing results motivate our search for new combinations of conditions (on control-plane verification, routing policy and export rules) that ensure that ASes have an incentive to honestly announce paths.

In addition to path verification (*e.g.,* S-BGP), we introduce a weaker form of control-plane verification called loop verification (Section 2.5.3), which roughly captures the setting in which an AS is caught and punished if it falsely announces a routing loop. Loop verification can be thought of as a formalization of "the fear of getting caught," and it may be easier to deploy than path verification.

In addition to policy consistency, we also consider the more restrictive next-hop policy, which roughly requires ASes to select paths to a destination based only on the immediate neighbor that advertises the path (Section 2.3.2). We also consider the Gao-Rexford conditions [42] (Section 2.3.3). These conditions, which are believed to reflect the economic landscape of the Internet [58], assume routing policies are restricted by business relationships between neighboring ASes, *i.e.,* by customer-provider relationships (the customer pays the provider for service) and peer-to-peer relationships (peer ASes transit each other's traffic for free).

Finally, we consider several classes of export rules (Section 2.3.4) that dictate whether or not an AS announces paths to its neighbors. An all-or-nothing export rule requires that, for each neighbor, an AS either announces every path or no paths. We also consider a more realistic consistent export rule [37] that roughly requires that ASes' export rules agree with their routing policies.

For many combinations of the conditions discussed above, we can still find examples in which ASes have an incentive to lie about their data-plane paths. However, for some combinations we obtain positive results, as sketched in Table 2.1. (These results all assume a network condition called "no dispute wheel" [53]; see Section 2.3.1.) Furthermore, our results are "tight", in that for every combination of the considered conditions, either one of our positive results applies or one of our negative examples does (as summarized in Tables 2.2–2.4).

Our positive results show that, for *every network* satisfying some combination of conditions, any utility an AS gains by lying can equivalently be obtained if that AS had instead *honestly announced paths to only an subset of its neighbors* and announced no paths to all other neighbors. That is, we show the existence of an *export rule* for which each AS obtains its optimal utility. As in previous work [73, 39, 37], our positive results for traffic-volume attractions (Section 2.4) and customer attractions (Section 2.6.2) also explicitly define an optimal export rule. Our positive

result for generic attractions (Section 2.5.4) shows that an optimal export *exists*, but does not explicitly state what it is (Section 2.5.5). We discuss the notions used for our positive results further in Appendix A.1.

2.1.5 Implications of our results.

Our results suggest that even with control-plane enforcement mechanisms, ASes may have incentive to lie in their BGP announcements, unless very strong restrictions are imposed on their policies. As sketched in Table 2.1, from the set of conditions we considered, we always need *every AS in the network* to obey (1) unrealistic restrictions on its preferences (such as next-hop policy) and (2) explicit restrictions on export rules. Most of our results also require (3) full deployment of either path or loop verification. Thus, our results point to a negative answer to the question that we set out to investigate—practically speaking, it is unlikely that we could use only control-plane mechanisms to remove the incentives for ASes to announce false paths in BGP.

This suggests a choice. We can either employ expensive data-plane path enforcement techniques [10, 74, 86, 107] when it is absolutely necessary to ensure that packets are forwarded on AS-level paths that match an AS's routing policies, or dismiss this idea altogether and instead content ourselves with some weaker set of goals for interdomain routing. It is certainly possible to formulate weaker but meaningful security goals and show that certain control-plane mechanisms or data-plane protocols meet these goals. However, doing this invites the question: if we are not interested in ensuring that AS paths announced in BGP are really used in the data plane, then why use a path-vector protocol at all?

2.2 Modeling Incentives and BGP

We now present the formal model in support of our results in Sections 2.4–2.6. The model builds on the literature [53, 36, 73] and extends prior work by explicitly considering traffic attraction. (We also make more explicit distinctions between control- and data-plane actions.)

2.2.1 The AS graph.

An interdomain-routing system is modeled as a labeled, undirected graph called an AS graph (see Figure 2.1). For simplicity, each AS is modeled as a single node, and edges represent direct (physical) communication links between ASes. Adjacent nodes are called neighbors. We denote nodes by lowercase letters, typically a, b, c, d, m, and n. We follow [53] and assume the AS-graph topology does not change during execution of the protocol.

Because, in practice, BGP computes paths to each destination separately, we follow the literature [53] and assume that there is a unique *destination node* d to which all other nodes attempt to establish a path. (Thus, like most previous work, we ignore the issue of route aggregation [77].) We denote paths by uppercase letters, typically P, Q, and R.

2.2.2 The interdomain-routing game.

We extend the model of Levin *et al.* [73] that describes interdomain routing as an infinite-round game in which the nodes of the AS graph are the strategic players. In each round, one node in the graph processes the most recent path announcements (if any) from its neighbors and then performs two *actions*: (1) it decides on an outgoing link (if any) to use in the data plane; and

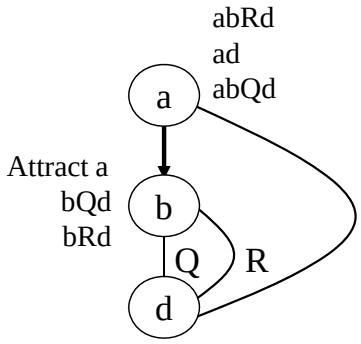


Figure 2.1: AS graph with traffic attraction.

(2) decides on paths (if any) to announce to its neighbors.[2] Note that, just as in [73], nodes have the opportunity to announce their true data-plane path choice, but they are not forced to do so. The order in which nodes act is called the *schedule*.

We assume that path announcements sent between neighbors on direct links cannot be tampered with (by a node not on the direct link). This can be enforced via the BGP TTL Security Hack [44] or via a pairwise security association between nodes using the TCP MD5 security options [56]. We further assume that each node has the opportunity to act infinitely often—*i.e.,* the schedule is *fair*.

Game outcome and stability. The *state* of a node n at some round in the game consists of a data-plane component (the outgoing link most recently chosen by n) and a control-plane component (the announcements most recently sent by n). This state is *transient* if it occurs only finitely many times and it is *persistent* otherwise. There could be many possible sequences of states; the sequence depends on both the schedule and the actions of nodes while playing the game. When we ask whether or not there is an incentive to lie, we are interested in the more precise question: Is there a fair schedule in which a node may have an incentive, in some round, to announce a route in the control plane that is not its data-plane choice?

The global state at some round is the collection of all node states at that round. A global outcome of a game is a global state that does not contain any transient node states.We note that there could be more than one such global state; in particular, a persistent control-plane oscillation among nodes is a sequence that infinitely transitions among non-transient node states, even for a fixed schedule. Our results in this work hold regardless of which of these is taken to be the global outcome.

If the state of a node is constant after some round then this state is locally stable. A global outcome is globally stable if all node states in it are locally stable. (This definition of stability is compatible with the original definition in [53].) We typically denote global outcomes by T or M. We may use "outcome" informally to mean the control-plane or data-plane component of the outcome when the component is clear from the context.

2.2.3 Utility, valuation, and attraction.

A *strategy* is a procedure used by a node to determine its actions in the game. In principle, a node can make decisions in any way that it wants, but here we assume that nodes are *rational*. In particular, each node b has a utility function $u_b(\cdot)$ mapping outcomes to integers (or $-\infty$); b tries to act to obtain an outcome T that maximizes $u_b(T)$.

[2] A node can also decide not to route on any link in the data plane, or not to announce anything to its neighbors.

We assume that every node b in the graph has a utility function of the form

$$u_b(T) = v_b(T) + \alpha_b(T) \tag{2.1}$$

where $v_b(T)$ is the valuation function that depends only on the simple data-plane path from b to d in T, and $\alpha_b(T)$ is the attraction function that depends only on the simple data-plane paths from other nodes to b in T. (We write the utility function as a *sum* of the valuation and attraction functions; in fact, our results require only that utility increases monotonically with each of the valuation and attraction functions.) In this work, utility depends on the data-plane component of outcome alone (because the control-plane component may not correspond to actual traffic flow in the network).

The valuation function $v_b(\cdot)$ is the same as was considered in previous work on incentives and BGP [39, 73, 35, 36, 37, 38, 87, 30]. It is meant to capture the intrinsic value of each outgoing path (*e.g.,* as related to the cost of sending traffic on this path, its reliability, the presence of undesirable ASes on it, *etc.*). We assume that nodes dislike disconnection, so that if node b has no data-plane path to the destination in outcome T, then $v_b(T) = -\infty$. (The implications of this are discussed further in Section 2.2.7.)

The attraction function $\alpha_b(T)$ is the new component of utility that we add in this work. Because we are interested in situations where nodes may want to attract traffic (and not deflect it), our most general form of the attraction function only requires that $\alpha_b(\cdot)$ does not increase when edges leading to b are removed from the data-plane outcome. Formally, for an outcome T and node b, let $T(b)$ be the set of edges along simple paths from other nodes to b in the data-plane component of T (*e.g.,* if T's data-plane links form a routing tree, then $T(b)$ is the subtree rooted at b). We assume that for every two outcomes T and T' and every node b, if $T'(b) \subseteq T(b)$, then $\alpha_b(T') \leq \alpha_b(T)$. This general condition covers many forms of traffic attraction; *e.g.,* attraction can depend on which links are traversed by incoming traffic at a node, and not just the nodes from which that traffic originates.

We also consider two specific forms of traffic attraction. First, traffic-volume attraction requires that $\alpha_b(T)$ depends only the origin of the incoming traffic, but not on the path that it takes. More formally, if $T(b)$ and $T'(b)$ include the same nodes then $\alpha_b(T) = \alpha_b(T')$. This also captures the idea of nefarious ASes who want to attract traffic for eavesdropping on or tampering with traffic (but see also Section 2.2.7).

Another specific form of attraction is customer attraction, in which the AS graph is assumed to have underlying business relationships, and $\alpha_b(T)$ depends only on customer nodes a that route through b on the direct a-b link between them. We further discuss this form of attraction and customer-provider relationships in Section 2.3.3.

We say that there is an *attraction relationship* between a and b if the attractor b increases its utility when the attractee a routes traffic through it (*e.g.,* as in Figure 2.1). In Figure 2.1, we depict the utility function of each node next to that node: say that the attraction function of b is such that it earns 100 points of utility when it attracts traffic from a, and that the valuation function of b is such that it earns 10 points of utility when using the path bQd and only 1 point of utility when using the path bRd. Then, following Equation 2.1, the use of data-plane path $abRd$ earns b 101 points of utility.

2.2.4 BGP-compliant strategies.

Recall that we are interested in ensuring that the interdomain-routing control and data planes match. When all nodes follow the rules prescribed by the BGP RFC [92] in their execution of the protocol, this is achieved. We call a strategy that obeys these rules a *BGP-compliant strategy*, as formalized below.

Definition 2.2.1. A BGP-compliant strategy for node n depends on two functions: A ranking function $r_n(\cdot)$ mapping each path to an integer or $-\infty$; and, an export rule $e_n(\cdot)$ that maps each path P to the set of neighbors to which n is willing to announce the path P. A path P is admitted at n if $r_n(P) > -\infty$. Paths that include routing loops or that do not reach the destination are not admitted at any node. We require that, for any two paths P and Q admitted at n that begin with different next hops, it holds that $r_n(P) \neq r_n(Q)$. (Note that $r_n(\cdot)$ and $e_n(\cdot)$ act *only* on path announcements, rather than game outcomes (*e.g.,* data-plane paths).)

The strategy of node n is BGP-compliant, with $r_n(\cdot)$ and $e_n(\cdot)$ as defined above, if n does the following in each round in which it participates. Node n first chooses the path P such that (a) P has highest rank of all the most recently announced paths received from neighbors, and (b) the first node a of P is the neighbor that announced P to n. Then, n performs the following two actions: (1) n chooses the outgoing link to a in the data plane; and (2) n announces the path nP to all neighbors in $e_n(P)$.

This definition explicitly assumes that the all traffic to the destination is routed over a single next-hop. (We do not address here the question of modeling multipath routing.) Also, we assume that, if n does not receive any announcements with an admitted path, then n does not route on any outgoing link or announce any paths to its neighbors. (Notice that we model *ingress filtering* using the concept of admitted paths and *egress filtering* using the concept of an export rule.)

Control-plane announcements from a node executing a BGP-compliant strategy match its next-hop choices in the data-plane. Thus, if *all* nodes in the network use BGP-compliant strategies, then the control and data planes will match. (We may informally call a node executing a BGP-compliant strategy a *BGP-compliant node*, or sometimes an *honest node*.) In the positive results from previous work [37,39,73] included in Table 2.1, the prescribed strategies are examples of BGP-compliant strategies in the sense of Definition 2.2.1. Thus, those results also achieved agreement between the control and data planes, but contrary to the current work, they do not consider traffic attraction.

We stress that Definition 2.2.1 gives BGP-compliant nodes the leeway to choose their ranking and export functions in any way they want, in order to try to achieve a utility-maximizing outcome in the game. In the next subsection, we discuss the relationship between utility and the ranking and export functions in a way that encompasses earlier work (without traffic attraction) and the results in this work (with traffic attraction).

2.2.5 From utility to ranking and export.

To map between our model and real-world implementation of BGP [92], we can think of the actions of the game described in Definition 2.2.1 (*i.e.,* (1) selection of next-hop, and (2) announcements to neighbors) as being executed by nodes, in practice, through setting parameters in the ranking and export functions. In previous work [39, 73], the ranking function was set equal to the valuation function (we denote this as $r_n(\cdot) \equiv v_n(\cdot)$)[3]: the larger the valuation of a path, the higher its rank. This follows from the fact that in previous work, the utility of an AS was defined to be its valuation function,[4] and thus the directly determined the ranking function. However, the direct translation from valuation to ranking does not always hold in our setting of traffic attraction: announcing an outgoing path with low valuation could be preferred because it brings incoming traffic from attractees. For example, in Figure 2.1, node b's valuation

[3]This is a slight abuse of notation, because r is formally defined on paths and v on outcomes. We ignore this formality from now on.

[4]Some previous work [36, 87, 35, 38, 37] allowed utilities that depend on monetary transfers, which we do not consider here.

function ranks path bQd over path bRd; but, b has higher utility when it claims that it routes on bRd because it then attracts traffic from node a.

Although this direct translation does not always hold, we do assume that BGP-compliant ASes are able to "compile" their utility functions (which depend on both valuation and attraction as in Equation 2.1) into ranking and export functions that then consistently determine their actions in the game, *i.e.,* their behavior during the BGP protocol. This compilation might be viewed as transforming utilities into functions that act on path announcements by, *e.g.,* setting BGP local preference. We think of the compilation process as being done "once and for all," and we analyze the network with respect to fixed ranking and export functions. We note that this is not entirely realistic: the "compilation" can, in principle, model an ongoing process in which an AS reacts to changes in network conditions, contractual agreements, new information that ASes learn about each other, *etc.*, to better attempt to maximize its utility. However, the time scale for compilation is usually much longer than the time scale for BGP itself (say, hours versus seconds); so, a once-and-for-all modeling may still be reasonable. (See also Section 2.7.)

There are many conceivable ways of compiling the utility into ranking and export rules. In many cases, it makes sense to use the simple compilation $r_b(\cdot) \equiv v_b(\cdot)$ by default, and to use a different compilation only when this is advantageous in terms of traffic attraction; *e.g.,* if there is a service-level agreement that obliges b to carry a's traffic via path bRd in return for monetary compensation α, then b might decide to set $r_b(bRd) = v_b(bRd) + \alpha$. In general, we mostly sidestep the question of how to compile the utility into ranking and export policy. However, our counterexamples work for any ranking function "reasonably compiled" from the utility function, and our positive results all hold for the setting $r_b(\cdot) \equiv v_b(\cdot)$.

2.2.6 Incentives to lie.

Because nodes are rational (*i.e.,* acting to maximize their utility in the global outcome), they may have an incentive to follow a strategy that is not BGP-compliant. As discussed in Section 2.1.1, although an AS knows the outgoing link on which it forwards traffic (and the next AS at the end of that link), it may not know the AS-path that the traffic takes further downstream. For example, in Figure 2.1, node b could deviate from BGP-compliance by announcing the path bRd in order to attract traffic from node a, while actually sending traffic over the path bQd; as a result the control and data planes would not match, unbeknownst to a.

Hence, in this work, as in [87, 73, 39, 37], we address the following high-level question: Are there sufficient conditions on the network that ensure that all nodes are honest (*i.e.,* use BGP-compliant strategies)? The earlier work studied this question using the game-theoretic notion of "incentive compatibility." In contrast to some uses of this notion in earlier work (*e.g.,* Thm. 3.2 in [73]), our positive results give nodes some additional flexibility in choosing their strategies, as long as these strategies are BGP-compliant. (We discuss this difference in some detail in Appendix A.1.)

Ideally, we would like conditions that ensure that nodes have no incentive to be dishonest, no matter what the other nodes do. Unfortunately, it is extremely difficult to find such conditions; see [87, 73, 39, 37]. Instead, we look for conditions that ensure that a node has no incentive to be dishonest *if it knows that everyone else is honest.* That is, we try to ensure that no node has an incentive to *unilaterally* deviate from using BGP-compliant strategies.

We discuss our technical formalizations after each of our positive results (Theorems 2.4.1, 2.5.1, and 2.6.1).

2.2.7 Additional remarks.

Modeling nefarious ASes. Our modeling assumes that $v_b(T) = -\infty$ implies $u_b(T) = -\infty$, so that nodes cannot derive any utility from outcomes in which they cannot reach the destination. Our negative examples do not depend on this assumption, but our positive results do. This means that our positive results do *not* hold if a manipulating node wants to attract traffic for nefarious purposes, like tampering or eavesdropping, *when it does not have a path to the destination.*

Single outgoing link. While we assume that all BGP-compliant ASes choose a single outgoing link for all their traffic, a misbehaving node m might send its outgoing traffic on more than one outgoing link. In this case, we assume that if m uses more than one path to d in T, then the valuation $v_m(T)$ is at most as high as the most valuable simple m-to-d path in the outcome T. This assumption was implicitly used in prior work, and it ensures that even for a manipulator m "the optimal strategy" is to send its outgoing traffic over a single link. This is because the valuation of the path cannot decrease if it uses only the "best outgoing link" instead of using a few of them, and the attraction function does not depend on the outgoing links that m uses.

Utility and outcomes. In this work we defined the utility function to depend on the data-plane component of outcome alone, because the control-plane component may not correspond to actual traffic flow in the network. However, this also means that an AS may be unaware of its actual utility (i.e., when its data-plane forwarding path differs from the control-plane path). An alternative approach would be to define the attraction function on the data-plane outcome and the valuation component on the control-plane outcome.

We note, however, that because in this work we consider only *unilateral* deviations (*i.e.,* the all nodes are honest except for a single manipulator), our results in this work hold just the same under this alternative approach. Since we suppose only one node can potentially deviate from honest behavior, we are assured that the data-plane forwarding path of the manipulator matches its control-plane path (since all the nodes on the manipulator's outgoing path must be honest), and so the manipulator utility can depend on either the control-plane or data-plane outcome.

2.3 Definitions: Policy and Export

2.3.1 No dispute wheel.

Griffin, Shepherd, and Wilfong [53] described a global condition on the routing policies in the AS graph, called *"no dispute wheel,"* that ensures that BGP always converges to a unique stable outcome. Roughly, a dispute wheel is a set of nodes, each of which prefers to route through the others rather than directly to the destination. More formally, there is a dispute wheel in the valuations if there exist nodes $n_1, \ldots, n_t$ such that, for each node n_i, there exists a simple path Q_i from n_i to the destination d and a simple path R_i from n_i to n_{i+1} for which $v_{n_i}(R_iQ_{i+1}) > v_{n_i}(Q_i)$.[5] (The index i is taken modulo t.) A dispute-wheel in the ranking functions (for BGP-compliant nodes) is defined similarly with r_{n_i} replacing v_{n_i}. Following the literature [39, 73], we *always* consider networks with no dispute wheels in the valuations. The result of [53] in our terminology states that, if all nodes use BGP-compliant strategies with $r_n(\cdot) \equiv v_n(\cdot)$ and there is no dispute wheel in the valuations, then the game's outcome is unique and globally stable.

[5] For readability, we somewhat abuse notation and use $v_n(P)$ to mean n's valuation of any outcome T in which its traffic uses the data-plane path P.

2.3.2 Policy consistency and next-hop policy.

Node a is *policy consistent* [37,39] in valuations with one of its neighbors b if, whenever b prefers some path bPd over bRd (and neither path goes through a), then a prefers $abPd$ over $abRd$. Formally, for any two simple paths $abPd$ and $abRd$, if $v_b(bPd) \geq v_b(bRd)$, then $v_a(abPd) \geq v_a(abRd)$. We say that *policy consistency* holds for the problem instance if every node is policy consistent with each of its neighbors. (Policy consistency is a generalization of next-hop routing and shortest-path routing; see [37, 39].)

Next-hop policy requires that a node only care about the neighbor through which its traffic is routed and nothing else. This class of routing policies is more restrictive than policy consistency (*e.g.,* node c in Figure 2.3 is policy consistent but does *not* use next-hop policy with node m). Formally, a uses next-hop policy with b if for every two simple paths $abPd$ and $abRd$ it holds that $v_a(abPd) = v_a(abRd)$. Notice that if a uses next-hop policy with b then it must either admit all simple paths through b or (ingress) filter all of them (*c.f.,* discussion in [101, 34]).

Similar definitions apply also to the ranking functions.

2.3.3 Gao-Rexford & customer attractions.

Gao and Rexford [42] described a set of conditions that are induced by business relationships between ASes [58]. In Gao-Rexford networks there are two kinds of edges: customer-provider edges (where typically the customer pays the provider for connectivity) and peer-to-peer edges (where two nodes agree to transit each other's traffic for free). A Gao-Rexford network obeys the following three conditions (GR1–GR3):

GR1. Topology. There are no customer-provider cycles in the AS graph, *i.e.,* no node is its own indirect customer.

GR2. Export. A node b only exports to node a paths through node c if at least one of nodes a and c are customers of node b.

GR3. Preferences. Nodes prefer outgoing paths where the next hop is a customer over outgoing paths where the next hop is a peer or a provider, and prefer peer links over provider links.[6]

GR3 always applies to the valuation functions of each node in a Gao-Rexford network, and can also apply to the ranking functions.

We also model customer attractions within the Gao-Rexford setting. Namely, we consider a fourth condition (AT4) that models the fact that service contracts in the Internet are made between pairs of neighboring nodes, where a customer pays its provider when it sends traffic over their shared link [58]. AT4 restricts the set of traffic attraction relationships that we allow in the AS graph, and thus does *not* model settings where, *e.g.,* an AS wants to attract traffic from ASes that are a few hops away.

AT4. Attractions. A node b may only have attraction relationships with its own customers. Furthermore, b only increases its utility if its attractee-customer a sends traffic over the direct a-b link.

When we draw Gao-Rexford networks, we represent a customer-provider relationship by a directed edge from customer to provider, and a peer-to-peer relationship by an undirected edge. We represent an AT4 attraction relationship with a **bold** arrow from attractee to attractor (*e.g.,* see Figure 2.2).

[6] The original version [42] of the Gao-Rexford conditions does not require nodes to prefer peer links over provider links. To make our results as general as possible, we use this weaker version of GR3 in all our theorems, while our counterexamples do satisfy the stronger version of GR3.

Verification?	Policy	Export	Incentive to Lie?	Result
$\star$	No restriction	$\star$	Yes	Inconsistent Policy
None / Loop	Consistent	$\star$	Yes	Nonexistent Path
Path / Loop	Next-hop	Inconsistent	Yes	Inconsistent Export
Path	Consistent	Consistent	No	Theorem 2.4.1
$\star$	Next-hop	All-or-nothing	No	Theorem 2.4.1

Table 2.2: Summary of our results for traffic-volume attractions. We also require no dispute wheel.

2.3.4 Export rules.

Our results about BGP-compliant strategies that achieve matching control and data planes in the setting of traffic attraction involve several types of export rules. The export-all rule (used, *e.g.,* in Thm. 3.2 of [73]) requires that a node exports all its admitted paths to all its neighbors. An all-or-nothing rule for a node n means that, for each neighbor a of n, either n exports all admitted paths to a or none at all. The consistent export rule [37] means that, if n exports to a neighbor a some path R, then it must also export every other path that is ranked at least as high as R; *i.e.,* if $r_n(Q) \geq r_n(R)$ and n exports R to a, then n must also export Q to a. Finally, in Gao-Rexford networks, the export rules used by BGP-compliant nodes satisfy GR2.

The export-all rule implies the all-or-nothing export rule, which in turn implies the consistent export rule. We emphasize that both the export-all and the all-or-nothing rules are often incompatible with the Gao-Rexford export condition GR2. As one example, the export-all rule may require an AS to export a path through one of its peers or providers to another one of its peers or providers, a violation of GR2.

2.3.5 Dispute wheels in Gao-Rexford networks.

As we discussed in Section 2.3.1, in this work we always consider AS-graphs with no dispute wheel in the *valuation* functions, *even if they obey the Gao-Rexford conditions.* Since in our model, export policy is part of the strategy from which nodes may deviate, we do not rely on GR2 to exclude paths from the valuation functions that may have caused dispute wheels; the valuation functions are only subject to GR1 and GR3. This is in contrast to other works on BGP convergence, *e.g.,* [42,41], which relied on GR2 to remove dispute wheels, because they assumed that every node honestly follows the GR2 export rule. More generally, in the setting where nodes may deviate from (prescribed) BGP-compliant strategies in order to better their own utility, we cannot say that the Gao-Rexford conditions imply that the BGP protocol converges, as in [42,41]. For example, it is possible to show a network in which a node unilaterally deviates from GR2 and thus causes the BGP protocol to oscillate forever. We discuss this further in Section 2.6.5.

2.4 Results: Volume Attractions

We start with some results for traffic-volume attractions, as defined in Section 2.2.3. We stress that this is a rather restricted form of traffic attraction, as it excludes the possibility of the utility depending on the path along which incoming traffic arrives. We begin with a series of counterexamples, demonstrating that even for this very restricted form of traffic attraction, ensuring that nodes have no incentive to lie is far from easy. (Most of our counterexamples

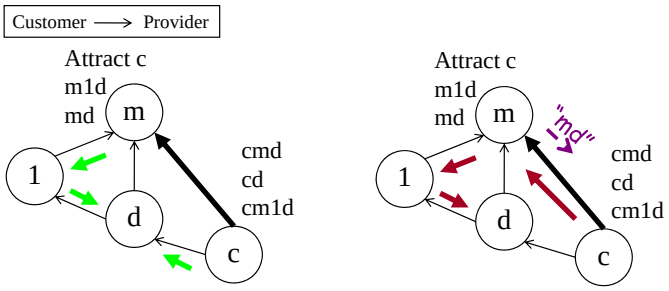


Figure 2.2: Inconsistent Policy

are Gao-Rexford networks that obey GR1–GR3 and sometimes also AT4 from Section 2.3.3.) We then present a positive result (Section 2.4.3), showing two sets of conditions, each of which suffices to ensure that a node honestly announces paths. The results from this section are summarized in Table 2.2.

2.4.1 Path verification is not enough.

Path Verification is the focus of most traditional work on securing BGP [21]; roughly, it ensures that nodes cannot announce paths that are not in the network. More formally, path verification is a control-plane mechanism that ensures that every node a only announces a path abP to its neighbors if its neighbor b announced the path bP to a. Path verification can be guaranteed when S-BGP [66] or IRV [52] is *fully* deployed in the network. (We note, however, that soBGP [106] does *not* provide path verification; soBGP only provides information about AS-graph topology, and not about path announcements.)

For the setting of no traffic attraction, a recent result of Levin *et al.* [73] shows that, in a network with path verification and no dispute wheel, no node has an incentive to unilaterally deviate from a BGP-compliant strategy with $r_n(\cdot) \equiv v_n(\cdot)$ and an export-all rule. They also show (in [72]) that the same is true in Gao-Rexford networks, but with an export rule that exports all paths except those that would violate GR2. However, we show that when there are traffic-volume attractions, a node can have an incentive to make a dishonest announcement, even when the network has path verification:

Figure 2.2: Inconsistent Policy demonstrates that a policy inconsistency between a manipulator m and its customer c can give m an incentive to dishonestly announce its forwarding path in order to attract traffic from c. On the left we show the outcome T that results when each node n uses a BGP-compliant strategy with $r_n(\cdot) \equiv v_n(\cdot)$, exporting all paths except those that would violate GR2. On the right, we show the manipulated outcome M, in which only a single manipulator node m does not use a BGP-compliant strategy. Here, m has an incentive to announce the path md to node c, while actually using path $m1d$, in order to attract c's traffic. Notice that this announcement can be made even with path verification, because node 1 announced $1d$ to m. In the outcome M, node m gains not only a traffic-volume attraction (because c routes through m in M but not in T), but also an AT4 attraction (because c is a customer that routes on the direct c-m link in M). Note that Inconsistent Policy is a Gao-Rexford network with no dispute wheel that obeys AT4.

We remark that the situation in Inconsistent Policy could arise quite naturally in practice. As an example, while c is a customer of both m and d, the service contracts of c with m and d are such that usage-based billing on the m-c link is lower than billing on the d-c link.

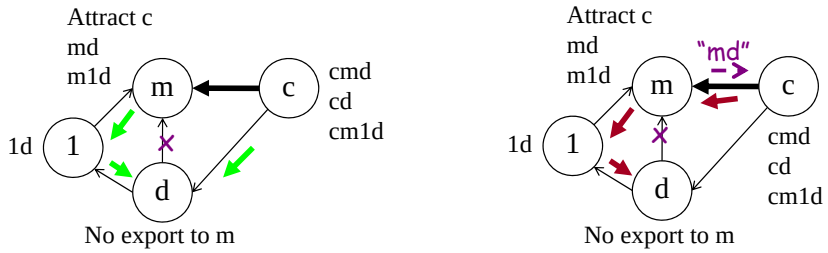


Figure 2.3: NONEXISTENT PATH

Then, c could prefer a path through m over the direct path to d as long as this path only increases AS-path length by a single hop. On the other hand, m could prefer to send traffic via 1 because 1 is, say, geographically closer to m than d.

2.4.2 Policy consistency alone is not enough.

Notice that, in INCONSISTENT POLICY, node c is not policy consistent with node m (Section 2.3.2). It is natural to ask if requiring policy consistency is sufficient to ensure that there is no incentive to lie. Indeed, for the setting of no traffic attraction, Feigenbaum *et al.* [39, 37] proved that in a network with policy consistency and no dispute wheel, then no node has an incentive to *unilaterally* deviate from a BGP-compliant strategy with $r_n(\cdot) \equiv v_n(\cdot)$ and consistent export. Perhaps surprisingly, it turns out that policy consistency is not sufficient to ensure that nodes have no incentive to lie when we consider traffic-volume attractions:

Figure 2.3: NONEXISTENT PATH demonstrates that, even in a policy consistent network, a manipulator m can have an incentive to announce a nonexistent path in order to attract traffic from its customer c. The outcome T, shown on the left, results when each node uses a BGP-compliant strategy with $r_n(\cdot) \equiv v_n(\cdot)$, where node d's export rule obeys consistent export but exports nothing to node m, and all other nodes export all paths allowed by GR2 (which implies consistent export). On the right, we show the manipulated outcome M, where only the manipulator m deviates from the BGP-compliant strategies described above. Here, the manipulator m has an incentive to announce to node c a false path "md" that *is not available to* m (because d does not export this path to m) in order to attract c's traffic. Again, node m gains both a traffic-volume attraction and an AT4 attraction in M that it could not have obtained by using a BGP-compliant strategy. Note that NONEXISTENT PATH is a policy-consistent Gao-Rexford network with no dispute wheel that obeys AT4.

Notice that c has the same preferences in both NONEXISTENT PATH and INCONSISTENT POLICY. However, in NONEXISTENT PATH, c is policy consistent with m; both prefer the nonexistent shorter path through md over the longer path through $m1d$.

2.4.3 But adding path verification or next-hop policy is enough!

In NONEXISTENT PATH, the manipulator m announces a path "md" was that was not announced to it by d (which would not be possible if the network had path verification), and that announcement matters because node c does not use a next-hop policy with m. It turns out that requiring *either* path verification (on top of policy consistency) *or* next-hop policies is sufficient to ensure honesty in any network with only traffic-volume attraction functions. In these settings, if each node sets its ranking equal to its valuation and honestly exports *all* paths to all neighbors, then no node has an incentive to *unilaterally* deviate from this behavior.

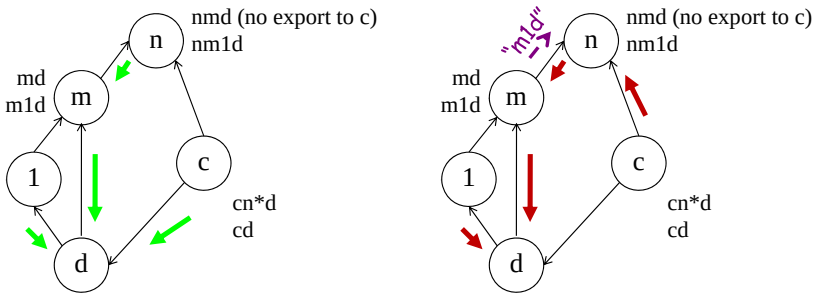


Figure 2.4: Inconsistent Export

Theorem 2.4.1. *Consider an AS graph with no dispute wheel in the valuations. Suppose that all nodes, except a single manipulator node m, use BGP-compliant strategies and set their ranking equal to their valuations ($r_n(\cdot) \equiv v_n(\cdot)$ for every node n). Suppose further that m has a traffic-volume attraction function, and that at least one of the following two conditions hold:*

a. The valuations function of all nodes are next-hop and the export functions of all the nodes but m obey all-or-nothing export; or

b. The valuations function of all nodes are policy consistent, the export functions of all the nodes but m obey consistent export, and the network has path verification.

Then there is a BGP-compliant strategy for m that sets $r_m(\cdot) \equiv v_m(\cdot)$ and obeys all-or-nothing export (and therefore also consistent export), such that this strategy is optimal (utility-maximizing) for m. In particular, using the export-all rule is one such optimal strategy.

Notice that Theorem 2.4.1 not only establishes the *existence* of an optimal consistent export rule for m, but also asserts that export-all is one such optimal rule. Hence it actually establishes a single strategy from which no node has an incentive to deviate. This notion of a single strategy is the same notion used in prior works including [39,37,73,87]. In the mechanism-design literature, this is called *incentive-compatibility in ex-post Nash equilibrium*; see [87] and Appendix A.1. We also comment that in a setting with path verification, the result is slightly stronger since it only requires that honest nodes use consistent export. (We do not know if consistent export suffices for the next-hop result.) The proof of Theorem 2.4.1 is presented in Appendix A.3, and makes heavy use of the result of Feigenbaum *et al.* [39,37].

2.4.4 Our results need consistent export.

Theorems 2.4.1 required a consistent export rule. We now show that we cannot drop this requirement, by presenting a counterexample that obeys all the conditions in Theorem 2.4.1 (policy consistency, next-hop policy, path verification) except consistent export, where node m still has an incentive to lie about its forwarding path in order to gain a traffic-volume attraction:

Figure 2.4: Inconsistent Export demonstrates that m can have an incentive to lie about its forwarding path in order to attract indirect traffic from node c, by taking advantage of the fact that some other node (n) does not use consistent export. Suppose that all nodes *except for* n use export-all rule (which implies consistent export). Now suppose that node n uses an *inconsistent* export rule; it exports the path $nm1d$ to node c, but not the more preferred path nmd. On the left we show the outcome T that results when all nodes use a BGP-compliant strategy with

Verification?	Policy	Export	Incentive to Lie?	Result
None	$\star$	$\star$	Yes	False Loop
$\star$	Consistent	$\star$	Yes	Bowtie
$\star$	Next-Hop	Consistent	Yes	Grandma
Path / Loop	Next-Hop	All-or-Nothing	No	Theorem 2.5.1

Table 2.3: Summary of our results for generic attractions. We also require no dispute wheel.

$r_n(\cdot) \equiv v_n(\cdot)$ and the export rules described above. In T, nodes m and n use the path nmd, but because n does not export this path to c, c routes directly to d. The manipulated outcome M is shown on the right, where only node m deviates from the BGP-compliant strategies described above. By announcing the false path "$m1d$", m manages to attract traffic from c, since now n is willing to export the path "$nm1d$" to node c. Notice that this false path can be announced even if the network has path verification, since node 1 announced "$1d$" to m. (Note that Inconsistent Export is a Gao-Rexford network that *does not obey AT4*, where there is no dispute wheel and all nodes use next-hop policy.)

The reader might object to the fact that in Inconsistent Export, node c prefers the long path $cnm1d$ over the short path cd. We note that this counterexample holds even we lengthen the cd path (say by replacing the c-d link by a path through four additional nodes). On the other hand, we agree that the inconsistent export rule used by node n is somewhat bizarre. Indeed, we believe that it is reasonable to require consistent export in a network that is already policy consistent.

2.5 Results: Generic Attractions

We now consider our most general notion of traffic attraction, in which the utility that nodes derive from attracting traffic can depend arbitrarily on the path that incoming traffic takes (see Section 2.2.3). For this general case, we show in Section 2.5.4 that nodes have no incentive to lie when all nodes use next-hop policy and all-or-nothing export and the network has path verification. (In fact, we show that a weaker enforcement mechanism called *loop verification* is also sufficient; see Section 2.5.3.) These conditions are extremely strong, but we show via a sequence of counterexamples that we cannot drop any one of these conditions without allowing an incentive to lie. The theorems and counterexamples in this section are summarized in Table 2.3.

2.5.1 Policy consistency & path verification is not enough.

In networks with only traffic-volume attraction, we were able to show that adding path verification to a policy-consistent AS graph is sufficient to ensure that nodes have no incentive to lie (Section 2.4.3). Unfortunately, this is not the case when we consider more general attraction relationships:

Figure 2.5: Bowtie demonstrates that, even in a network that is policy consistent and has path verification, a manipulator m can have an incentive to lie about its forwarding path in order attract traffic from a customer c on the direct m-c link. Suppose node m has an attraction function such that (1) m has an AT4 attraction relationship with its customer c, and (2) m has a traffic-volume attraction with its provider n. The outcome T that results when every node uses a BGP-compliant strategy with $r_n(\cdot) \equiv v_n(\cdot)$ and exports all paths allowed by GR2, is shown on the left. The manipulated outcome M is shown on the right, where only node m deviates from the BGP-compliant strategy we described above.

Here, m has an incentive to dishonestly announce the path "$m1d$" to all of its neighbors in order to attract traffic from the attractee c on the direct c-m link. Node m can make

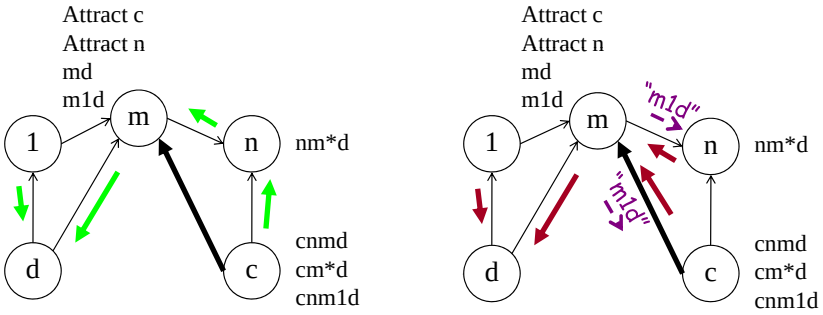


Figure 2.5: BOWTIE

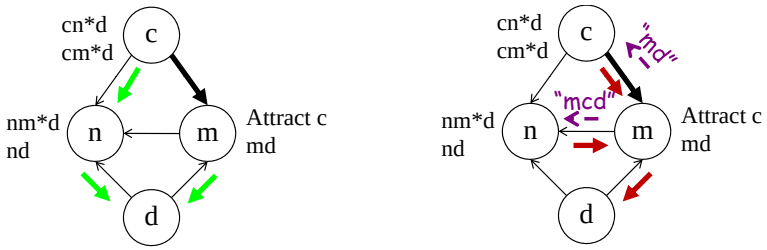


Figure 2.6: FALSE LOOP

this announcement, even with path verification, because node 1 announced the path $1d$ to m. Moreover, there is no BGP-compliant strategy for m that allows it to attract traffic from both c and n while maintaining its preferred data-plane forwarding path md. Note that BOWTIE is a policy-consistent, Gao-Rexford network with path verification *that does not obey AT4* and has no dispute wheel in the valuations.

We remark that even though c's traffic is routed via m in both T and M (*i.e.,* m does *not* gain a traffic-volume attraction), the manipulation in BOWTIE is quite reasonable in practice. For example, m might prefer the outcome in M over the outcome in T for load-balancing purposes, because incoming traffic from c and n is spread over two links in M. As another example, m might prefer the outcome M because it has a usage-based billing contract with c on the m-c link, whereas node m is not able to bill its provider n for carrying c's traffic (which occurs in outcome T).

2.5.2 Next-hop policy alone is not enough.

From BOWTIE, we learn that policy consistency is *not* sufficient to ensure honest announcements (even when using path verification). So we throw up our hands and ask if it suffices to require that every node uses next-hop policy. With next-hop policy, it is tempting to conclude that lying about an outgoing path will not help an attractor convince an attractee to 'change its mind' and route through it in a manipulated outcome. (Notice that the manipulations in INCONSISTENT POLICY, NONEXISTENT PATH and BOWTIE were of this form.) Furthermore, next-hop policy is sufficient when considering only traffic-volume attractions (Section 2.4.3).

Quite surprisingly, this intuition fails. We now present our most important counterexample, which shows that if the network does not have path verification, then even requiring next-hop policy is not sufficient:

Figure 2.6: FALSE LOOP demonstrates that, even in a network where all nodes use next-hop

policies, a manipulator m can gain traffic from its customer c by falsely announcing a path through c to m's other neighbors. Suppose that m announces no paths to neighbor n and all paths to everyone else, and that all other nodes export all paths allowed by GR2. On the left is the outcome T, where each node compiles $r_n(\cdot) \equiv v_n(\cdot)$ and uses the BGP-compliant strategy with the export rules described above. The manipulated outcome M is on the right, where only m deviates from the BGP-compliant strategy above. In M, the manipulator m has an incentive to announce a false outgoing path "mcd" to n in order to attract traffic from its attractee c (on the direct c-m link). Notice that the outcome M results whenever there is no control-plane verification mechanism such as path verification, since the 'false loop' "$nmcd$" will either cause node n not to announce any path to node c, or instead cause node c to ignore the announcement. Also, m has no BGP-compliant strategy that allows it to gain an AT4 attraction from c, since c would have sent his traffic on the c-n link if m had either (a) honestly announced some path to n, or (b) announced no path to n (as in outcome T). Note that False Loop is a Gao-Rexford network with no dispute wheel that obeys AT4, in which all nodes use next-hop policies.

2.5.3 Introducing loop verification.

To deal with the manipulation in False Loop, we introduce loop verification, a new control-plane mechanism that deals with detecting and preventing "false loops."

BGP allows two different approaches for detecting and preventing routing loops. One is sender-side loop detection, where a node a will *not* announce path aRd to node b if b happens to be on the path R. The other is receiver-side loop detection where a *will* announce the path aRd to b, so that b will detect the loop and discard that announcement. Receiver-side loop detection has the advantage of allowing a node b to hear announcements that *falsely* include a path that b did not announce. Notice that for b to detect a "false loop," b need only perform a *local* check to see if the path it receives matches the one that b actually announced. (This local check is less onerous than the one that is required for path verification, which requires participation from all ASes on the path.)

Loop verification encourages ASes to avoid lying in BGP announcements because they should fear getting caught. We define loop verification as the use of receiver-side loop detection by *all* nodes in a network, with the additional requirement that when node b receives an announcement of a path $P = QbRd$, such that b did not announce the path bRd to its neighbors, then b "raises an alarm." Then, the first node who announced a path that includes bRd will be punished with utility reduced to $-\infty$. This punishment process models the idea that b can catch and shame the node that announced the false loop, *e.g.,* via the NANOG list.

The properties of loop verification are strictly weaker than those of path verification. Namely, if a network has path verification, then no node will raise an alarm in loop verification. This follows from the fact no node can announce a path that includes bRd unless b announces the path bRd.

2.5.4 Next-hop policies & loop verification is enough!

Now that we defined loop verification, we are ready to present the main result of this section. If we add loop verification to a next-hop network with no dispute wheel, we can eliminate the manipulation performed by m in False Loop. We also require all nodes to use an all-or-nothing export rule. The following holds even if the network does *not* obey the Gao-Rexford conditions:

Theorem 2.5.1. *Consider an AS graph where the valuation functions are next-hop and contain no dispute wheel. Suppose that all nodes, except a single manipulator node m, use BGP-compliant strategies where they set their ranking equal to their valuations ($r_n(\cdot) \equiv v_n(\cdot)$ for*

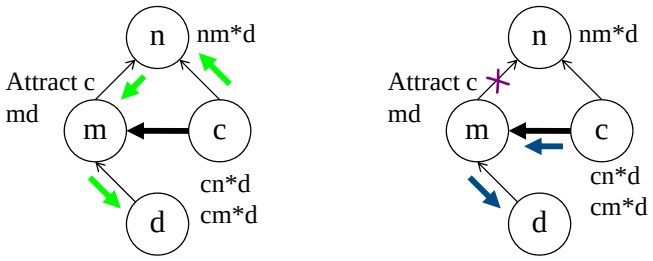


Figure 2.7: Access Denied.

every node n), and obey all-or-nothing export. Suppose further that the network uses either loop verification or path verification. Then there exists a BGP compliant strategy for m that uses $r_m(\cdot) \equiv v_m(\cdot)$ and obeys all-or-nothing export, which obtains the best possible stable outcome *in terms of the utility function of m.*

On an intuitive level, Theorem 2.5.1 proves that any gains a manipulator gets from lying can be obtained by using a clever export rule.[7] That is, Theorem 2.5.1 shows the *existence* of an optimal all-or-nothing export rule for the manipulator; however, this optimal export rule for m depends on the export rules chosen by the other nodes in the network. Furthermore, unlike prior work or the result from Section 2.4, this result does *not* explicitly describe this optimal export rule.

The proof of Theorem 2.5.1 is quite technically involved, so we present it in Appendix A.4. Roughly, the proof amounts to showing that when all nodes use next-hop policy with their neighbors, the only strategically useful lie available to the manipulator is to announce a false loop. Then, we show that if the network has loop verification, some node detects the false loop and punishes the manipulator for its lie; since the utility of the manipulator drops down to $-\infty$ when it gets caught, it no longer has an incentive to announce a false loop, and the theorem follows.

2.5.5 Export-all is not always optimal.

Theorem 2.5.1 unfortunately does *not* explicitly describe the optimal export rule for the manipulator. We now show that the export-all rule (which was shown to be optimal in *e.g.,* Theorem 2.4.1 and [73]) is not necessarily optimal in this setting:

Figure 2.7: Access Denied demonstrates that m can attract traffic from its customer c over the direct m-c link by denying export to some of m's other neighbors. Here, the network has path and loop verification, next-hop policies at every node, and m is interested in attracting traffic only from c (but not from n) in an AT4 attraction. Suppose that all nodes, including m, honestly announce paths. On the left we present the outcome when every node, including m, uses export-all. On the right, we illustrate the outcome when m uses a different all-or-nothing export rule: in particular, m announces all paths (honestly) to c, and no paths to n. As a result, m attracts traffic from c on the direct c-m link. If m had announced paths to n, then c would not have sent its traffic on the c-m link, as in the outcome on the left. Thus, we see that the export-all rule is not optimal for m. Note that Access Denied is a network that obeys GR1, GR3, and AT4, and has no dispute wheel.

[7] We remark that this result only rules out the possibility of obtaining a better *stable* outcome by lying, it does not rule out the possibility of m gaining utility by inducing a non-stable outcome. See Section 2.2.2.

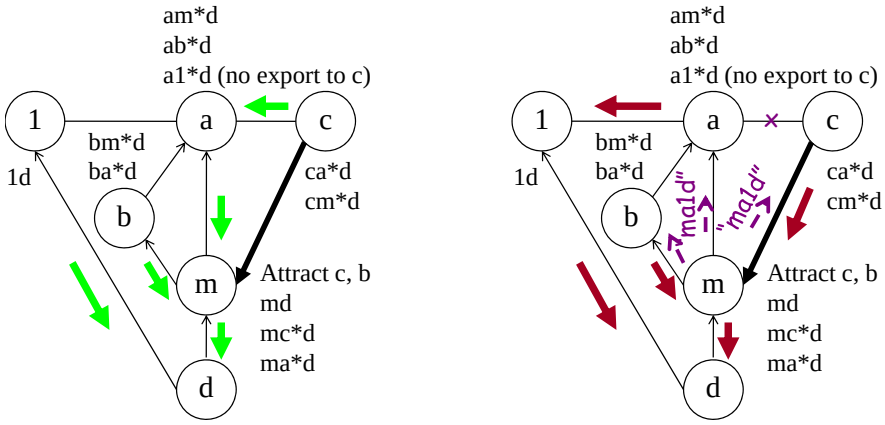


Figure 2.8: GRANDMA.

We pause here to observe that in the outcome on the right, n has no path to the destination if node c only exports the paths allowed by GR2. We discuss this issue in Section 2.6.4.

2.5.6 Theorem 2.5.1 needs all-or-nothing export.

The requirement that all nodes use an all-or-nothing export policy in Theorem 2.5.1 is extremely strong, especially because most networks that obey the Gao-Rexford conditions (in particular GR2) violate this export rule. We now present our most devastating (and complicated) counterexample that shows Theorem 2.5.1 does not hold with a more realistic export rule like consistent export:

Figure 2.8: GRANDMA demonstrates that a manipulator m can have an incentive to lie in order to attract traffic from a customer c if some other node a does not use an all-or-nothing export policy. Furthermore, GRANDMA shows that this is possible even when all nodes use path verification and next-hop policies.

In GRANDMA, m has an AT4 attraction relationship with its customer c, a traffic-volume attraction relationship with its provider b, and no other attractions. Suppose now that all nodes export all paths allowed by GR2; thus, a does *not* export paths through its peer 1 to its peer c. While a uses a consistent export rule (since a filters only its lowest ranked path through 1), a does not use all-or-nothing export rule. On the left is the outcome T that results when all nodes act honestly, *i.e.,* use BGP-compliant strategies with $r_n(\cdot) \equiv v_n(\cdot)$ and the export rules above. The manipulated outcome M is shown on the right, where only the manipulator m deviates from the BGP-compliant strategies above.

In M, the manipulator m dishonestly announces the path "$ma1d$" while actually routing on md. To arrive at the outcome M on the right, node m sits quietly until node a exports "$a1d$" to it. Then m announces "$ma1d$" to all nodes, while routing on md in the data plane. Node a cannot route through m (because it thinks that m routes through it); so, a continues to route on $a1d$. Next, because a does not export paths through 1 to its peer node c, node c has no choice but to route through node m. Meanwhile, m's machinations have no effect on b, who routes through m regardless. Notice that loop or path verification would not help, since node a is indeed routing along "$a1d$". Furthermore, m manages to retain in M its traffic-volume attraction with b and gain an AT4 attraction with customer c. Also, m has no BGP-compliant strategy that obtains as large a utility as it obtains from M. Note that GRANDMA is a Gao-Rexford network with no dispute wheel *that does not obey AT4*, where all nodes use next-hop

AT4	Verification	Policy Consist.	Next-hop policy	Export	Incentive to Lie?	Result
No	$\star$	$\star$	$\star$	Consist.	Yes	GRANDMA
Yes	None	$\star$	$\star$	$\star$	Yes	FALSE LOOP
Yes	$\star$	None	All nodes w. peers & providers	$\star$	Yes	ORION
Yes	None / Loop	All nodes	None	$\star$	Yes	NONEXISTENT PATH
Yes	Loop / Path	All nodes	Attractees w. peers & providers	Consist.	No	Theorem 2.6.1

Table 2.4: Summary of our results for Gao-Rexford networks (obeying GR1-GR3) with no dispute wheel.

policy with all their neighbors.

2.5.7 The need for ubiquitous participation.

BOWTIE and GRANDMA highlight another important point; namely, that even if one node follows the conditions specified in our theorems, *e.g.,* next-hop policy, it is still possible for that node to learn a false path, if some other node in the network fails to follow the specified conditions. For example, in BOWTIE (Figure 2.5), even though attractee node n uses next-hop policy, n still learns a false path because node c does not. Thus, we emphasize that all the theorems in this paper only hold if *every node* in the network follows the specified set of conditions.

2.6 Results: Customer Attractions in Gao-Rexford Networks

We now focus on Gao-Rexford networks (see Section 2.3.3). In Section 2.5, we used GRANDMA (Figure 2.8) to show that Theorem 2.5.1 does not hold with consistent export in place of the unrealistic all-or-nothing export rule (which is usually not compatible with GR2). Fortunately, GRANDMA did not obey the AT4 attraction condition. Thus, we now weaken the assumption of all-or-nothing export by focusing on the AT4 setting, in which an attractor can increase its utility only if a *customer* routes on the direct link between them. It turns out that AT4 also allows us to weaken the next-hop-policy restrictions required in Theorem 2.5.1. Our results are summarized in Table 2.4, which also shows how dropping any one of the conditions in our positive result (Section 2.6.2) may create an incentive to lie.

2.6.1 It's not sufficient to restrict policy at attractees only.

The requirement in Theorem 2.5.1 that every node in the network uses a next-hop policy with all of its neighbors is very strong indeed. Ideally, we would have preferred to require only *attractees* to use next-hop policy with their attractors. Unfortunately, even requiring every attractee to use next-hop policy with *all its neighbors* may not remove the incentive to lie:

Figure 2.9: ORION is a Gao-Rexford network with no dispute wheel that obeys AT4. In ORION, only the attractee (node c) uses next-hop policy with all its neighbors (nodes m, n). Every other node uses next-hop policy with its peers and providers, but not necessarily with its customers. Notice that node a is not policy consistent with its customer m: node m prefers path $m1d$ to path md (say, because it is cheaper to route directly to 1), while node a prefers the path amd to the path $am1d$ (say, because it prefers shorter paths).

On the left is the outcome T that results when each node uses a BGP-compliant strategy with $r_n(\cdot) \equiv v_n(\cdot)$, exporting all paths allowed by GR2. The manipulated outcome M is shown on the right, where the manipulator m deviates from this BGP-compliant strategy. In the manipulated outcome M, m dishonestly announces the outgoing path "md" to all of its neighbors so that

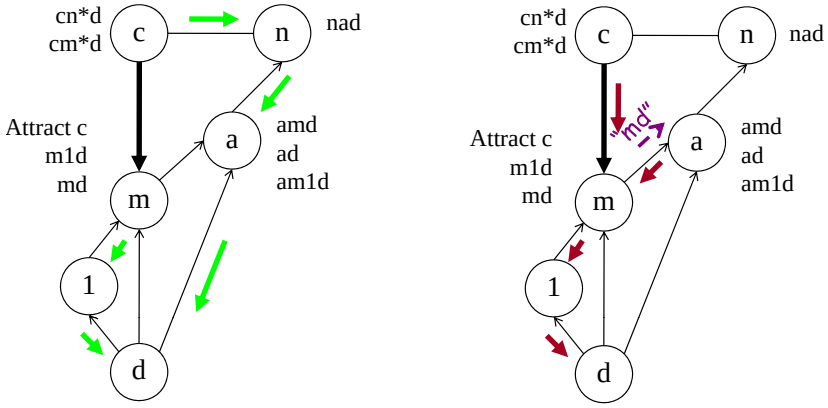


Figure 2.9: ORION.

node a decides to route through m on the *amd* path. However, node n does not admit the path *amd* and thus is left with no path to the destination d. The attractee c has no choice but to route through m, increasing m's utility. Observe that m has no BGP-compliant strategy that obtains as large a utility as it obtains from M.

Notice that n uses a "forbidden-set policy" [35], in which it prefers using no path at all over using a path through m. Such preferences could arise in practice if node n does not trust node m to carry its traffic (say, because it perceives node m to be adversarial).

2.6.2 Policy consistency everywhere with next-hop policy at attractees is enough!

Earlier, we saw that, even in the Gao-Rexford setting with AT4, dropping either path or loop verification may create an incentive to lie (as in FALSE LOOP in Figure 2.6). Furthermore, from ORION above, we learn that policy restrictions only on attractees can leave an incentive to lie. The manipulation in ORION is possible because node a is not policy consistent with node m; we now show that requiring policy consistency, along with other conditions satisfied by ORION, is enough to ensure no incentive to lie.

Theorem 2.6.1. *Consider a policy-consistent, Gao-Rexford network that obeys AT4, in which there is no dispute wheel in the valuations and all attractees use next-hop policies with their providers and peers. Suppose that all nodes, except a single manipulator node m, uses a BGP-compliant strategy with $r_n(\cdot) \equiv v_n(\cdot)$ and a consistent export rule that satisfies GR2. Suppose further that the network has path or loop verification.*

Then there exists a BGP-compliant strategy for m with $r_m(\cdot) \equiv v_m(\cdot)$ and a consistent export rule obeying GR2 that obtains the best possible stable *outcome in terms of the utility function of m. In particular, exporting all paths to customers and no paths to providers and peers is one such optimal strategy.*

The proof, in Appendix A.5, consists of a series of technical arguments that use the Gao-Rexford conditions (GR1-GR3) and AT4 to show that if m can increase its utility in the manipulated outcome, then the network must have a customer-provider loop.

2.6.3 Our result needs next-hop at attractees.

We note that we cannot drop the requirement in Theorem 2.6.1 that all attractees use next-hop policy with all their peers and providers. To see why, recall that a manipulation is possible in NONEXISTENT PATH (Figure 2.3), which satisfies all the conditions of Theorem 2.6.1 (loop verification, policy consistency at all nodes, Gao-Rexford, AT4, no dispute wheel, consistent export) except that the attractee node c does not use next hop policy with its provider m. However, the manipulation in NONEXISTENT PATH would *not* be possible with path verification (instead of loop verification). Thus, in this work we have *not* ruled out the possibility that we can drop the requirement for attractees to use next-hop policy if we replace loop verification with path verification.

2.6.4 It's best to export only to your customers.

Observe that Theorem 2.6.1 not only shows the *existence* of an optimal export rule for the manipulator, but also explicitly describes one such export rule. It therefore provides a specific strategy from which no node has an incentive to unilaterally deviate.[8] However, this strategy requires that *m never announces any paths to its peers and providers.* While this export rule obeys consistent export and GR2, a network in which every node uses this "export-nothing-to-non-customers" rule would be a very sorry network indeed: Peer paths would not exist, and nodes would never transit traffic from their providers, even if that traffic is destined for their customers!

Unfortunately, there are cases in which the optimal export rule for the manipulator is to "export nothing to non-customers." For example, consider ACCESS DENIED in Figure 2.7 and observe that m's optimal strategy is to announce no paths to n (which means that when c's export rule obeys GR2, node n has no path to the destination). Furthermore, this network obeys the strongest conditions considered in this work (next-hop policy at all nodes and path verification). Hence, within the conditions considered here, we cannot hope to get a result where m's optimal export policy necessarily allows it to announce paths to peers and providers.

This suggests that AT4 may *not* be a reasonable model for attraction relationships; *e.g.,* a node could improve its utility by attracting traffic from a provider or peer if it *delivers* this traffic to a customer. Finding a more appropriate model for attraction relationships in Gao-Rexford networks remains open for future research.

2.6.5 Our result needs no dispute wheel.

Notice that in addition to obeying the Gao-Rexford conditions, Theorem 2.6.1 also requires that the valuation functions have no dispute wheel. As we discussed in Section 2.3.3, this means that in addition to obeying GR1 and GR3, the valuation functions must contain no dispute wheel *even without excluding paths that are removed by the GR2 export rule.* This is a very strong requirement indeed, since GR2 often excludes paths from the network that would have created dispute wheels. Ideally, we would like to drop this requirement from Theorem 2.6.1. Unfortunately, this is not possible:

Figure 2.10: DISPUTED PATH demonstrates that, if a network has a dispute wheel, a manipulator m can have an incentive to falsely announce paths in order to attract traffic from a customer c. Furthermore, DISPUTED PATH shows that this is possible even if there is path

[8] However, as in Theorem 2.5.1, we add the disclaimer that this result only applies to *stable* manipulated outcomes.

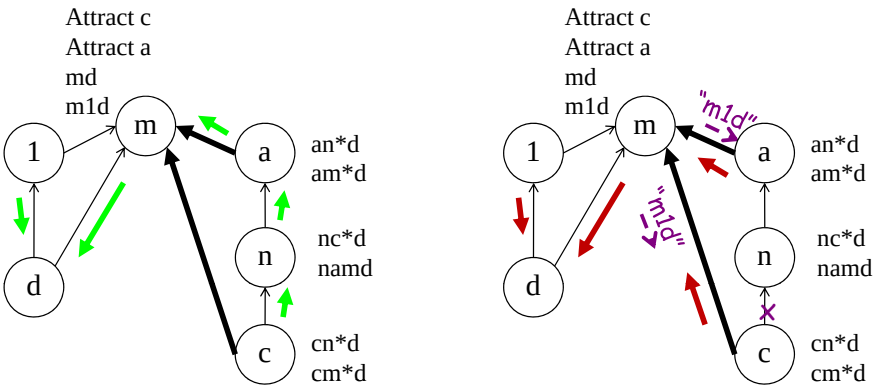


Figure 2.10: DISPUTED PATH.

verification, all nodes are policy consistent, and every attractee (nodes c, a) use next-hop policy with all their neighbors (nodes m, n).

On the left is the outcome T that results when each node uses a BGP-compliant strategy with $r_n(\cdot) \equiv v_n(\cdot)$ and exports all paths that do not violate the GR2 export condition. The manipulated outcome M is shown on the right, where only node m deviates from this strategy. In the manipulated outcome M, m announces a false outgoing path "$m1d$" to all of its neighbors. This is possible even with path verification since 1 announced the path $1d$ to m. Notice that while node n is policy consistent with all his neighbors, he does not admit the path $nm1d$. Furthermore, since c obeys GR2, he does not export any paths to n. As a result, n is left with no path to the destination, and c routes through his attractor m instead. However, the other attractee node a continues to route through m even when m announces this false path. Furthermore, m has no export rule for which he can achieve the same utility that obtained in M. Note that DISPUTE PATH is a Gao-Rexford network where all nodes are policy consistent, every attractee use next-hop policy with all neighbors, and there is path verification. DISPUTED PATH has a dispute wheel between nodes c, n; n prefers paths through its customer c over paths through its provider a, but c prefers paths through its provider n over paths through its provider m.

One way to get rid of the requirement for no dispute wheel is to change our interpretation of the Gao-Rexford conditions. Namely, we could assume instead that paths that are usually excluded by the GR2 export rule are also *not admitted by the valuation function of all nodes*. This means that paths that violate GR2 are filtered on ingress, (rather that filtered on egress, as per Section 2.3.3). This approach is discussed in [73]. (However, we emphasize here that Theorem 2.6.1 does *not* hold under this alternate interpretation of the Gao-Rexford conditions.) While this interpretation may lead to better positive results, it may be unrealistic; for instance, in DISPUTED PATH, node c has no reason to announce the path $cnm1d$ to node n, since both m and n are providers of c and c only stands to lose money by transiting traffic from one provider to another. Thus, it seems reasonable to expect c to refuse to export this path. Meanwhile, n has no reason not to admit the path $ncm1d$, since this path is through his customer c. Furthermore, in practice, business relationships between ASes are often kept private. Thus, it is not clear how n would learn that node m is c's provider, and therefore that node n should not admit the path $ncm1d$.

2.7 Related work

We discussed some related work in Sections 2.1–2.2. Further discussion is below. Griffin, Shepherd, and Wilfong [53] developed a formal model of BGP which assumes ASes choose paths based on an arbitrary preference function that ranks *outgoing paths*. They used this model to initiate a study of sufficient conditions to ensure that BGP converges to a unique outcome (Section 2.3.1). This study was continued by many subsequent works; most relevant here are the results of Gao and Rexford [42] who considered constraints that arise due to business relationships between ASes (Section 2.3.3), and those of Feamster, Johari, and Balakrishnan [34] who studied the effect of filtering (Section 2.3.4).

In contrast to the works on BGP convergence, the game theoretic studies of BGP [39, 73, 35, 36, 37, 38, 87, 30], discussed in Section 2.1.2 and throughout this paper, looked for mechanisms that induce incentives to comply with the protocol (which, in particular, means that ASes would have no incentive to lie). These works interpret the preference function in Griffin *et al.* [53] as a measure of utility for each AS, and model ASes as rational agents who act selfishly to maximize utility. This is equivalent to assuming that utility is uniquely determined by *outgoing paths*. To our knowledge, our work is the first to model the effect of *incoming traffic* on the *incentive* to lie in BGP announcements. Earlier versions of our work appeared as [45] and [65].

Recently, the literature on BGP convergence has begun to model the effect of *incoming traffic* on BGP dynamics. These works [43, 102, 103] focus on the context of traffic engineering, and assume that ASes honestly announce paths; they do not consider ASes that lie. Gao, Dovrolis and Zegura [43] and Wang *et al.* [102] study algorithms for traffic attraction and deflection using AS-path prepending. (Our work does not model prepending.) Wang *et al.* [103] study oscillations that can occur if the BGP decision process depends on incoming traffic as well as outgoing paths. In contrast, our work allows *utility* to depend on incoming traffic (Section 2.2.3) but assumes that the BGP dynamics are based on *ranking* functions (Section 2.2.2) that depend only on outgoing paths. The ranking functions are derived from a "compilation" of the utility function (Section 2.2.5). Thus, in some sense, Wang *et al.* study the oscillations that can result as ASes continuously adjust their compilation. Indeed, Figure 2 of [103] shows conditions under which Inconsistent Policy in our Figure 2.2 could experience such oscillations.

2.8 Conclusions

In this work, we considered control-plane mechanisms that provide incentives for rational ASes to announce their true data-plane paths in BGP messages. We find that conditions previously shown to be sufficient for honesty no longer suffice if we assume that ASes can benefit by attracting incoming traffic from other ASes. We demonstrated that, within the *control-plane* mechanisms we considered here, ensuring honesty in the face of traffic attraction requires very strong restrictions on routing policy (at the very least, policy consistency everywhere, and sometimes also next-hop policy at certain ASes), as well as control-plane verification (loop-verification or path-verification protocols like Secure BGP [66]). Thus, our results suggest that in practice, it will be difficult to achieve honesty without resorting to expensive *data-plane* protocols that verify and enforce AS-level paths. By highlighting the difficulty of matching the control and data planes, even under the assumption that ASes are rational (and not arbitrarily malicious), our results can also help inform decisions about whether security protocols should be deployed in the control plane, in the data plane, or in both.

Chapter 3

Path-Quality Monitoring: Failure Detection

3.1 Introduction

Path-quality monitoring is a crucial component of flexible routing techniques (e.g., intelligent route control, source routing, and overlay routing) that give edge networks greater control over path selection. Monitoring is also necessary to verify that service providers deliver the performance specified in Service-Level Agreements (SLAs). In both applications, edge networks need to determine when path quality degrades beyond some threshold, in order to switch from one path to another or report an SLA violation. The problem is complicated by the presence of nodes along the path who try to interfere with the measurement process, out of greed, malice, or just misconfiguration. In this chapter, we design and analyze light-weight path-quality monitoring (PQM) protocols that detect when packet loss or delay exceeds a threshold, even when adversaries try to bias monitoring results. Our solutions are efficient enough to run at line rate on the high-speed routers connecting edge networks to the Internet.

3.1.1 The presence of adversaries

Today, path-quality monitoring relies on active measurement techniques, like ping and traceroute, that inject special "probe" packets into the network. In addition to imparting extra load on the network, active measurements are vulnerable to adversaries that try to bias the results by treating probe packets preferentially. Instead, we want to design protocols that provide accurate information even when intermediate nodes may *adversarially delay, drop, modify, inject or preferentially treat* packets in order to confound measurement. Our motivations for studying this adversarial threat model are threefold:

1. It covers active attacks. Our strong threat model covers a broad class of malicious behavior. Malicious adversaries can easily launch routing-protocol attacks that draw packets to (or through) a node of their choosing [14], or compromise one of the routers along an existing path through the Internet [57, pg. 14]. Biasing path-quality measurements allows the adversaries to evade detection, while continuing to degrade performance or impersonate the legitimate destination at will. In addition, ISPs have both the economic incentive and the technical means to preferentially handle probe packets, to hide discrimination against unwanted traffic like Skype [85] or BitTorrent [1], and evade detection of SLA violations. (In fact, commercial monitoring services, like Keynote, claim to employ "anti-gaming" techniques to prevent providers from biasing measurement results [3].) Finally, adversaries controlling arbitrary end

hosts (such as botnets) can add "spoofed" packets to the stream of traffic from one edge network to another, to confound simplistic measurement techniques (e.g., such as maintaining a counter of received packets).

2. It covers all possible benign failures. By studying the adversarial setting, we avoid making *ad hoc* assumptions about the nature of failures caused by normal congestion, malfunction or misconfiguration. Even benign modification of packets may take place in a seemingly adversarial manner. For example, an MTU (Maximum Transmission Unit) mismatch may cause a router to drop large packets while continuing to forward the small probe packets sent by ping or traceroute [75]. As another example, link-level CRC checks are surprisingly ineffective at detecting the kinds of errors that corrupt IP packets [98]. Since the adversarial model is the strongest possible model, any protocol that is robust in this setting is automatically robust to all other kind of failures.

3. It is challenging to satisfy in high-speed routers. We choose to work in a difficult space, where we assume the strongest possible adversarial model, and yet design solutions for high-speed routers on multi-Gbit/sec links, where computation and storage resources are extremely limited. We view it as an important research goal to understand what can and cannot be done in this setting, to inform practical decisions about what level of threats future networks should be designed to withstand. Furthermore, designing protocols for this adversarial setting is not simply a matter of adding standard cryptographic tools to existing non-adversarial measurement protocols. Indeed, naive ways of combining such protocols with cryptographic tools may be either insecure or very inefficient (*e.g.,* encrypting and authenticating all traffic).

Despite the strong threat model we consider in this chapter, we are still able to design secure PQM protocols that can be implemented in the constrained environment of high-speed routers. Our protocols are competitive, in terms of efficiency, with solutions designed for the non-adversarial setting [33, 59] and for weaker threat models. As such, we believe that our protocols are strong candidates for deployment in future networks, even where our strong security guarantees may not be essential.

3.1.2 Our results

We say that a *packet delivery failure* (*failure* for short) has occurred on a path if a packet sent by the source was dropped, modified, or delayed beyond a certain timeout period, regardless of whether the drop is due to congestion, malfunction or adversarial behavior. The goal of a PQM protocol is to *detect* when the fraction of failures on a path rises above a certain fraction β (say $\beta = 0.01$) of all packets sent. We emphasize that a PQM protocol does not *prevent* failures. A *secure* PQM protocol achieves its goal even when there is an intermediate node on the path between source and destination that can adversarially drop, modify, or inject both data and protocol-related packets to the path in order to bias the measurement results. Most existing PQM protocols, such as ping, traceroute, and counter-based solutions [99] completely break down in this setting (we show why in Section 3.2.2).

To have efficient solutions that can run on high-speed routers, we design secure PQM protocols based on two main classes of data-reduction techniques:

Secure sketch. In Section 3.5, we present a protocol for monitoring packet-loss rates that makes extremely efficient use of communication and storage resources. Our secure sketch protocol uses ℓ_2*-norm estimation sketches* [7, 5, 24, 100] to aggregate information about the failures that occur during an interval, in which T packets are sent, into a *sketch* of size $O(\log T)$ bits; the communication overhead is just a single report packet per time interval. Assuming that about 10^7 packets are sent during an 100ms interval, our protocol requires between 250–600 bytes of

storage at the source and destination, and a report can easily fit into a single IP packet. In the course of analyzing this protocol, we provide an improved formal analysis of the performance of [24]'s sketching scheme that may be of independent interest.

Secure sampling. In certain settings, an edge-network may require accurate round-trip delay measurements in addition to monitoring if the failure rate rises above a threshold. Section 3.4 describes a secure PQM protocol that achieves this by measuring performance for a sample of the traffic that is obtained using a cryptographic hash function. For PQM with threshold β, this sampling-based protocol requires $O(n/\beta)$ bits of storage at the source, where n is the output length of the hash function. We present two variants: (1) *Symmetric Secure Sampling* is designed for the setting where source and destination can devote an equal amount of resources to the running of the protocol, and (2) *Asymmetric Secure Sampling*, which is designed for a client-server setting where the client contributes the bulk of the resources, and the server participates in path-quality monitoring with many clients simultaneously.

Precise definition of security. Evaluating the security of a protocol is challenging in practice. In many problem domains, *e.g.,* intrusion detection, the only viable approach is to enumerate a set of possible attacks, and then show how the protocol defends against these specific attacks. One way to do this is to evaluate the protocol on, say, packet traces of real-world attacks. However, there is always a risk that an adversary might devise a new attack that we have not considered or that was not expressed in the trace. Fortunately, in our problem domain, a more comprehensive security evaluation is possible. Namely, instead of enumerating ways the protocol can break down (*i.e.,* attacks), we can instead give a precise definition of the functionality we require from the protocol, and then guarantee that the protocol can carry out these functions *even in the face of all possible attacks by an adversary* with a specific set of capabilities.

To do this, in Section 3.2 we precisely define our requirements for a secure PQM protocol and the powers that we give to the adversary. Then, to evaluate the security of our protocols, we use formal analysis to *prove* that our protocols achieve this functionality *no matter what* the adversary does, short of breaking the security of the basic cryptographic primitives (*e.g.,* digital signatures and hash functions) from which the protocol is constructed. In Section 3.6 we prove that *any* secure PQM protocol (as per Definition 3.2.1) would need to employ the same basic security machinery—secret keys and cryptographic operations—used by our secure sketching and sampling protocols.

Evaluating performance. The performance and cost of any particular implementation of our protocols would depend on memory speed and the particular choice of cryptographic primitives. As such, we count separately the different resources—computation, storage and communication—used by our protocols, bound the resource utilization using formal analysis, and also show somewhat better bounds through numerical experiments. Our protocols use cryptographic hash functions in an *online* setting, where an adversary has very limited time to break the security before the hash parameters are refreshed; this allows us to use fast implementations of these hash functions (details in Appendix B.1). We emphasize that all except one of our protocols *do not modify data packets* in any way, and so they may be implemented off the critical packet-processing path in the router. Not marking packets also makes our protocols backwards compatible with IP while minimizing latency at the router, allows the parties to turn on/off PQM protocols without the need to coordinate with each other, and avoids problems with increasing packet size and possibly exceeding the MTU. For efficiency reasons, we specifically avoid solutions that require encryption and authentication of all the traffic sent on the path, as in IPsec. We further discuss and compare the performance trade-offs for our sketch and sampling protocols with known solutions like IPsec in Section 3.7.

3.2 The statistical security model

In our model, a source *Alice* sends packets to a destination *Bob* over a path through the Internet. Fix a set of T consecutive packets sent by Alice, which we call an *interval*, we define a *packet delivery failure* to be any instance where a packet that was sent by Alice during the interval fails to arrive unmodified at Bob (before the last packet of interval arrives at Bob). An adversary *Eve* can sit anywhere on the path between Alice and Bob, and we empower Eve to drop, modify, or delay every packet or add her own packets. A *path quality monitoring (PQM) protocol* is a protocol that Alice and Bob run to detect whether the number of failures during the interval exceeds a certain fraction of total packets transmitted.

Definition 3.2.1. Given parameters $0 < \alpha < \beta < 1$ and $0 < \delta < 1$, we say a protocol is a (α, β, δ) *secure PQM protocol* if, letting T be the number of packets sent during the interval:

1. *(Few false negatives.)* If more than βT packet delivery failures occur then the protocol raises an alarm with probability at least $1 - \delta$, *no matter what Eve does.*

2. *(Few false positives.)* If no intermediate node behaves adversarially (*i.e.,* no packets are added or modified on the path, but packets may be reordered or dropped due to congestion) and at most αT failures occur then the protocol raises an alarm with probability at most δ.

We assume that the T packets sent during an interval are distinct, because of natural variation in packet contents, and the fact that even successive packets sent by the same host have different ID fields in the IP header [33] (note that even retransmissions of the same TCP segment correspond to distinct IP packets, because of the IP ID field).

3.2.1 Properties of our security definition

Our definition is strongly motivated by our intended application of enabling routing decisions or SLA violation detection. The most important security guarantee it provides is that *no matter what Eve does* she cannot prevent Alice from raising an alarm when the failure rate for packets that Alice sent to Bob exceeds β. As such, our definition encompasses attacks by nodes on the data path that include (but of course are not limited to): colluding nodes that work together in order to hide packet loss, an adversarial node that intelligently injects packets based on timing observations or deep packet inspection, a node that preferentially treats packets that it knows are part of the PQM protocol, and a node that masks packet loss by injecting an equal number of nonsense packets onto the data path. We emphasize that we never make any assumptions on the distribution of packet loss on the path; our model allows for any possible 'failure model', including one where, say, packet loss is correlated across different packets.

On the other hand, as a routing-decision enabling tool, we do not require PQM protocols to *prevent* packet delivery failures but rather only *detect* them. Second, rather than determining *exactly* how many failures occurred, the protocol is only required to detect if the number of failures exceeds a certain threshold. (While solutions that exactly count failures certainly exist, *e.g.,* see discussion on IPsec in Section 3.7, they typically require cryptographically authenticating and/or encrypting all traffic and hence are more expensive to operate in high-speed routers.) Third, we do not require our protocols to distinguish between packet failures occurring due to adversarial tampering or due to "benign" congestion or malfunction.

Next, while our security definition requires that our protocols do not raise a (false) alarm when the one-way failure rate is less than α for the *benign setting*, we do allow for the possibility

of raising an alarm due to adversarial tampering even when fewer than an α fraction of failures occur. This is because an adversarial node has the power to arbitrarily tamper not just with data packets, but also with any packets that are sent as part of the PQM protocol; thus Eve can always make a path look worse by selectively dropping all acknowledgment or report messages that Bob sends to Alice, even if all the original packets that Alice sent to Bob were actually delivered. (In this chapter,we will assume that any acknowledgment or report messages that Bob sends to Alice are sent repeatedly to ensure that, with high probability, they are not dropped due to normal congestion.) When this happens, it may very well make sense for the protocol to raise an alarm, and the router to look for a different path.

While we allow adversarial nodes to *add* an arbitrary number of packets to the path, we ignore denial of service (DoS) attacks in which an adversary exhausts the *computational capacity* of Alice or Bob by flooding the path with packets. That is, we will assume that the adversary cannot exhaust the computational capacities of Alice and Bob; in practice, there are standard techniques, *e.g.,* rate limiting, that deal with these sorts of DoS attacks. See also the discussion of *monotonicity* in Section 3.7.1.

Finally, while in principle α, β can be chosen arbitrarily, there are a number of practical issues involved in the choice of these parameters. Firstly, we shall show in Sections 3.4-3.5 that the (communication, and storage) overhead of our protocols is related to the ratio $\frac{\alpha}{\beta}$; a smaller ratio leads to less overhead. Furthermore, the absolute value of α is sometimes constrained by interval synchronization; we discuss these issues further in Appendix B.2.

3.2.2 Related works

The literature on path-quality monitoring typically deals only with the benign setting; most approaches either have the destination return a count of the number packets he receives from the source, or are based on active probing (ping, traceroute, [60, 95, 96] and others). However, both approaches fail to satisfy our security definition. The counter approach is vulnerable to attack by an adversary who hides packet loss by adding new, nonsense packets to the data path. Active probing fails when an adversary preferentially treats probe packets while degrading performance for regular traffic, or when an adversary sends forged reports or acknowledgments to mask packet loss. Even known passive measurement techniques, where normal data packets are marked as probes, either explicitly as in IPPM [60] and Orchid [82] or implicitly as in Trajectory Sampling [33] and PSAMP [59], are vulnerable to the same attacks as active probing techniques if the adversary can distinguish the probe packets from the non-probe packets (*e.g.,* see [47] for attacks on PSAMP).

To obtain path-quality monitoring protocols that work in the adversarial setting, we have developed protocols that are more closely related to those used for traffic characterization. For example, our secure sampling protocol uses passive measurement techniques similar to those of [33, 59], that are designed for characterizing traffic mix. Similarly, our secure sketch protocol draws on ℓ_2-norm estimation schemes [7, 5, 24, 100] that are typically uses to characterize data streams. (See *e.g.,* [108] for a survey of data streaming algorithms used in networking.) Because our protocols are designed for the adversarial setting, they require the use of keys and cryptographic hash functions (see sections 3.3 and 3.6) in order to prevent an adversary from selectively adding and dropping packets in a manner that skews the estimate returned from the sketch. On the other hand, we can use the special structure of the path-quality monitoring setting to prove new analytical bounds which result in *provably lower communication and storage requirements* than those typically needed in traffic characterization applications. Also, at the end of Section 3.5.4 we discuss how the new result of [80] for sketching adversarially-chosen sets could be applied to our setting.

Our results are also related to works in the cryptography and security literature. In the security literature, traditional works on providing availability typically give guarantees on a *per-packet* basis, resulting in schemes with very high overhead, see *e.g.,* [32] [89] and later works. While *statistical* PQM protocols have been considered in the security literature [81, 99, 12], ours is the first work in this area to provide a formal security definition and to *prove* the security of our protocols within this model. We argue that such a model is crucial to understanding the security guarantees provided by a protocol. Indeed, one of Fatih's [81] PQM approaches is based on a simple counter (and is therefore vulnerable to the attack described above), while Listen [99] is a protocol that does not use cryptographic operations, and is thus vulnerable to attack by an intermediate node that injects false acknowledgments onto the path. Finally, while Stealth Probing [12] is secure in our model, it incurs the extra overhead of encrypting and authenticating all traffic.

3.3 Cryptographic primitives

Our PQM protocols use several cryptographic primitives, with different security properties and performance. We describe the security properties of these primitives below:

Keys. Each of our protocols require some sort of key infrastructure; the secure sketch (Section 3.5) and symmetric secure sampling (Section 3.4.1) protocols require parties to share a pairwise secret key, while the asymmetric secure sampling protocol (Section 3.4.2) require public-keys. Notice that the requirement for pairwise secret keys, does not imply that we must maintain an infrastructure of pairwise keys for the Internet. All of our protocols require participation of only two parties. Parties can derive pairwise keys via, *e.g.,* authenticated Diffie-Hellman key exchange (as used in TLS/SSL [31]) using Public Key Infrastructure such as DNSSEC or some out-of-band secure channel. Furthermore, an organization owning multiple routers running PQM might have an incentive to assign pairwise secret keys. Our protocols require two types of keys: *master keys*, and *interval keys*. Master keys are strong keys that are set up when the protocol initializes, and must remain secure for the lifetime of the protocol. Interval keys are ephemeral keys that are derived at the beginning of each interval, and must remain secret only while packets belonging to that interval are in flight on the path between Alice and Bob.

Collision-Resistant Hash (CRH) function is a function H for which it is infeasible (for any computationally-bounded algorithm) to find a collision, *i.e.,* $m \neq m'$ fulfilling $H(m) = H(m')$. (This informal definition suffices for the purposes of this chapter. For a more precise definition of CRH see [93].) Typical choices of H are SHA-1 and (truncated) SHA-256. The output of $H(x)$ is called the *digest* of x, and we assume it is 160 bits long.

PseudoRandom Function (PRF) [50] is a keyed function $h_k(\cdot)$ that maps an arbitrary length string to an n-bit string using a key k; in our case, $n = 64$ or 96 usually suffice. If the key k is chosen at random, then to an adversary with no knowledge of k the function $h_k(\cdot)$ looks totally unpredictable and cannot be distinguished (except with an insignificant probability) from a truly random function (where each input is mapped independently to a uniformly random output). Hence, in our analysis we may treat h_k as if it is truly random. Our protocols use PRFs in two ways.

- A PRF h is used to derive interval keys from the pairwise shared master key and the interval number. To derive interval key k_u for interval u from master key k, each party need only compute $k_u = h_k(u)$. Notice that as long as parties have synchronized their interval numbers u, they can use their knowledge of the master key k to independently compute k_u (without requiring a key-agreement protocol or a handshake).

Because the PRF h is used only once per interval, and also needs to be resilient against many queries, we will let h be traditional conservative pseudorandom function. The most common way to (heuristically) realize pseudorandom hash functions (PRFs) is using a full-fledged cryptographic hash functions such as SHA-1 in HMAC mode [67], or with a block cipher like AES in a MAC mode of operation. Their typical performance in a software implementation is 10–20 cycles per input byte, which suffices for many applications.

- All our protocols require a hash computation on the entire contents of every sent packet,[1] and all subsequent processing of the packet relies only on this hash value. For packet hashing, we will use a PRF keyed with the interval key k_u. The k_u is used only for the duration of single interval (typically about 100ms); once the interval ends, the key no longer needs to be kept secret. It follows that the security requirement on this PRF is weaker than is typically required for most applications. Thus, our packet-hashing PRF should be (a) fast enough to keep up with multi-Gbit/sec packet streams, (b) remain secure after $T = 10^7$ applications and/or for about 100ms. While designing PRFs that are especially suited to this purpose remains an interesting area for future research, in Appendix B.1 we discuss some realizations of our packet-hashing PRF in *both hardware and software* based on known cryptographic hash functions.

Universal hash functions are keyed hash functions similar to PRFs, but have a weaker security requirement; PRFs are indistinguishable from functions that map *every* input to an random *independent* outputs, while universal hash functions only require independence between some small number of outputs [23]. In Sections 3.5.4 and 3.5.4, we shall show that packet hashing can sometimes be performed using these weaker hash functions, instead of PRFs.

In this work, we consider two types of universal hash functions.

- **ε_g-almost universal hash function** g producing n-bit outputs guarantees that for any pair of distinct inputs x, x', then

$$\Pr\left[g_{k_u}(x) = g_{k_u}(x')\right] \leq \varepsilon \tag{3.1}$$

  where the probability is over the choice of k_u used to key g. There are many possible realizations of such hash functions, see for example [18] for a survey. In this work, we sometimes use GHASH [78] to compute sample parameters for our constructions. GHASH is an ε_g-almost universal hash function that produces n bit outputs. GHASH hashes variable-length packets by breaking the packet into blocks of length m and iteratively hashing each block. For a packet of length ℓ, GHASH has $\varepsilon_g = \frac{\ell}{m} 2^{-n}$.

- **4-wise independent hash function** g producing n-bit outputs guarantees that for any four distinct inputs x_1, x_2, x_3, x_4 and (not necessarily) distinct outputs y_1, y_2, y_3, y_4, then

$$\Pr\left[g_{k_u}(x_i) = y, \forall i = 1...4\right] = \left(\tfrac{1}{2^n}\right)^4 \tag{3.2}$$

  where the probability is over the choice of k_u used to key g. We shall use 4-wise independent hash functions to realize (theoretically)-faster packet hashing in our 'secure sketch' protocol. To realize a 4-wise independent hash function, we use polynomials of degree 3 [23], for example, to compute $g_{k_u}(x)$ set key $k_u = (a_0, a_1, a_2, a_3)$ and output $a_3x^3 + a_2x^2 + a_1x + a_0$. This computation can be done in three multiplications using Horner's rule.

[1] For convenience, we abuse notation and say that whenever the PRF is applied to a packet, the non-invariant fields of the packet header are discarded from the input. In the case of IPv4, this means excluding the ToS, TTL and IP checksum (see [33, Section II.A]).

Notice that a PRF provides a strictly stronger theoretical guarantees that a universal hash function (since every PRF is also a universal hash function). However, in practice a PRF could be *faster* than a 4-wise independent function! While this certainly seems counterintuitive, it follows because in practice we typically realize 4-wise independent functions based on constructions that come with *rigorous proofs of correctness* (*e.g.,* polynomials of degree 3 [23]), while we use PRFs that come with only *heuristic* guarantees of correctness (*e.g.,* GHASH-AES [78] is a PRF under the heuristic assumption that AES is a fixed-input-length PRF, see Appendix B.1). However, even fast (heuristic) constructions of PRFs are typically based on ε_g-almost universal hash functions (see Appendix B.1), and as such, we can safely assume that ε_g-almost universal hash functions are always faster than PRFs.

Message Authentication Code (MAC) is a basic cryptographic primitive that can be realized using a PRF: using a shared key k, for a message m, one party will send $(m, h_k(m))$ and the other party can verify that a pair (m, t) satisfies $t = h_k(m)$. The value $h_k(m)$, called the *tag*, cannot be feasibly forged by an adversary that does not know k. We denote $\text{MAC}_k(m) = (m, h_k(m))$. We shall assume that the MAC tag (*i.e.,* PRF output) is $n = 96$ bits.

Digital signatures provide authenticity in the public-key setting. Here a private key SK is used to sign a message m and obtain a signature σ; we denote this with $\sigma = \text{Sign}_{SK}(m)$. A public key PK is known to all parties and is used to verify the signature; the $\text{Verify}_{PK}(\sigma)$ operation outputs a message m for valid signatures and aborts otherwise. Digital signatures are more computationally expensive than MACs, so we use them only for infrequent synchronization data.

3.4 Secure sampling PQM

In a sampling-based protocol, Alice and Bob agree on a small set of packets (the probes) for which Alice expects acknowledgments from Bob. Then, Alice can detect when the path quality is unacceptable when too many probes are unacknowledged. These protocols limit the storage and communication overhead because only a small fraction of traffic is monitored, and also allow Alice to measure round-trip delay by monitoring arrival time of acks.

However, such protocols are inherently vulnerable to adversaries that preferentially allow probes to travel unharmed, but drop, delay, or modify other packets. Since most packets are not probes, such an adversary can disrupt traffic without Alice realizing that something went wrong. To prevent such attacks, in our secure sampling protocols Alice and Bob use a shared PRF to coordinate their sampling. The cryptographic properties of the PRF, discussed in Section 3.3, prevent an adversary from distinguishing probes from non-probes.[2] Use of a PRF in our setting is necessary for security; in Appendix B.3 we show an example of why a non-cryptographic hash function (*e.g.,* CRC) is insufficient.

We present three protocols. The Symmetric Secure Sampling protocol is designed for the setting where Alice and Bob share pairwise secret keys. The two Asymmetric Secure Sampling protocols (one for senders and one for receivers) use a variant of *delayed-exposure techniques* (*c.f.,* TESLA [90] [17, 20, 25] and the references therein) to eliminate the need for pairwise keys, at the cost of some increased storage at Alice or Bob. The asymmetric protocols are especially advantageous when one of the parties is a server that needs to engage in simultaneous PQM sessions with many clients. These protocols also have some nice scaling properties (Section 3.4.2).

[2] We stress that probes are ordinary data packets that are part of the data stream and are not explicitly marked. Marking/modifying ordinary packets is undesirable for several reasons: (a) it must be undone by the receiver prior to processing or forwarding, (b) it may cause backward-compatibility problems by introducing packet formats that is are unrecognized by devices along the path, (c) it may run into MTU limitations, etc.

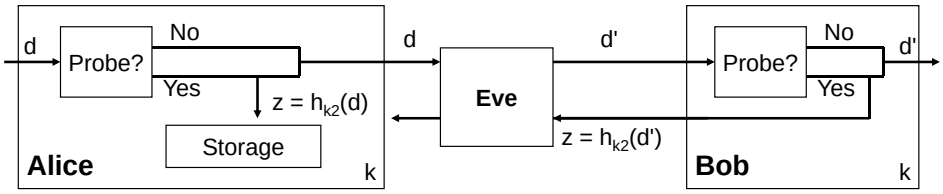


Figure 3.1: Secure Sampling.

3.4.1 Symmetric Secure Sampling

We assume Alice and Bob share a secret (master) key k. They also know a parameter p, called the *probe frequency*. During each interval, our symmetric secure sampling protocol operates as follows:

1. Alice and Bob derive an interval-specific secret key by applying a PRF keyed with the master key k to the interval number u, *i.e.,* $(k_1, k_2) = h'_k(u)$. In Appendix B.2, we give a detailed treatment of techniques that can be used to achieve interval synchronization between Alice and Bob.

2. After transmitting each packet d, Alice decides whether d is a probe. Specifically, she uses k_1 and the probe frequency p to run a Probe function that is implemented using a (packet-hashing) PRF h keyed with k_1 and outputting an integer in $\{0, \ldots, 2^n - 1\}$, as follows:

$$\mathsf{Probe}_{k_1}(d) = \begin{cases} \text{YES}, & \text{if } \frac{h_{k_1}(d)}{2^n} < p; \\ \text{NO}, & \text{else.} \end{cases} \tag{3.3}$$

   If $\mathsf{Probe}_{k_1}(d)$ outputs YES then Alice stores the tag $z = h_{k_2}(d)$ in a table.[3]

3. Bob receives d' and computes $\mathsf{Probe}_{k_1}(d')$. If it outputs NO then do nothing; if it outputs YES then transmit the tag $z' = h_{k_2}(d')$ back to Alice.

4. Alice receives the acknowledgment z' and removes it from her table if it is present in her table. If the acknowledgement is invalidly MAC'd or not present in her table, Alice ignores it.

At the end of an interval, Alice raises an alarm if and only if her table contains more than $pT\sqrt{\alpha\beta}$ remaining entries.[4] Otherwise she does not raise an alarm.

Theorem 3.4.1. *The symmetric secure sampling protocol is an (α, β, δ)-secure PQM protocol for $\alpha < \beta \leq 4\alpha$ as per Definition 3.2.1, whenever the probe frequency p and number of packets per interval T satisfy*

$$pT > \ln(\tfrac{1}{\delta}) \tfrac{3}{(\sqrt{\beta} - \sqrt{\alpha})^2} \quad . \tag{3.4}$$

[3]When h uses a modified Wegman-Carter construction (see Appendix B.1),the computation of $h_{k_2}(d')$ can reuse the universal hash already computed for $h_{k_1}(d')$, and thus amounts to a single AES or DES invocation.

[4]To obtain this threshold, we could have used the mid point between $p\alpha T$ and $p\beta T$. However to get much better parameters for our protocols, we can apply maximum likelihood estimation to obtain the threshold above, since (from proof of Theorem 3.4.1) V, or the number of unacknowledged probes in Alice's table, is a binomial random variable. We obtain the threshold above using maximum likelihood estimation as $(\mu_\alpha \sigma_\beta + \mu_\beta \sigma_\alpha)/(\sigma_\alpha + \sigma_\beta)$ where $\mu_\alpha = p\alpha T$ is the mean of V when the loss rate is αT and $\sigma_\alpha^2 = (1-p)p\alpha T$ is the variance of V when the loss rate is αT, and σ_β and μ_β are defined analogously. Then, we get the threshold $pT\frac{\beta\sqrt{\alpha} + \alpha\sqrt{\beta}}{\sqrt{\alpha} + \sqrt{\beta}} = pT\sqrt{\alpha\beta}$.

When $\alpha = \beta/2$ we can use [6, Thm. 19] to obtain a slightly better bound $pT > \ln(\frac{1}{\delta})\frac{2\ln 2}{(\sqrt{\beta}-\sqrt{\alpha})^2}$, so that when $\delta = 1\%$, we require $pT > 75/\beta$.

Proof of Theorem 3.4.1. First, we observe that *regardless of any strategy Eve adopts*, and independently of all other packets, the probability each dropped/modified packet is a probe is p. Suppose that $h_{k_1}(\cdot)$ in Probe were replaced by an independent truly random function (for each choice of k_1). We claim that every sent packet would be a probe independently with probability p. To see why, first consider a single interval. Recall that within a single interval, we assumed that packets sent by Alice are unique. Furthermore, Eve cannot use her observations of past packets and acks to determine if a given packet is a probe. Next, recall that the interval key is refreshed at the end of an interval; it follows that the packets selected as probes in a given interval are independent of the packets selected as probes in all other intervals (even if packets are not unique across intervals). Next, notice that the above must hold for the real implementation of Probe using h_{k_1}, since otherwise Eve could distinguish between the PRF and a truly random function, contradicting the security of the PRF. Notice also that if Eve wants to use Bob to test if a given packet is a non-probe (and thus may be dropped), she must first *send* (a test copy) of that packet to Bob. However, once the packet is received by Bob, Eve cannot do any more damage; since we assume that sent packets are unique, it follows that Bob has already received a copy of the packet, and Eve gains nothing by dropping it. Recall also that, here, we do not consider denial of service attacks in which Eve exhausts the computational resources of Alice or Bob by flooding the link with packets.

For the false positives condition of Definition 3.2.1, suppose the failure rate is less than α. Notice also the false positives condition of Definition 3.2.1 is conditioned on the fact that no node behaves adversarially, *i.e.,* maliciously drops ACKs (or synchronization messages, see Section B.2). Thus, the probability of misdetection is the probability that a larger than $\sqrt{\alpha\beta}$-fraction of the samples are dropped. Let V be the number of remaining (unacknowledged) entries in Alice's table. When each packet is independently sampled with probability p, then if $\beta < 4\alpha$ we can find the false positive probability

$$\begin{aligned} P_{FN} &= \Pr[\, V > pT\sqrt{\alpha\beta} \mid \text{failure rate } = \alpha \,] \\ &= \Pr[\, V > p\alpha T(1 + \tfrac{\sqrt{\beta}-\sqrt{\alpha}}{\sqrt{\alpha}}) |\, E[V] = p\alpha T \,] \\ &\leq e^{-\frac{(\sqrt{\beta}-\sqrt{\alpha})^2}{3}\, pT} \end{aligned} \tag{3.5}$$

where the equality follows from the fact that when the failure rate is α, we expect that the estimator V to be a p-fraction of the number of dropped packets, αT. The inequality follows from the fact that V is a binomial random variable $\mathcal{B}(\alpha T, p)$, and the Chernoff bound[5]of [9, Fact 4], which holds when $0 < \frac{\sqrt{\beta}-\sqrt{\alpha}}{\sqrt{\alpha}} < 1$ or $\alpha < \beta < 4\alpha$. By our observation above, this inequality still holds (up to a negligible additive factor) when we sample probes using a pseudorandom function.

[5]We use the following Chernoff bounds. Let X_i be i.i.d indicator variables with mean μ, and let

$$\Pr\left[\sum_{i=1}^{n} X_i \leq (1-\gamma)N\mu\right] \leq e^{-\gamma^2 N\mu/C_1} \tag{3.6}$$

$$\Pr\left[\sum_{i=1}^{n} X_i \geq (1+\gamma)N\mu\right] \leq e^{-\gamma^2 N\mu/C_2} \tag{3.7}$$

If $0 < \gamma < 1$ then [9, Fact 4] gives $C_1 = 2$ and $C_2 = 3$. If $0 < \gamma < \frac{1}{2}$ then [6, Thm. 19] gives $C_1 = C_2 = 2\ln 2$.

Next, consider the false negatives condition of Definition 3.2.1. First note that Eve cannot forge a valid Ack to a packet that was not received by Bob, since she only sees the output of the PRF h_{k_2} on packets that Bob receives, and cannot predict its value on any other input. Therefore all that Eve can do is to bias the measurement by preferentially dropping non-probes. Using [9, Fact 4]again, if probes are sampled independently with probability p then

$$\begin{aligned} P_{FP} &= \Pr[\, V > pT\sqrt{\alpha\beta} \mid \text{failure rate } > \beta \,] \\ &= \Pr[\, V > p\beta T(1 - \tfrac{\sqrt{\beta}-\sqrt{\alpha}}{\sqrt{\beta}}) |\, E[V] = p\beta T \,] \\ &\leq e^{-\frac{(\sqrt{\beta}-\sqrt{\alpha})^2}{2}\, pT} \end{aligned} \tag{3.8}$$

where the equality follows from simple algebra and the fact that when the drop rate is β, V is a binomial random variable $\mathcal{B}(\beta T, p)$, and the inequality again follows from the Chernoff bound of [9, Fact 4], which holds when $0 < \frac{\sqrt{\beta}-\sqrt{\alpha}}{\sqrt{\beta}} < 1$ or when $\alpha < \beta$. As observed above, (3.8) still, holds up to a negligible factor, when the probes are sampled using a PRF. Notice that dropping Acks (or synchronization messages, see Section B.2) cannot help Eve, as it only makes the source *more* likely to raise an alarm. It follows from equations (3.5), (3.8) and Definition 3.2.1 that, given α, β and δ, such that $\beta < 4\alpha$, the protocol is secure whenever (3.4) holds. □

3.4.2 Asymmetric Secure Sampling

This section describes variants of the above protocol for the case where a single router (the *server*) deals with a large number of other routers (the *clients*). Our protocols support server scalability by minimizing the per-client cost of the server. In particular, the server will not need to establish a separate key for every client. We will, however, assume that the clients can dedicate more resources to the PQM protocol. We provide two different protocols, depending on whether the server is receiving from, or sending to, its clients (of course, the two PQM protocols can be applied jointly to monitor both directions).

We again divide time into intervals, and the idea is that the server performs his operations (as either sender or receiver) with private keys, which we call the *salt*, unknown to anyone except himself until the end of the interval, at which time he releases the salt to all interested clients. The point is that by the time the server releases the salt it is too late to cheat; note that even dishonest clients cannot cheat honest clients because no one except the server knows the salt until the end of the interval.

Instead of using symmetric keys between each pair of parties, here we assume that the server has a public/private key pair (PK, SK) where the public key PK is known to all parties (e.g., through a Public Key Infrastructure). To ensure that the computationally-expensive public-key operations are used infrequently, we will use cryptographic delayed-exposure techniques (*c.f.,* TESLA [90] [17,20,25] and the references therein) that require secure clock synchronization. We assume that each client securely synchronizes her clock so that it lags behind the server's clock by at most τ seconds, where τ is a constant known to all parties. In Appendix B.2.2 we present a simple secure protocol for achieving this synchronization.

Receiving-Server Secure Sampling (RSSS)

We first consider the case where a single server (Bob) is receiving traffic from multiple clients (each playing the role of Alice). The following protocol allows every client to monitor the path quality for traffic that it sends to the server, while the server requires *no* storage and can use the same key to engage in PQM with every client. During the u-th interval, the RSSS protocol operates as follows:

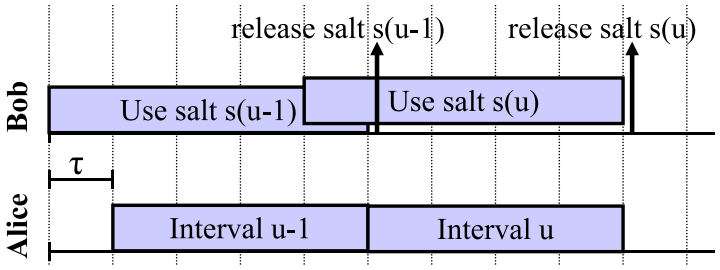


Figure 3.2: Timing for Asymmetric Secure Sampling.

1. *(Interval Setup.)* Bob, the receiver, randomly chooses a pair of salt values $(\mathsf{s}_1(u), \mathsf{s}_2(u))$ that he keeps secret until the very end of the interval.

2. *(Packet Transmission.)* Packet transmission during the interval proceeds as follows:

   - For each packet d Alice wishes to send, she stores the digest $H(d)$ (computed using a collision-resistant hash) in her table. Suppose Alice sends T packets in the interval. (This means Alice stores T digests. In Section 3.4.3 we discuss how Alice can independently subsample packets to reduce her storage requirements.)
   - Upon receiving each packet d', Bob computes its digest $z' = H(d')$. He then evaluates $\mathsf{Probe}_{\mathsf{s}_1(u)}(z')$; if NO then he does nothing, and if YES then he transmits an ACK of the form $\mathrm{MAC}_{\mathsf{s}_2(u)}(z', u)$ back to Alice.
   - Each sender (Alice) stores all the ACKs received which included the current interval u.

3. *(Salt Release.)* Bob maintains the secrecy of the salt until τ seconds after interval u ends. At that time he reveals the salt $(\mathsf{s}_1(u), \mathsf{s}_2(u))$ to all clients by sending a SALTRELEASE packet containing $\mathrm{Sign}_{SK}(u, \mathsf{s}_1(u), \mathsf{s}_2(u))$(see Figure 3.4.2).

4. *(Security check.)* If Alice fails to receive a SALTRELEASE containing a signature σ within 1 RTT after the interval u ends, or if $\mathrm{Verify}_{PK}(\sigma)$ doesn't return a tuple $(u, \mathsf{s}_1(u), \mathsf{s}_2(u))$, then Alice raises an alarm. Otherwise, she uses salt $\mathsf{s}_1(u)$ to run the Probe function on the packet digests in her table, and salt $\mathsf{s}_2(u)$ to verify the ACKs in her table. Then Alice counts the number of packets for which $\mathsf{Probe}_{\mathsf{s}_1(u)}(z) =$ YES and no valid ACK is stored in her table; call this count V. Finally, Alice raises an alarm if $V > pT\sqrt{\alpha\beta}$. Notice that this step of the protocol can be computed *offline*.

Notice that our protocol does *not* require Bob to send out the salt immediately at the end of the interval. However, we observe from Step 5 above, that there is a tradeoff between frequency of salt release messages and storage at Alice; the longer Bob delays sending out the salt, the longer Alice has to wait before she can clear her table.

Assume for now that all parties' clocks are perfectly synchronized. Then Eve cannot cheat within any single interval:

Theorem 3.4.2. *The RSSS protocol is an (α, β, δ)-secure PQM protocol for $\alpha < \beta \leq 4\alpha$ as per Definition 3.2.1, whenever the probe frequency p and number of packets per interval T satisfy*

$$pT > \ln(\tfrac{1}{\delta}) \tfrac{3}{(\sqrt{\beta}-\sqrt{\alpha})^2} \quad . \tag{3.9}$$

packet digest	ACK	Probe
$z_1 = H(d_1)$		YES
$z_2 = H(d_2)$	$\text{MAC}_{\mathsf{s}_2(u)}(B, z_2, u,)$	Yes
$z_3 = H(d_3)$		NO
$z_4 = H(d_4)$		NO
$z_5 = H(d_5)$		NO
$z_6 = H(d_6)$		NO
$z_7 = H(d_7)$		NO
$z_8 = H(d_8)$	$\text{MAC}_{\mathsf{s}_2(u)}(B, z_8, u)$	YES
$z_9 = H(d_9)$		NO
$z_{10} = H(d_{10})$		NO
$z_{11} = H(d_{11})$	$\text{MAC}_{\mathsf{s}_2(u)}(B, z_{11}, u)$	YES
$z_{12} = H(d_{12})$		YES
$z_{13} = H(d_{13})$		NO

Figure 3.3: Alice's table after at the end of interval u. Here Alice observes packet-delivery failures for packets $1, 12$.

When $\beta = 2\alpha$ we can use a tighter bound of [6, Thm. 19](instead of (3.9)) to find that when $\delta = 1\%$, we require $pT > 75/\beta$.

When clocks are perfectly synchronized, we omit the proof, since it is almost identical to that of Theorem 3.4.1 (because the salt is kept secret until the end of the interval). Furthermore, notice that even dishonest senders cannot bias an honest sender's measurements, since they learn nothing about the salt until the interval is over. Now suppose that Alice's clock lags Bob's clock by at most τ seconds. It follows that there will be period of time of length $< \tau$ where Alice is operating in interval $u - 1$ while Bob has already moved into interval u. To deal with this, during the first τ seconds of each interval, Bob uses *both* the salt of the current interval $\mathsf{s}(u)$ and the salt from the *previous* interval $\mathsf{s}(u-1)$ in order to create his ACKs. While most Internet routers are able to maintain a clock with accuracy of 21ms or less [79], *secure* clock synchronization is a non-trivial problem. In Appendix B.2.2 we show a simple stateless protocol that allows Alice and Bob synchronize their clocks to within 1.5 round trip times.

Transmitting-Server Secure Sampling (TSSS)

We now turn our attention to the case where a single server is *sending* to multiple clients, and each client wants to monitor the traffic it receives from the server while imposing minimal cost on the server. Note that the server is now Alice and the client is Bob. Here the server keeps a single counter per client, and modifies the packets it sends by appending a short MAC tag, that is keyed with *same* key for each client.

The TSSS protocol proceeds as follows. As before, the server picks random salt values $(\mathsf{s}_1(u), \mathsf{s}_2(u))$ at the beginning of the interval, and releases them at the end of the interval. Here, however, the server will keep, for each client B, a count $T_{\text{A}}(B)$ of the number of packets it sends to B during the interval. The server also authenticates all traffic that she sends using the (client-independent) salt: for a packet d, the server will compute a packet digest $z = H(d)$ and then appends the tag $h_{k_2}(u, z)$ to the packet that he sends the client.

The client will randomly sample a p-fraction of the packets received. For each such packet d', he stores the corresponding digests $z' = H(d')$ and the received tag. At the end of the interval, the server reveals the salt as above, and also sends $\text{Sign}_{SK}(T_{\text{A}}(B))$ to B. Each client B verifies the electronic signature and checks all its stored packet digests and tags using this salt. Let T_{B} be the number of valid packets thus found by B; then B estimates the number of failures as

$V = pT_A(B) - T_B$. As before, the client raises an alarm if $V > pT_A(B)\sqrt{\alpha\beta}$. Using an argument similar to Theorem 3.4.1, the protocol is secure if $\beta < 4\alpha$ and

$$pT_A(B) > \ln(\tfrac{1}{\delta})\tfrac{3}{(\sqrt{\beta}-\sqrt{\alpha})^2} \quad . \tag{3.10}$$

A note on the scalability of RSSS

Our Receiving-Server Secure Sampling protocol has useful scaling properties, that make it attractive even outside the client-server setting. Consider a network with M routers, where each router would like to run a pairwise PQM protocol (acting as both a sender and receiver) with all other routers. Then, while the Receiving-Server Secure Sampling protocol requires that each router to store the $M-1$ public-keys, the storage and *online* computation overhead at each router is independent of the number of routers M! To see why, first notice from Section 3.4.2 that when the router acts as a receiver, its incurs no storage overhead, while its computation/communication overhead depends only on the total volume of traffic it receives, and not on the number of senders $M-1$. Next, when the router acts as a sender, it only needs to keep a table of digests of the packets it sends (see *e.g.,* Figure 3.3) that additionally specifies the receiver the each packet stored in table was sent to. The size of this table depends on the volume of sent traffic, but is independent of M. Furthermore, it is computed (using a collision resistant hash) independently of the keys of any of the $M-1$ receivers. Indeed, receiver-specific computations are *only* required during the *offline* 'security check' step of the protocol.

3.4.3 Some sample parameters

Suppose $\alpha = \frac{1}{2}\beta$ and $\beta = 1\%$. We assume a fully utilized 5 Gbps link with an average packet of 3000 bits and an average round trip time (RTT) of 100 msec. Then about $T = 10^7$ packets are sent during an RTT.

Symmetric Secure Sampling. Using the improved bound from [6, Thm. 19] in Theorem 3.4.1 our symmetric sampling protocol is secure when the probe frequency is $p > \frac{75}{\beta T} = 7.5 \times 10^{-4}$. This p is also the communication overhead, *i.e.,* the amount of added Ack packets as a fraction of the data traffic. Using 96-bit packet digests (see Section 3.3), Alice needs about $pT \sim 90$ KB of storage during a single round trip time. The amount of storage required for Alice can be reduced without compromising security by noting that (3.4) gives a tradeoff p and T. Alice can decrease her sampling rate to p' if she is willing to use a longer interval $T' = Tp/p'$. Since almost every probe packet tag will be deleted after 1 RTT, this nominally reduces Alice's storage to $p/p' \cdot 90$ KB. This comes at the cost of reduced PQM temporal resolution, due to the longer intervals. (Notice that Alice can arbitrarily decrease her sampling rate without coordinating with Bob simply by changing the parameter p in her Probe function.)

RSSS. As described above, the Receiving-Server Secure Sampling protocol requires the sending client to store information about *every* packet she sends to Bob for the duration of a interval (which may last from a few milliseconds to a few RTTs depending on synchronization quality). In case the intervals last an RTT or more, it is not practical to expect the sender to keep digests of over 10^7 packets in her storage, and so we apply *subsampling* here to reduce the fraction of packets stored: each sender only stores a q fraction of the packets she sent, where each packet is stored independently with probability q. In term of monitoring this is essentially the same as reducing the packet stream by a factor of q, so when $\beta = 2\alpha$ from the improved version of (3.9) we can see that $pqT > \frac{75}{\beta}$ suffices, giving a tradeoff between storage at Alice qT, and probe frequency and communication overhead p. For example, suppose that the probe frequency is $p = 0.2$. Then, by (3.9), Alice should store $qT \approx \frac{75}{p\beta} = \frac{75}{0.2 \cdot 0.01} = 3.75 \times 10^4$ packet digests (160

bits each), and about p times as many corresponding ACK tags (96 bits each). Overall, this takes $3.75 \times 10^4 \cdot (160 + 0.2 \cdot 96)/8 \approx 840$ KB of storage. Thus, if intervals last for 1 RTT, so that $T \approx 10^7$, then the subsampling rate must be at least $q = 3.7 \times 10^{-3}$.

TSSS. Here, the sending server stores one 32-bit counter per client, and attaches a 96-bit tag to each message. Following (3.10), and using same parameters as above, the client needs to store $qT \approx \frac{75}{\beta} = 7.5 \times 10^3$ digests and tags, for a total storage of $7.5 \times 10^3 \cdot (160 + 96)/8 \approx 240$ KB.

3.5 Secure sketch PQM

In our secure sketch PQM protocol, Alice and Bob aggregate all traffic Alice sends to Bob into a short data structure called a *sketch*. (The difference between a sketch and a sample is that a sketch, although short, usually provides approximate information about the aggregate stream of packets, while a sample provides exact information about a single packet in the stream.) At the end of the interval, Bob sends his sketch to Alice and she compares the sketches to decide whether the failure rate exceeded β.

We can apply several sketching techniques [7, 5, 100, 24] for *second moment estimation* (or ℓ_2-norm estimation) into our framework to give secure PQM protocols. While sketches have been used before in the networking community (to estimate properties of data streams that are too long to be stored in their entirety; *c.f.* [100, 24] and the references in [108]), to the best of our knowledge this is the first time that they have been applied to the problem of path-quality monitoring. Furthermore, the special structure in the PQM problem allows us to obtain new and improved analytical bounds on the performance of these schemes. Also, it turns out that the path-quality setting has particular properties that enable us to achieve better performance for some of these schemes; indeed, we prove a new bound on the performance of [24]'s scheme that may be of independent interest.

In this section we start by explaining the relationship between moment estimation and path-quality monitoring, and then present our PQM protocol and discuss its security. We then show how the protocol works with several known moment estimation sketches and give settings of parameters based on both analytical guarantees and numerical experiments. Our results show that the secure sketch protocol is almost as lightweight, in terms of storage and communication, as the trivial (but insecure) idea of keeping counters of the number of packets sent and received.

3.5.1 PQM as moment estimation

We now show how why p^{th}-moment estimation (for $p \geq 1$) is sufficient to realize PQM. Later on, we will use this argument to how show second-moment estimation (for which a number of highly efficient and simple schemes are known [7, 5, 24, 100]) can be used to realize PQM.

Preliminaries. Recall that Alice sends a stream of T packets to Bob during an interval and let U be the "universe" of all possible packets (*e.g.,* if packets are 1500 bytes long then $|U| \approx 2^{1500 \cdot 8}$). We define the *characteristic vector* of a stream to be a U-dimensional vector that has c in the position corresponding to packet x if packet x appears in the stream c times (*e.g.,* for the stream 1,2,4,2,2 of packets drawn from universe $U = [4]$ the characteristic vector is [1 3 0 4].) In our setting, a characteristic vector is too long ($2^{1500 \cdot 8}$) to be represented explicitly; we will use it only for the purpose of explaining our protocols. Also, recall that p^{th}-moment of a vector $\mathbf{v}$ is $\|\mathbf{v}\|_p^p = \sum_i (v_i)^p$. (Note that the ℓ_p-norm is just the p^{th} root of p^{th} moment of the vector.)

Relationship between PQM and the p^{th} moment for $p \geq 1$. Let $\mathbf{v}_\mathrm{A}$ be the characteristic vector for the stream of packets sent by Alice, and let $\mathbf{v}_\mathrm{B}$ be the characteristic vector for the

stream of packets received by Bob. Now consider the characteristic vector $\mathbf{x} = \mathbf{v}_B - \mathbf{v}_A$. We can decompose any $\mathbf{x}$ into two vectors $\mathbf{x} = \mathbf{d} + \mathbf{a}$. The vector $\mathbf{d}$ is the characteristic vector of packets *dropped* on the path from Alice to Bob, and contains the non-negative components of $\mathbf{x}$. The vector $\mathbf{a}$ is the characteristic vector of packets *added* on the path from Alice to Bob, and contains the non-positive components of $\mathbf{x}$. Also notice that the non-zero coordinates of $\mathbf{d}$ and $\mathbf{a}$ are disjoint.

Now let D be the number of packets dropped on the path from Alice to Bob during the interval, and let A be the number of packets added during the interval. (We count a single packet that was *modified* on the path from Alice to Bob as a single *dropped* packet plus a single *added* packet.) Thus, we have the following simple, but very useful identity:

$$\|\mathbf{x}\|_p^p = \|\mathbf{d}\|_p^p + \|\mathbf{a}\|_p^p = D + \|\mathbf{a}\|_p^p \geq D + A \tag{3.11}$$

The first equality follows because the non-zero coordinates of $\mathbf{d}$ and $\mathbf{a}$ are disjoint. The second equality follows because every packet that Alice send is unique so that that $\mathbf{d}$ is a $\{0,1\}$-vector for every $i \in [K+1]$. Finally, the last inequality follows because $\mathbf{a}$ is an integer vector, so that for any $p \geq 1$, it follows that $\|\mathbf{a}\|_p^p \geq |\mathbf{a}|_1 = A$ with equality when $p = 1$.

Now, recall Definition 3.2.1. To satisfy the "few false positives" condition we need to consider the *benign case* in which at most $D \leq \alpha T$ packets are dropped during the interval, and no packets are added so that $\|\mathbf{a}\|_p^p = 0$. From (3.11) it follows that satisfying the "few false positives" condition (in the benign case), just requires that Alice should not raise an alarm if $\|\mathbf{x}\|_p^p = D + 0 \leq \alpha T$.

To satisfy the "few false negatives" condition we need to consider the *malicious case* in which Eve drops $D \geq \beta T$ packets, and adds an arbitrary number of (potentially non-unique) packets $A \geq 0$. (We think of a packet modification as a dropped packet *plus* an added packet.) From (3.11) it follows that satisfying the "few false positives" condition (in the malicious case), requires that Alice raise an alarm if $\|\mathbf{x}\|_p^p \geq \beta T$.

The discussion above suggests the following ridiculous PQM protocol: Have Bob and Alice maintain $\mathbf{v}_\mathrm{A}$ and $\mathbf{v}_\mathrm{B}$, and have Bob send Alice $\mathbf{v}_\mathrm{B}$ at the end of the interval. Then define a decision threshold $\Gamma \in [\alpha T, \beta T]$, and have Alice raise an alarm if the first moment $\|\mathbf{v}_\mathrm{A} - \mathbf{v}_\mathrm{B}\|_p^p > \Gamma$. Notice that, in the malicious case, adding packets doesn't help Eve; whenever Eve adds packets, she simply increases $\|\mathbf{v}_\mathrm{A} - \mathbf{v}_\mathrm{B}\|_p$, and makes Alice more likely to raise an alarm.

Sketches. Of course, the PQM protocol described above are completely ridiculous because the $\mathbf{v}_\mathrm{A}, \mathbf{v}_\mathrm{B}$ vectors are much too large to be stored or transmitted explicitly. This is where *sketching* comes in. A p^{th}-moment estimation sketch is a set of probabilistic algorithms that allows us to estimate the p^{th}-moment of a vector $\mathbf{v}$ of length $|U|$ from a shorter sketch $\mathbf{w}$ of length N; typically, the length of the sketch is depends only on the number of packets in the stream, and not on the size of the universe U.

We will concern ourselves with moment estimation schemes where the sketch may be derived from $\mathbf{v}$ via a random linear map, *i.e.,* $\mathbf{w} = R\mathbf{v}$ where R is a *random* $N \times |U|$ matrix drawn from some distribution $\mathcal{S}$. Then an estimator V for p^{th} moment of v is computed from $\mathbf{w}$; in the all schemes we consider here, the estimator will simply be $\|\mathbf{w}\|_p^p$.

Of course, since R is also as long as $|U|$, in our setting R is too large to store explicitly. However, notice that 'sketching' a packet x is exactly equivalent to adding the x^{th} column of R to the sketch $\mathbf{w}$. This suggests the following efficient approach to sketching: initialize $\mathbf{w} = 0$, and for every packet x in the stream, generate the x^{th} column of R by hashing the packet with h and adding the hash value $h(x)$ to the sketch $\mathbf{w}$. As long h can generate length N-vectors that are distributed identically to the column vectors of matrices drawn from $\mathcal{S}$, these is exactly equivalent to computing the sketch via an explicit random linear map $\mathbf{w} = R\mathbf{v}$.

Thus, we now have a practical PQM protocol based on p^{th}-moment estimation for $p \geq 1$: Alice and Bob share a hash function h and compute $\mathbf{w}_A = R\mathbf{v}_A$ and $\mathbf{w}_B = R\mathbf{v}_B$ on their streams using the hashing approach described above. Bob then *securely transmits* $\mathbf{w}_B$ to Alice. Since $\|\mathbf{w}_A - \mathbf{w}_B\|_p = \|R(\mathbf{v}_A - \mathbf{v}_B)\|_p$; thus, if the sketches *accurately estimate* the first moment, it suffices to raise an alarm if $\|\mathbf{w}_A - \mathbf{w}_B\|_p^p > \Gamma$ for $\Gamma \in [\alpha T, \beta T]$.

Dealing with adversaries. However, our work is not complete. Recall that we would like our PQM protocol to operate correctly in the presence of adversaries on the path. Thus, we still need to discuss what we mean by the terms *secure transmission* and *accurate estimation* in protocol we described above.

Recall that Eve occupies the path between Alice and Bob, and consider the practical PQM protocol that we described above. In the malicious case, there are a number of ways that Eve could attempt to bias the results of this protocol.

- Eve could try to convince Alice than βT packet drops occurred by altering the sketch $\mathbf{w}_B$ that Bob sends to Alice. Preventing this attack is simple; we shall require that Bob send his sketch $\mathbf{w}_B$ to Alice in a message that is authenticated with a MAC.

- Next, observe that because Eve occupies the path between Alice and Bob, she has the power to choose which packets Bob receives in his stream. Thus, if the adversary can predict the outputs of the hash function h used to map packets to the sketch, she can choose to add and drop packets to Bob's stream in a way that cannot be detected by Alice! For instance, Eve could drop some set of packets and replace them with a different set of packets that map to the sketch in an identical way.

  Typically, the correctness of p^{th}-moment estimation schemes relies on the fact that the randomness used for sketching (*i.e.,* to choose the hash function h) is chosen *independently* of the stream to be sketched. However, in our setting, this is not necessarily the case; if the hash function h is public, then adversary choice of Bob's stream $\mathbf{v}_B$ can *depend* on the randomness used for sketching, h. (This observation was independently made by Mironov, Naor, and Segev [80].) For this reason, we replace the public hash function h used for sketching with a *keyed* hash function h_{k_u}, keyed with a secret key k_u is shared between Alice and Bob, and is refreshed every interval.

3.5.2 The secure sketch protocol

We are finally ready to describe our secure sketch PQM protocol. Our protocol works in intervals. We assume Alice and Bob share a secret (master) key (k_1, k_2), and derive an interval key k_u for each interval u (see Section 3.3). In Appendix B.2, we provide a detailed treatment of techniques that Alice and Bob can use to synchronize their intervals. Within interval u, our secure sketch protocol operates as follows:

1. *(Sketch.)* Alice runs a sketching algorithm, using a keyed hash $h_{k_u}(\cdot)$ keyed with secret interval key k_u to incrementally compute a sketch $\mathbf{w}_A$ of the vector $\mathbf{v}_A$ induced by the packet it sends. Bob similarly uses $h_{k_u}(\cdot)$ to compute sketch $\mathbf{w}_B$ of the vector $\mathbf{v}_B$ induced by the packets he receives.

2. *(Interval End.)* After sending the T^{th} packet in the interval, Alice sends an 'Interval End' message to Bob, authenticated with the master key k_1, and containing her sketch $\mathbf{w}_A$ and the next interval number $u + 1$. She then refreshes her sketch (*i.e.,* sets $\mathbf{w}_B = 0$) and refreshes the interval key (*i.e.,* computes k_{u+1} using a PRF keyed with the master key k_2 as described in Section 3.3).

3. *(Report.)* Upon receiving the 'Interval End' message and verifying the correctness of its MAC, Bob computes the 'difference sketch' $\mathbf{w}_A - \mathbf{w}_B$, and sends a 'Report' message to Alice, authenticated with the master key k_1, containing the 'difference sketch' $\mathbf{w}_A - \mathbf{w}_B$, and the current interval number u. Bob then refreshes his sketch and computes the interval key for the next interval $u + 1$.

4. *(Security Check.)* Upon verifying the MAC on the 'Report Message', Alice uses the difference sketch $\mathbf{w}_A - \mathbf{w}_B$ to compute an estimate V of $\|\mathbf{v}_A - \mathbf{v}_B\|_2^2$ and raises an alarm if and only if $V > \Gamma = 2\alpha\beta T/(\beta + \alpha)$ or if the report is missing or has an invalid MAC.

Our protocol has a number of attractive properties. First, notice the we require the transmission of only two control messages ('Interval End', and 'Report'), and no packet modifications. Second, notice that the 'Security Check' phase can be computed offline. Finally, notice that Alice and Bob need only store single sketch at any given time; at the end of each interval, Alice and Bob immediate transmit their sketches as control messages, refresh their sketches, and begin monitoring a new interval.

3.5.3 Security of the secure sketch protocol

We formalize the intuition of Section 3.5.1 with the following theorem.

Theorem 3.5.1. *Suppose that the sketch algorithm guarantees, that if the hash function used for sketching is chosen randomly and independently of* $\mathbf{v}$*, then with probability at least* $1 - \delta$*, the estimate of the* p^{th}*-moment* $\|\mathbf{v}\|_p^p$ *for* $p \geq 1$ *is within* $(1 \pm \varepsilon)$ *for* $\epsilon = \frac{\beta-\alpha}{\beta+\alpha}$*. Then, the secure sketch protocol is a* (α, β, δ)*-secure PQM protocol as per Definition 3.2.1.*

Proof of Theorem 3.5.1. Consider the *malicious* case. First observe that Eve cannot forge the 'Interval End' or 'Report' control messages, since the control messages are authenticated using a secure MAC (and dropping the report will only cause Alice to raise an alarm). Thus, we shall assume that for both the *benign* and *malicious* case, Alice gets a consistent version of the difference sketch $\mathbf{w}_A - \mathbf{w}_B$ at the end of the every interval.

Now, observe that (a) no effect of the hash function h_{k_u} is visible to Eve until after the interval ends, and (b) k_u is kept secret from Eve and thus chosen (pseudo)randomly and independently of $\mathbf{v}_A$ and $\mathbf{v}_B$. It follows that the sketching algorithm generates a $(1 \pm \varepsilon)$-estimate of V of $\|\mathbf{v}_A - \mathbf{v}_B\|_2$ with probability $1 - \delta$. Thus, letting $\mathbf{x} = \mathbf{v}_A - \mathbf{v}_B$, we have:

1. No false positives: if $D \leq \alpha T$ and $A = 0$, then as discussed in Section 3.5.1 it follows that $\|\mathbf{x}\|_p^p = \|\mathbf{d}\|_p^p = D \leq \alpha T$. Now, with probability $1 - \delta$ we have that the estimate

$$\begin{aligned} V &\leq (1 + \varepsilon)\|\mathbf{x}\|_p^p \\ &\leq (1 + \tfrac{\beta-\alpha}{\beta+\alpha})\alpha T \\ &= \tfrac{2\beta\alpha}{\beta+\alpha} T = \Gamma \end{aligned}$$

2. No false negatives: if $D > \beta T$, then as discussed in Section 3.5.1 it follows that $\|\mathbf{x}\|_p^p = \|\mathbf{d}\|_p^p + \|\mathbf{a}\|_p^p > \|\mathbf{d}\|_p^p > \beta T$. Similarly, with probability $1 - \delta$, it follows that the estimate V is greater than is $(1 - \frac{\beta-\alpha}{\beta+\alpha})\beta T = \frac{2\beta\alpha}{\beta+\alpha} T = \Gamma$.

1. and 2. guarantee that with probability $1 - \delta$ Alice can use the decision threshold Γ to decide between cases where $D < \alpha T$ and $D > \beta T$. □

Recall that $\mathbf{d} \in \{0,1\}^U$ because Alice sends unique packets. A closer look at the proof shows it suffices if the sketch guarantees that (a) the estimate is at most $(1+\varepsilon)\alpha T$ for vectors that have all entries in $\{0,1\}$ and with norm $\|\mathbf{v}\|_p^p \leq \alpha T$, and (b) the estimate is at least $(1-\varepsilon)r$ for vectors $\mathbf{v}$ that have at least $r \geq \beta T$ entries in $+1$ (and possibly other nonzero entries as well). It turns out this observation is crucial for obtaining improved parameters for our protocol; see Theorem 3.5.2 below.

Turning the protocol on and off. In order to reduce resource consumption, it sometimes makes sense for a router to 'turn off' the secure sketching protocol. However, an adversary could take advantage of the fact the protocol is 'off' for certain intervals in order to bias monitoring results, selectively dropping packets when the protocol is 'off', and behaving itself while the protocol is 'on'. Thus, it is crucial to ensure that intervals when the protocol is 'on' indistinguishable from intervals when the protocol is 'off'.

Notice that from Eve's perspective, the only indication that the protocol is 'on' are the two control messages ('Interval End' and 'Report'). Thus, while in an 'off' interval, Alice and Bob need not compute hashes over packet contents or to maintain sketches (so that there are significant savings in storage and computation), we still require the appropriate control messages to be sent. In an 'off' interval, we require (a) Alice to count the number of packets she sends to Bob and send a dummy 'Interval End' message each time the counter reaches T, and (b) Bob to respond with a dummy 'Report' packet. To make the dummy control messages indistinguishable from real control messages, we will also require (c) that *all* information fields in the control messages sent by the protocol are encrypted and padded to a fixed length (and subsequently authenticated).

With this approach, a sender with the resources to run only K instances of the 'secure sketch' protocol, can engage in PQM with $M > K$ receivers by choosing a (pseudo)random set of K of M receivers for which the protocol should be 'on' in a given interval. Note that that selection of 'on' intervals should be *random*, in order to prevent an adversary from selectively attacking the 'off' intervals by using side-channel information (*e.g.,* observing if the sender switches to a new path) to distinguish between which intervals that 'on' or 'off'.

3.5.4 Plugging in sketching schemes

In this section we show how to instantiate our PQM protocol with known p^{th}-moment estimation sketching schemes, such that the schemes satisfy the requirements of Theorem 3.5.1. We will focus on two highly efficient schemes for estimating the *second*-moment: the 'classic' sketching technique [7,5] based on the Johnson-Lindenstrauss lemma, and the more efficient 'CCF' sketch of Charikar, Chen and Farach-Colton [24].

In each scheme, packet hashing can be done with either 4-wise independent hash function or PRF. We consider both cases. While 4-wise independent hash functions are *theoretically* faster than PRFs (see also the discussion in Section 3.3),using these weaker hash function comes at the cost of worse sketch parameters N. Both schemes operate by taking a single pass over the data stream to compute the sketch $\mathbf{w}$, and compute the estimator as $V = \|\mathbf{w}\|_2^2$. When packet-hashing is done with *either* a 4-wise independent hash function *or* a PRF [7,5,100], both schemes have estimators V with expectation $\|\mathbf{v}\|_2^2$ and variance $\frac{2}{N-1}\left(\|\mathbf{v}\|_2^4 - \|\mathbf{v}\|_4^4\right)$.

We next describe each scheme, and show how they compare in terms of update time per incoming packet and storage requirements (*i.e.,* the number of bins in the sketch, N, and the size of each bin). We also derive new bounds for the storage requirements of these schemes.

Classic Second-Moment Estimation Sketches

Alon, Matias, and Szegedy [7] suggest the following approach to sketching: when receiving a packet d map it to a vector $b \in \{-\frac{1}{N}, \frac{1}{N}\}^N$ and add b to the sketch $\mathbf{w}$. Thus, the update time per incoming packet is exactly N, the number of bins in the sketch.

Packet hashing with a 4-wise independent hash. Alon et. al [7] require a hash function such that each entry of b in 4-wise independent, and every entry of b is completely independent of all other entries of b. They then show how to estimate $\mathbf{v}$ within $(1 \pm \varepsilon)$ with probability δ using a sketch with $N \geq \frac{2}{\varepsilon^2 \delta}$ bins.

Packet hashing with a PRF. Achlioptas [5] obtained a bound on N by requiring each entry of b to be computed using an (independent) PRF producing either $+\frac{1}{N}$ or $-\frac{1}{N}$ with probability $\frac{1}{2}$. Achlioptas showed that obtaining an (ε, δ)-approximation of the second moment requires

$$N > \frac{12}{\varepsilon^2} \frac{1}{3-2\varepsilon} \ln \frac{1}{\delta} \tag{3.12}$$

bins in the sketch. Notice the that PRF approach requires $O(\log \frac{1}{\delta}/\delta)$ less storage then the 4-wise independent hashing approach.

Sizing each bin in the sketch. To prevent overflow, we can take each bin in the sketch to hold integers in $[-K, +K]$ where $K = \sqrt{2T \ln(\frac{200N}{\delta})}$, so that each bin requires $1 + \log_2 K$ bits of storage. To see why, suppose that when we store the sketch, we drop the $\frac{1}{N}$ factor. We now find K such that the probability that each bin overflows is at most $\frac{\delta}{N}\frac{1}{100}$. If X_i is an indicator variable that equals 1 with probability $\frac{1}{2}$ and -1 otherwise, then the count in each bin is the random variable $X = \sum_{i=1}^{T} X_i$. Then, from the Chernoff bound we have that $\Pr[\ |X| \ \geq \ K\ ] \ \leq \ 2\exp(-\frac{K^2}{2T}) \ \leq \ \frac{\delta}{100N}$. Finally, we get $K = \sqrt{2T \ln(\frac{200N}{\delta})}$. (We can also change the protocol to raise an alarm if any bin overflows, since this will happen with low probability in the benign case.)

CCF second-moment estimation sketch.

The sketch of Charikar, Chen, and Farach-Colton [24] can be adapted [100] to give a second-moment estimation algorithm with a faster update time; instead of updating *all* N bins each time a new packet arrives as in the classic sketch, the CCF scheme only updates a *single* bin. In our context, the CCF update algorithm requires that each incoming packet d is hashed to a pair (i, b) where $i \in [N]$ and $b \in \{\pm 1\}$, and b is added to the i^{th} bin in the sketch $\mathbf{w}$. To prevent overflow in each bin, we take each bin to hold integers in $[-K, +K]$ where $K = 2\sqrt{\frac{T}{N} \ln(\frac{200N}{\delta})}$. Thus, we require

$$1 + \tfrac{1}{2}\log_2\left(4\tfrac{T}{N}\ln(\tfrac{200N}{\delta})\right) \quad \text{bits / bin} \tag{3.13}$$

To obtain (3.13), we find K such that the probability that each bin overflows is at most $\frac{\delta}{N}\frac{1}{100}$. If X_i is a random variable that equals 1 with probability $\frac{1}{2N}$, -1 with probability $\frac{1}{2N}$, and 0 otherwise, then the count in each bin is the random variable $X = \sum_{i=1}^{T} X_i$. Then, adapting the Chernoff bound that appears in Levchenko [71], we have that $\Pr[\ |X| \ \geq \ K\ ] \ \leq \ 2\exp(-\frac{K^2}{4T\mathrm{VAR}[X_i]}) \ \leq \ \frac{\delta}{100N}$. Finally, we get $K = 2\sqrt{\frac{T}{N} \ln(\frac{200N}{\delta})}$ since $\mathrm{VAR}[X_i] = 1/N$.

Packet-hashing with 4-wise independent hashes. In Appendix B.5.1 we show that if the incoming packet d is hashed with two independent 4-wise independent hash functions, (where i is computed using a 4-wise independent hash with output domain $[N]$ and b is computed using a 4-wise independent hash with output domain $\{-1, 1\}$), then we require at $N \geq \frac{2}{\varepsilon^2 \delta}$ bins in our sketch.

Packet-hashing with a PRF. In the general case, the faster update time of CCF comes at the cost of increased storage. More precisely, in order to get a $(1 \pm \epsilon)$ accuracy with probability $1 - \delta$, the CCF schemes require a larger $N = \Theta(\frac{1}{\delta\epsilon^2})$, rather than $N = \Theta\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$ of the classic scheme. Unfortunately, this increased storages is generally required even if hashing is performed with a truly random function![6] To see why, consider the following counterexample: consider the vector $\mathbf{v} = 10^{10}e_x + 10^{10}e_{x'}$ where e_x is the vector with 1 in coordinate x and zero elsewhere, and $x \neq x'$. Then $\|\mathbf{v}\|_2^2 = 2 \cdot 10^{20}$, but with probability $1/2N$ a sketch of $\mathbf{v}$ will be 0.

Fortunately, our setting has special properties that allow the CCF scheme to avoid incuring the cost of increased storage! We now prove that in our setting the CCF scheme we can have $N = O\left(\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\right)$. This follows because (a) we assume that Alice sends unique packets, and (b) we only care about deciding whether $\|\mathbf{v}\|_2^2$ lies above or below a threshold, rather than getting an accurate estimate of $\|\mathbf{v}\|_2^2$. (See also the discussion after Theorem 3.5.1.)

Our theorem supposes that packet hashing is performed using an two independent random function: one to chose $i \in [N]$ and another to choose $b \in \{-1, 1\}$. When the CCF algorithm uses a random function for hashing, we can think of the sketch $\mathbf{w}$ as computed from the characteristic vector $\mathbf{v}$ via a random linear map, *i.e.,* $\mathbf{w} = R\mathbf{v}$, where R is chosen *uniformly at random* from set $\mathcal{S}_{\text{CCF}}$. For the CCF algorithm, $\mathcal{S}_{\text{CCF}}$ is the set of $N \times |U|$ matrices where each column has ± 1 in some row and zeros everywhere else. Hence, we have the following theorem:

Theorem 3.5.2. *For any vector $\mathbf{v} \in \mathbb{Z}^U$, choose the $N \times U$ matrix S uniformly from $\mathcal{S}_{\text{CCF}}$ and set $\mathbf{w} = S\mathbf{v}$. Then, for all $\varepsilon \in [0, 1)$ and η such that $\left(\frac{1-\eta}{1+\eta}\right)^2 = \max\left(\frac{1+\frac{\varepsilon}{2}}{1+\varepsilon}\ ,\ \frac{1-\frac{3\varepsilon}{4}}{1-\frac{\varepsilon}{2}}\right)$, choosing*

$$N \geq \tfrac{24}{\varepsilon^2}\ln\tfrac{2}{\delta} \tag{3.14}$$

$$q, r \geq \tfrac{3N}{\eta^2}\ln\tfrac{4N}{\delta} \tag{3.15}$$

ensures that the following two items occur with probability at least $1 - \delta$:

1. *If $\mathbf{v} \in \{-1, 0, 1\}^U$, and $\|\mathbf{v}\|_2^2 \leq q$, then $\|\mathbf{w}\|_2^2 < (1 + \varepsilon)q$.*
2. *The number of* non-zero *entries in $\mathbf{v}$ is r, then $\|\mathbf{w}\|_2^2 > (1 - \varepsilon)r$.*

See Appendix B.5.2 for Theorem B.5.4 a tighter and more precise statement of Theorem 3.5.2, as well as its proof. The theorem bounds the number of bins in the sketch, N, as well as both the number of non-zero elements in $\mathbf{v}$. The fact that the number of bins in the sketch, N, must be large is not so surprising. However, our proof also needs $\mathbf{v}$ to have many non-zero elements because CCF does not work as well when very sparse vectors $\mathbf{v}$ cause high variance in the number of entries in the bins of $\mathbf{w}$ (see for instance the counterexample we discussed above). This condition on $\mathbf{v}$ holds in our setting because the number of bins in the sketch is much smaller than the total number of packets. Similar conditions apply in many other sketch applications; thus, we believe that this theorem may be of independent interest.

Applying Theorem 3.5.2. To apply the theorem into our setting, assume that the PRF used for packet hashing is indistinguishable from a random function. Then, set $\varepsilon = \frac{\beta-\alpha}{\beta+\alpha}$, set $\mathbf{x} = \mathbf{v}_{\text{A}} - \mathbf{v}_{\text{B}}$, and set $q = \alpha T$. The false positive condition is satisfied because we have $\mathbf{x} \in \{0, 1\}^U$ and $\|\mathbf{x}\|_2^2 = |\mathbf{x}|_1 = D \leq \alpha T$, so with probability $1 - \delta$,

$$V = \|\mathbf{w}\|_2^2 < (1 + \varepsilon)\|\mathbf{x}\|_2^2 \leq (1 + \varepsilon)\alpha T = \tfrac{2\alpha\beta}{\alpha+\beta}T$$

[6] CCF's [24] sketch can attain better success probability (even with 4-wise independent hashing) by using the median of estimates obtained from M independent sketches, for some number M. However, this increases the storage and update time by a factor of M.

Scheme	Packet Hashing	N, Bins in Sketch	Bits/bin
Classic	4-wise independent	$\frac{2}{\varepsilon^2\delta}$	$1+\frac{1}{2}\log_2\left(2T\ln(\frac{200N}{\delta})\right)$
	PRF	$\frac{12}{\varepsilon^2}\frac{1}{3-2\varepsilon}\ln\frac{1}{\delta}$	
CCF	4-wise independent	$\frac{2}{\varepsilon^2\delta}$	$1+\frac{1}{2}\log_2\left(4\frac{T}{N}\ln(\frac{200N}{\delta})\right)$
	PRF[7]	$\frac{24}{\varepsilon^2}\ln\frac{2}{\delta}$	

Table 3.1: (Analytically-derived) parameters for secure sketch PQM.

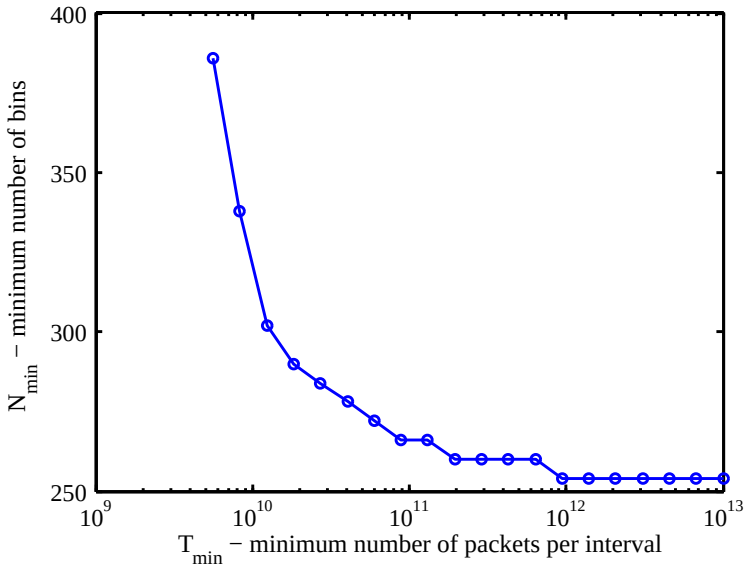


Figure 3.4: Theorem B.5.4 is used to obtain bounds on sketch size, N for a given choice of T_{min}, the minimum number of packets per interval. Here $\delta = \beta = 2\alpha = 1\%$.

The false negative condition is satisfied because we have the number of drops is $r > \beta T$. So, with probability $1-\delta$, we get that

$$V = \|\mathbf{w}\|_2^2 > (1-\varepsilon)r \geq (1-\varepsilon)\beta T = \frac{2\alpha\beta}{\alpha+\beta}T$$

where the first inequality comes from the fact that $\mathbf{x} = \|\mathbf{d}\|_p^p + \|\mathbf{a}\|_p^p$ and $\|\mathbf{d}\|_p^p$ is a $\{0,1\}$-vector with r entries that are $+1$.

Some sample parameters and experiments

In the following, we use the following sample parameters: We suppose the detection threshold is $\beta = 0.01$, the false alarm threshold is $\alpha = \beta/2$ and about $T = 10^7$ packets are sent during an interval. We will require a confidence of $1-\delta = 99\%$.

Comparing analytic results. Combining these sample parameters with the analytic results summarized in Table 3.3, we see that when 4-wise independent hashing is used, both classic and CCF sketching require $N = 1800$ bins in the sketch. However since CCF requires only 10 bits/bin compared to the 16 bits/bin required for classic sketching, we see that the CCF

[7]This analytic bound on N also requires that $\alpha T > \frac{3N}{\eta^2}\ln\frac{4N}{\delta}$ for η such that $\left(\frac{1-\eta}{1+\eta}\right)^2 = \max\left(\frac{1+\frac{\varepsilon}{2}}{1+\varepsilon}\ ,\ \frac{1-\frac{3\varepsilon}{4}}{1-\frac{\varepsilon}{2}}\right)$. See Theorem 3.5.2.

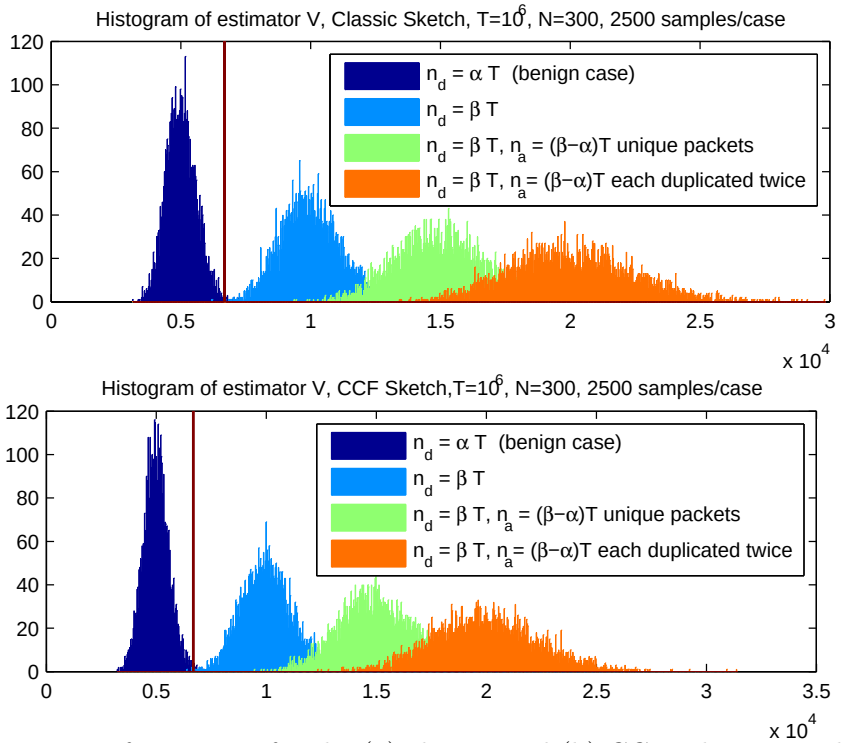


Figure 3.5: Histogram of estimator for the (a) classic, and (b) CCF schemes, each using packet-hashing with a PRF and with $N = 300$, $T = 10^6$, $\beta = 2\alpha = 1\%$ and threshold $\Gamma = 6667$. Histogram computed via numerical experiments.

	Bins in Sketch	**Sketch Size**				
β/α	**N**	$T = 10^4$	$T = 10^5$	$T = 10^6$	$T = 10^7$	$T = 10^8$
2	128	128B	144B	176B	208B	208B
4	64	64B	88B	88B	104B	104B
8	32	36B	48B	48B	52B	52B
16	32	36B	48B	48B	52B	52B
32	32	36B	48B	48B	52B	52B
64	32	36B	48B	48B	52B	52B

Table 3.2: Minimum N bins per sketch, when N is taken as a power of 2, computed via numerical experiments for PQM using CCF with a PRF for packet hashing. Sketch size is computed by multiplying experimentally-obtained value for N with the value obtained from equation (3.13). We fix $\beta = \delta = 1\%$.

requires smaller sketches (3.6KB sketch for classic, 2.25KB for CCF sketching). Storage become noticeably smaller when we use a PRF. From Table 3.3, when a PRF is used we find that the classic scheme requires $N = 214$ bins with 15 bits/bin for a sketch of size 400B. For the CCF scheme, we can apply the refined version of Theorem 3.5.2 in Appendix B.5.2 to obtain bounds on N, the number of bins in the sketch, for different values of T_{min}, the minimum number of packets per interval. We did this in Figure 3.5.4 for $\beta = \delta = 2\alpha = 1\%$, and we found that if there are at least $T_{min} = 1.2 \times 10^{10}$ packets in the interval, we can use $N = 300$ with counters of $b = 14$ bits if we take intervals containing at least $T = 10^9$ packets.

Our Theorem 3.5.2 for CCF with a PRF introduces an awkward bound on $T_{\min}$, the minimum number of packets that must be sent per interval. However, we believe that this bound is an artifact of our proof technique. As we discuss below, our numerical experiments for CCF with a PRF indicate (though do not conclusively prove) that even $N = 300$ bins suffices even if we use much shorter interval lengths, for instance $T = 10^4$.

Numerical experiments: histograms. Figure 3.5 is a histogram of the classic and CCF estimators V for (from left to right) the benign case where $D = \alpha T$ (here we want the estimator to be below the threshold Γ so that Alice does not raise an alarm), and for three cases where $D = \beta T$ so we want Alice to raise an alarm: a case where Eve does not add any packets, a case where Eve adds $(\beta - \alpha)T$ distinct packets, and a case where Eve adds $(\beta - \alpha)T$ total packets where each packet is duplicated twice. Notice that the threshold Γ clearly distinguishes between cases where $D = \beta T$ and the benign cases where $D = \alpha T$. Also, notice if Eve adds packets to the link, she only increases the probability that Alice raises an alarm, as predicted by equation (3.11). Figure 3.5 also suggests that taking $N = 300$ suffices for CCF even if we have shorter interval lengths of $T = 10^6$.

Numerical experiments: CCF with a PRF. We further studied the CCF with PRF approach by performing a number of numerical experiments to determine N, the number of bins in the sketch. In every experiment, we chose N as a power of 2.[8]C code for these experiments is available by request. Our results are presented in Figure 3.2. Firstly, from Figure 3.2 we see that N varies with the ratio β/α, which confirms the analytic results summarize in Figure 3.5.4. However, our numerical experiments suggest that as long as there are $T > 1/\alpha$ packets/interval, the choice of T does not really impact the value of N; for a given β/α ratio, the minimum choice of N as power of 2 was the same for any value of T ranging from 10^4 to 10^8. Next, Figure 3.2 indicates that sketch size grows with T; this growth is logarithmic in T (as expressed by the equation for the number of bits / bin for CCF in equation (3.13)).

Reducing computation with pre-hashing

In our setting, hash function computation can be expensive because we need to compute a PRF (or 4-wise independent) hash over up to the entire packet, which may be up to 1500 Bytes long. While Appendix B.1 discusses generic techniques for fast per-packet PRF computation based on efficient ε_g-almost universal hash functions, it turns out that we can have even faster hashing constructions for our sketching protocols.

To do this, we again use ε_g-almost universal hash functions. We reduce the cost of PRF computation by first mapping packets from U to a short n_1-bit string using the efficient ε_g-almost universal hash function, and then using a PRF (or 4-wise independent) hash to map from n_1-bits to the sketch. Thus, if n_1 is sufficiently small, this means that our PRF can

[8]We take N to be a power of two because this makes packet hashing in the CCF sketch more convenient. That is, if $N = 2^\eta$ and we use PRF that produces η (pseudo)random bits, then the binary representation of these η bits uniformly choose an element of $[N]$. However if N is not a power of 2, a more complicated mapping of these η bits is required to uniformly choose an element of $[N]$.

be constructed using a *single* invocation of a block cipher like AES. (Similarly, our 4-wise independent hash need only operate on small number of inputs, making it possible to realize using only *e.g.,* three n_1 bit multiplications.). In Appendix B.4, we adapt the analysis in [100] to show that an (α, β, δ)-secure PQM protocol requires a ε_g-almost universal hash function with

$$\varepsilon_g \leq \frac{\delta}{10^3 T} \frac{\beta - \alpha}{\beta + \alpha} \tag{3.16}$$

Sample parameters using GHASH. Consider using GHASH [78] as our ε_g-almost 2-wise independent hash function. Suppose GHASH produces outputs of length n_1, and each packet is at most 1500B long, and block lengths are m, where m is taken as a power of two. Since $\varepsilon_g = \frac{1500 \cdot 8}{m} 2^{-n}$, for $T = 10^7$ packets/interval, and $\delta = \beta = 2\alpha = 1\%$, applying (3.16) we find that it suffices to choose GHASH with $n = m = 64$ bits. This choice of n_1 is quite short! (For most other applications, GHASH requires block lengths of $m = 128$ bits. Indeed, in Appendix B.1 we found that we needed $m = 128$ bits blocks when we analyze it as part of the PRF.) Thus, it follows that (a) hardware implementations of GHASH will be fast, and (b) PRF computation (to map n_1-bit strings to the sketch) amounts to a single invocation of AES.

Other sketches.

TZ Sketch. Thorup and Zhang [100] gave a variant of the CCF scheme where, instead of updating a bin in the sketch with a randomly chosen element in $\{\pm 1\}$, the bin is always updated with a $+1$. (While the update algorithm in TZ is as in the Count-Min sketch [27], the analysis there is different.) However, their second-moment estimation scheme requires a larger bin size (roughly twice the number of bits/bin) than CCF, so we don't consider this scheme any further.

Other p^{th}-moment estimation schemes. As discussed in Section 3.5.1, any p^{th}-moment estimation protocol, for $p \geq 1$, can be plugged into our protocol to achieve PQM. However, it turns out that the algorithms for second-moment estimation are preferable in our setting because they have the simplest packet-hashing algorithms. For instance, for the classic and CCF sketches, the packet-hashing algorithm amounts to choosing integers in $\{-1, 1\}$ and thus can be efficiently implemented in high speed routers. *c.f.,* with first moment estimation protocols that require choosing real numbers from a Cauchy distribution [63].

Relationship to the adversarial sketch model. In concurrent work, Mironov et.al. [80] considering a setting which Alice and Bob are required to sketch *adversarially-chosen* sets, and then compute metrics on their sets after exchanging sketches over a secure channel; their model maps directly to our PQM model, where Alice and Bob's sets (*i.e.,* packet streams) may be chosen adversarially, and then sketches are exchanged via an authenticated channel. Our work deals with the fact that streams are chosen adversarially by requiring Alice and Bob to compute their sketches using shared secret keys. However, Mironov et.al require that sketching is performed *without* any shared randomness.[9] Their approach has significant advantages, including reduced key-management overhead, and extensions to the client-server setting. For instance, a server can engage in a sketching protocol with multiple clients without using a shared key for each, and then uses a report authenticated using a public key as in the asymmetric protocol of Section 3.4.2. Indeed, following the discussion in Section 3.5.1, any protocol for p^{th}-moment estimation (where $p \geq 1$) in the adversarial sketch gives an PQM protocol (for the client-server setting) as well. However, the lack of shared randomness in the model of Mironov

[9] In Section 3.6, we argue that PQM protocols require shared randomness; the existence of Mironov et.al's protocol does not contradict this. As discussed in Section 3.6, if we used Mironov et.al. sketching results in a PQM protocol, we would require shared randomness to cryptographically authenticate the report messages (containing the sketches) sent from Bob to Alice.

*et al.*comes at a significant cost; they show that any moment estimation protocol for sets of size T requires at least $\Omega(\sqrt{T})$ storage at Alice and Bob. Thus, these protocols are much less efficient than the $O(\log T)$-storage *keyed* sketches that we considered here.

Summary

For large interval lengths T, our analytic results show the most efficient (in terms of storage and update time) instantiation of our secure sketch protocol uses CCF's second-moment estimation scheme [24] with a PRF for packet hashing. Furthermore, our numerical experiments suggest that this is the case even for smaller values of T. While in theory, using a PRF instead of 4-wise independent hash is more computationally expensive, in practice, this not usually the case (see discussion in Section 3.3). Furthermore, we also showed how to reduce the computational cost of computing a PRF on each packet by pre-hashing packets with an ε_g-almost universal hash function, and then applying a single invocation of a fast PRF.

3.6 Necessity of cryptography

All of our protocols require keys between participating nodes, and cryptographic computations. We now show that this overhead is inherent by arguing that *any* PQM protocol satisfying Definition 3.2.1 requires a key infrastructure and the invocation of cryptographic operations. These results also immediately imply that any PQM protocol that does not use keys or cryptography, *e.g.,* Listen [99], is insecure according to Definition 3.2.1.

3.6.1 Keys are necessary

We argue that there must be some form of shared secret information between Alice and Bob. To see that keys are necessary, we argue in the contrapositive: suppose Bob has no secrets from Eve. Then, since Eve occupies a node on the path between Alice and Bob, she receives the same information that Bob receives and can compute the same responses. It follows that Eve can simply run the PQM protocol on her own (responding to Alice with the appropriate acks or reports), and then drop all the packets going to Bob. This breaks security because Alice has no way to know that anything went wrong. Notice further that this suggests that Alice needs Bob's participation in order to run a secure PQM protocol.

We emphasize that this argument only proves that one of the parties (Alice or Bob) has some secret, while the other party holds some information that depends on that secret. For instance, Alice and Bob could share symmetric keys, or Bob might have a public-private key pair (PK, SK) while Alice has the public key PK,

We further remark that the necessity of keys holds only if Eve has the power to *add* packets. However, we believe that it is unrealistic to assume that the adversary cannot add even a single packet; indeed, the security of some protocols (*e.g.,* our secure sketch protocol) can be broken if the adversary successfully forges (*i.e.,* adds) a single 'Report' packet!

3.6.2 Cryptography is necessary

We now argue that the keys must be used in a "cryptographically-strong" manner. Note that our previous result that keys are necessary does not imply that cryptography is necessary; for example [33] uses secret keys in a non-cryptographic way and obtains a protocol that is not secure by our definitions. To show that cryptography is necessary, we show that any secure PQM protocol is at least as complex as a secure *keyed identification scheme (KIS)*, which is known

to be equivalent to many cryptographic tasks like encryption and message authentication [61]. Intuitively, our result follows from the fact that in order for Alice to believe Bob, she must be assured that all the information she is getting indeed came from Bob in a way that Eve cannot impersonate.

A Keyed Identification Scheme (KIS) is a challenge-response protocol in which the two parties share a secret key, and Alice wants to verify Bob's identity. To do this, Alice typically sends Bob a challenge, that Bob must respond to using his secret key. A KIS is secure if Percy, an impersonator who eavesdrops on the interactions between Alice and Bob but does not know the secret key, cannot impersonate Bob by coming up with a correct response to the challenge (with probability better than just randomly guessing the response).

We use a *reduction* to prove that *any* PQM scheme that is secure according to Definition 3.2.1 is at least as complex as KIS. First, we show that given any secure PQM protocol, we can construct a secure KIS. The construction is simple: the challenge in the KIS are the T packets that Alice sends to Bob during an interval of the PQM protocol. The correct response in the KIS is the acks/reports that Bob sends to Alice during an interval of the PQM protocol. Next, we show that if the PQM scheme used in the above construction is secure according to Definition 3.2.1, then our KIS construction is also secure. We do this in contrapositive, by showing that if there existed an efficient adversary Percy that breaks the security of this KIS construction, then Percy can be used to construct an adversary Eve that breaks the security of the PQM protocol. To do this, we show how Eve can break the security of the PQM protocol if she is given access to Percy: First, whenever Percy wants to eavesdrop an interaction between Alice and Bob, Eve lets Percy observe an interval of the PQM protocol. Next, when Percy is ready to impersonate Bob, Eve gives the T packets that Alice sends to Bob to Percy as his KIS challenge, but now, instead of forwarding Alice's packets on to Bob, Eve *drops* T packets and instead responds to Alice with Percy's KIS response. The proof follows from the fact that Alice will not raise an alarm (and therefore Eve breaks the security of the PQM protocol) whenever Percy produces a successful response the challenge in the KIS (and therefore breaks the security of the KIS).

Remark. One could hope for the stronger statement that some kind of cryptographic operation is necessary for *every packet sent by Alice*. However, this is false. Indeed, consider a secure sketch PQM protocol that uses the first-moment estimation protocol of Mironov *et al.* [80], as discussed in Section 3.5.4. Then, we have a PQM protocol that uses no cryptographic computations for packet hashing, and only uses a two cryptographic operations per interval, *i.e.,* computing the MACs on the 'Interval End' and 'Report' packets sent the end of the interval. We suspect that proving a statement of this form (*i.e.,* a cryptographic operation is required for each packet sent) would also restrictions on the storage at Alice and Bob.

3.7 Comparison of Protocols

Because we want PQM protocols that can be deployed in high-speed routers, we have focused on efficiency considerations; namely, we evaluated our protocols' efficiency in (a) *communication overhead*, (b) *computation* of cryptographic operations, and (c) use of dedicated *storage* in the router. We now explore a wider space of design objectives for evaluating our PQM protocols, discuss how our three protocols perform under these objectives, and compare them with two existing solutions for PQM: Stealth Probing [12] and IPsec. We argue that obtaining PQM protocols that perform well for one particular objective often involves trading off some other objective.

3.7.1 A broader space of design objectives

Marking packets. We prefer protocols that do not modify any packets sent by the source edge-network, *e.g.,* by packet marking or encryption. This approach has the advantage of allowing the PQM protocol to be backwards compatible with IP, not increasing packet size, minimizing latency in the router, and allowing the source to turn the PQM protocol on and off without having to coordinate with the destination. Furthermore, avoiding packet marking also means we can implement the PQM protocol in a monitor located off the critical packet-processing path in the router.

Estimating delay. We prefer protocols that allow Alice to estimate round-trip delay, without making assumptions about the clock synchronization between Alice and Bob.

Feedback latency. We prefer protocols that perform well for small interval lengths, so that Alice need only send a small number of packets before she has sufficient information to decide whether or not to raise an alarm. In general, due the high variance in network conditions, it is better to avoid making routing decisions using measurement made over short timescales [94]. However, an PQM protocol that provides fast feedback empowers the edge network to react quickly when situations are particularly dire (*i.e.,* when a path fails completely). Furthermore, fast feedback can be used to detect transient faulty conditions, and can be used when enforcing SLAs to ensure that repeated, short periods of poor performance are not detected because the PQM protocol uses large interval lengths.

Client-server v.s., *peers.* We consider both (a) the *peer* setting, where the source and destination can devote equivalent computational resources to the protocol, (*e.g.,* a corporation that wants to ensure availability between a pair of sites in geographically-disparate locations), and (b) the *client-server setting*, where one party can devote more resources to the protocol (*e.g.,* a client wanting to ensure that his packets are correctly delivered at a web server).

Symmetric vs. public keys. Per our negative results in Section 3.6, all of our protocols require some sort of cryptographic key infrastructure. However, there are many settings, (*e.g.,* when a client has only a very short connection with a web server), where we prefer to design protocols that do not require a handshake protocol between each source-destination pair in order to generate a symmetric key. Furthermore, when one edge network runs PQM protocols with *multiple* other edge networks, it is extremely useful to have protocols that allow an end-point run concurrent PQM protocols using a *single key* (*e.g.,* a public key). This way, the edge network need not lookup a key each time he sends/receives a packet. Such protocols are also particularly useful for *multicast communications.*

Detecting traffic discrimination. Recently, there have been cases of ISPs that degrade performance for certain classes of unwanted traffic like Skype [85] or BitTorrent [1]. Thus, we prefer protocols that can be adapted to determine if a path is selectively dropping specific classes of traffic.

Symmetric vs asymmetric paths. Our PQM protocols are designed to ensure that Alice raises an alarm when the performance of the forward path (from Alice to Bob) degrades unacceptably. However, consider a situation where the performance of the forward path is acceptable, but Alice still raises an alarm because the adversary was tampering with messages sent on the reverse path (from Bob back to Alice). Our protocols *do not* protect against such situations; indeed, to design PQM protocols that give this guarantee, we would either need to assume that source and destination have an out-of-band communication channel that cannot be attacked by the adversary, or consider running PQM protocols over multiple alternate paths. Notice that when the forward path and reverse paths are identical, *i.e.,* symmetric paths, Eve has no incentive to drop acknowledgments and reports; doing this simply makes the path she occupies

look worse. In contrast, with asymmetric paths, an adversary occupying only the reverse path may have an incentive to drop acknowledgments and reports, perhaps to confuse the source into thinking that the forward path is faulty.

However, some of our PQM protocols contain clues that Alice can often use to distinguish between situations where the forward path is actually faulty, and when an adversary on the reverse path is simply dropping reports.

Monotonicity. We say that that a protocol is *monotone* if Helen cannot trick the source into detecting faults on the data path simply by adding packets to the path. To see why this important, consider an adversary, Helen, that does *not* occupy a node on the data path and thus cannot drop or delay packets, but can *inject* packets onto the data path. Helen might have an incentive to trick Alice into raising an alarm this in order to force the Alice to switch her traffic to a different path. In practice, no protocol is completely monotone, since Helen can always cause a denial-of-service attack by flooding the path with nonsense packets and exhausting the computational resources of Alice or Bob. However, we typically want to avoid protocols where Helen can trick the source into detecting a failure (when all packets were delivered) because of additional packet injections.

3.7.2 Evaluating the tradeoffs

We now discuss how each of our three protocols fits into the tradeoff space we described above. This discussion is summarized in Table 3.3.

Secure sketching. Our secure sketch protocol makes extremely efficient use of storage and communication. Furthermore, these requirements are (roughly) independent of the threshold chosen, and so can be used even to detect very small degradations in path performance. On the other hand, the secure sketch protocol does not allow us to easily measure round trip time, since packets are aggregated into one sketch. It requires both the sender and the receiver to maintain keys and (small) storage, which might be a problem in the client/server setting where a server is communicating with many clients, and does not want to maintain per-client storage for the purposes of running PQM protocols. Finally, the sketch protocol is not *monotone*: it will raise an alarm if many packets are *added* into the path, even if no packet is actually dropped. This could be an issue if an adversary that does *not* sit on the path is able to inject packets into the path.

Secure sampling. Our secure sampling protocols are best suited for situations where Alice needs immediate feedback and accurate measurements of round-trip delay (which she can easily obtain, even in the absence of synchronized clocks, by timing the arrival of acks). Furthermore, the protocols are *monotone* in the sense that if an adversary adds packets to the path or spoofs acks, Alice can simply ignore all the acks that do not correspond to the packets that she sent. *Symmetric Secure Sampling* is best suited when Alice and Bob are peers that have equal resources to devote to the protocol. Furthermore, the protocol is best when we do not want to make any clock synchronization assumptions, or when we want fast feedback (which can be obtained by adjusting the probe frequency p appropriately, see Section 3.4.3). *Asymmetric Secure Sampling* is best suited for the client-server setting, where the server wants to run PQM protocols with many clients without using dedicated storage and using only a single key for all clients.

However, the sampling protocols (save for the TSSS protocol of Section 3.4.2) have a disadvantage in the *asymmetric path* setting— when the forward (Alice to Bob) path is not the same as the reverse (Bob to Alice) path. The reason is that since only a p-fraction of sent packets are acknowledged, each dropped ack looks like $\frac{1}{p}$ dropped packets. Thus, in the asymmetric path setting, an adversary on the reverse path can arbitrarily increase the source's estimate of the

	Secure Sampling		Secure
	Sym	**Asym**	**Sketching**
Storage/Communication[10]	90KB	240–840KB	0.2–2.3KB
Peer setting	✓		✓
Client-server setting		✓	
No clock sync	✓	coarse	✓
Estimates delay	✓	✓	
Fast feedback	✓		
Monotonicity	✓	✓	

Table 3.3: Tradeoff space for our protocols.

failure rate on the forward path by dropping acks. In contrast, in the secure sketch protocol only a single authenticated report packet is sent on the reverse path, and so if it does not arrive Alice can deduce that the problem is in the reverse rather than the forward path (unless the forward path is completely blocked and Bob is not even aware of Alice's existence). This issue also means that the sketch protocol is better suited for SLA-compliance monitoring applications, especially in the asymmetric paths setting (where the report packet could even be sent out-of-band). When PQM is used to inform routing decisions in the asymmetric setting, Alice and Bob can always coordinate switching their forward and reverse paths once an alarm is raised.

IPsec. IPsec is a standard for symmetric-key encryption and authentication of packets at the network layer. However, it requires invoking a cryptographic operation, modifying, and adding tags to every packet sent on the path, which could be quite expensive when operating at multi Gbit/sec rates. Also, IPsec currently does not include a standard for providing authenticated acknowledgments and so needs additional machinery, like *Stealth Probing* [12], in order to provide secure PQM at the network layer. On the other hand, if we perform PQM at a higher layer, we can use TCP over IPsec (so that we have authenticated acknowledgments for every single packet sent) or even SSL. These protocols provide very strong security guarantees; they not only provide confidentiality, but also allow a source to detect if a failure occurs for *every single packet it sends*. But given the high cost associated with these guarantees, these protocols are arguably, more appropriate when confidentiality and integrity are necessary for other reasons, or when PQM functionality is required at the end-host, rather than in the high-speed routing setting that we focus on here.

Stealth Probing. Stealth Probing [12] is a network layer protocol that provides statistically-secure path-quality monitoring (satisfying Definition 3.2.1) by designating specific packets as 'probes' that must be ack'd by the destination, and then masking the choice of probe by encrypting and authenticating all traffic using IPsec. This protocol shares many of the traits of our symmetric secure sampling protocol. However, it incurs the extra overhead of encrypting all traffic, and is probably best suited when confidentiality is required in addition to PQM in the peer setting.

Note that all our protocols can be tuned to measure the performance on a particular subset of the traffic, for the purposes of detecting whether some intermediate nodes treat certain packets (such as Skype [85] or BitTorrent [1]) differently than others. The same is true for IPsec based solutions such as Stealth probing. In fact, the latter solutions make selective (mis)treatment of packets by the adversary much harder, as they encrypt all traffic.[11]

[10] Storage and communication are given for an interval of $T = 10^7$ packets with $\beta = 0.01$, $\alpha = \beta/2$, and $1 - \delta = 99\%$.

[11] Of course, if packets are encrypted but not padded to a fix constant length, an adversary can still selectively mistreat certain packets based on their length. Furthermore, encryption does not prevent the adversary from using timing attacks to discriminate between packets, see *e.g.,* [97].

Application layer protocols. Here we did *not* consider protocols that detect packet loss at the application layer by using the semantics of packet contents (*e.g.,* the fact that a webpage does not display correctly). We only considered protocols that operate at the application layer, and assume nothing about packet contents (apart from the fact that packets are unique in a given interval). While protocols that leverage packet semantics are more appropriate for certain settings, we do not consider them here because we would like to design general-purpose protocols that operate *inside high-speed routers* to inform routing decisions or provide fast feedback about SLA violations.

3.8 Conclusion

In this chapter, we have designed and analyzed efficient path-quality monitoring protocols that give accurate estimates of path quality in a challenging environment where adversaries may drop, delay, modify, or inject packets. Our protocols have reasonable overhead, even when compared to previous solutions designed for the *non-adversarial* settings, and all except TSSS do not modify data packets in any way. In fact, one possible deployment scenario for our protocols is to start by deploying protocols that use hash functions with publicly-known keys, to monitor path quality in manner that is robust to non-adversarial failures such as congestion, misconfiguration, and malfunctions. Then, the same router support could be leveraged, using secret keys, to operate in an adversarial setting as needed. Another possibility is to use our protocols with publicly known keys, but combine them with IPsec for paths where protection against adversarial nodes is required; this will be secure, albeit at a higher overhead than using our protocols on their own. We believe that our PQM protocols, and our associated models of their properties, are valuable building blocks for designing future networks with predictable security and performance.

Chapter 4

Path-Quality Monitoring: Failure Localization

4.1 Introduction

The Internet is an indispensable part of our society, and yet its basic foundations remain vulnerable to attack. Secure routing protocols seek to remedy this by not only providing guarantees on the correct setup of paths from sender to receiver through a network (*e.g.,* Secure BGP [66]), but also by verifying that data packets are actually delivered correctly along these paths. Packet delivery is surprisingly susceptible to simple attacks; in the current Internet, packets are typically sent along a single path from sender to receiver, and so a malicious node along the data path can easily drop or modify packets before they reach their destination. While small amounts of random packet loss are considered to be a natural part of the Internet's operation, there are many situations in which a sender would like to detect and respond to unusually high rates of packet loss or corruption along a path. To this end, the networking community has recently been studying monitoring and measurement protocols that return information about packet loss events on a data path (*e.g.,* [28, 33, 76, 99, 86, 13, 11, 81, 10]). The motivation for these protocols is twofold. First, they provide the sender with information that he can use during path setup to select a single, high-performance path to the receiver from the multiple available paths through the network [55]. Second, since Internet service is a contractual business, where senders pay nodes along the data path to carry their packets, information from Internet measurement protocols is highly valuable for enforcing contractual obligations between nodes. Indeed, a number of works [69, 26, 10] argue that quality of service on the Internet will could degrade unacceptably that if there is a lack of *accountability*, *i.e.,* mechanisms that empower senders to detect and respond to degraded performance on a data path that violate contractual obligations. Note also that if Internet measurement protocols are used to enforce contracts, nodes may have an economic incentive to bias the information obtained from these protocols.

In this work we provide a rigorous cryptographic examination of *secure* monitoring protocols that are robust even in the presence of malicious nodes on the data path. In particular, we study techniques that allow a sender to *localize* the specific links along the data path where packets were dropped or modified— a task that we call *failure-localization path-quality monitoring.* While some protocols for this task are deployed in the Internet today (*e.g.,* traceroute [4]), they are not robust to nodes that behave adversarially in order to bias measurements.

4.1.1 Our results

We make the following contributions to the study of secure failure-localization path-quality monitoring protocols (in the rest of the paper we call these simply *failure localization* or FL protocols). Throughout the paper, we use the word "packet" to denote data that the sender wishes to transmit, and "message" to refer to both data packets and FL-protocol-related control messages.

Definition. In Section 4.2, we give the first formal definition of security for failure localization protocols. We note that some of the previous FL protocols suggested in the literature, such as [86, 13, 10], do *not* satisfy our definition. (We sketch attacks in Appendix C.1.)

We give two variants of the definition— *per-packet* security requires localizing a link each time a packet is not delivered, while *statistical* security only requires this when a noticeable fraction of packets fail to arrive. An important feature of our definition is that it accounts for the fact that messages can be dropped in the Internet for benign reasons like congestion. We note that care must be taken to design protocols that are simultaneously robust to both adversarial behavior and benign congestion. We discuss the effect of this assumption on some previous work [13] in Appendix C.1.

Protocols. We present three simple protocols satisfying our per-packet (Section 4.3.1) and statistical (Section 4.3.2) security definitions. All of these protocols do not modify the packets sent on the path; instead, they add additional messages. Thus our protocols have the important advantage of allowing backwards compatibility with the current techniques for processing packets in a router, minimizing latency in the router, and not increasing packet size.

Because routers are highly-resource constrained devices that are designed to communicate large amounts of information while storing very little, the most important measure of efficiency for our protocols is storage overhead (*i.e.,* the amount of state each router needs to keep as part of the protocol). We are also concerned with communication overhead (*i.e.,* the number and size of messages added by the protocols), and the computational overhead (*i.e.,* the complexity of the computation that each router needs to perform per packet that it processes).

Our per-packet protocol requires each router to store an $O(n)$-length tag for each packet that it sends, and adds a single $O(n)$-length message to every packet sent (n is the security parameter), and one $O(Kn)$-length messages when a failure occurs. (Typically in the Internet, the path length K is less than 20, when nodes represent individual routers, and when nodes represent Internet Service Providers (ISPs) then there are on average $K \approx 4$, and no more 10 nodes on a typical path [66].) However, the communication and storage overhead of the protocol is considered severe, so we present this mostly for pedagogical purposes and move on to our statistical FL protocols.

Our first statistical protocol is based on sampling, and needs to store and communicate $O(pT)$ tags of length $O(n)$ each when the sampling rate is p and T packets are sent. For clarity and correctness, we present a version of the protocol based on the Symmetric Secure Sampling protocol from Chapter 3 ; this version of the protocol requires each intermediate node to share *symmetric cryptographic keys* with Alice *and* Bob. However, we emphasize that is possible to construct an analogous statistical PQM protocol for the public-key setting as well, based on the Asymmetric Secure Sampling protocols in Chapter 3 . Such a protocol would require each node to perform a similar amount of symmetric cryptographic operations on a per-packet basis, and require only a single public-key cryptographic operation for each T packets sent.

Next, we present much more efficient statistical FL protocol based on the *secure sketching* protocol from Chapter 3 . This protocol requires the each node to share a symmetric key with Alice only, and requires each node to store a single $O(K^2 \log T)$-sized array of counters, called a sketch. The communication overhead of the protocol is only two additional messages of length

$O(K^3 \log T + Kn)$ for every T packets sent, and we do not require any modifications to the packets sent by Alice. However, unlike our sampling-based protocols, this protocol cannot be generalized to public-key setting.

Lower bounds. Like many of the protocols in the literature [11,13,86,109,81,10], both of our protocols require cryptographic keys and computations at each node. These requirements are considered severe in the networking literature; setting up a key infrastructure and agreeing on cryptographic primitives is challenging in the distributed world of the Internet, where each node is owned by a different entity with sometimes incompatible incentives. However, in Section 4.4 we show that these requirements are to some degree *inherent* by:

1. Proving that every secure (per-packet or statistical) FL protocol requires a key infrastructure, or more precisely, that intermediate nodes and Alice and Bob must all share some secret information between each other. This shared secret information can be pairwise symmetric keys, or public-private key pairs.

2. Proving that a one-way function can be constructed from any secure FL protocol.

3. Giving evidence that any practical per-packet secure FL protocol must use these keys in a cryptographic way at *every node* (*e.g.,* , it does not suffice to use the secret information with some simple, non-cryptographic, hash functions as in [33]). We show that in every black-box construction of such a protocol from a random oracle, where at most $O(\log n)$ protocol messages are added per packet, then every intermediate node must query the random oracle. We note that practical protocols designed for Internet routers typically avoid using non-black-box constructions or adding more than a constant number of protocol messages per packet. We also show that for statistically-secure FL, or FL protocols adding $\omega(\log n)$ messages per packet, the necessity of cryptography depends on subtle variations in the security definition.

Implications of our results. Our lower bounds raise questions about the practicality of deploying FL protocols. In small highly-secure networks or for certain classes of traffic, the high key-management and cryptographic overhead required for FL protocols may be tolerable. However, FL protocols may be impractical for widespread deployment in the Internet; firstly because intermediate nodes are owned by competing business entities that may have little incentive to set up a key infrastructure and agree on cryptographic protocols, and secondly because cryptographic computations are expensive in the core of the Internet, where packets must be processed at extremely high speeds (about 2 ns per packet). Thus, our work can be seen as a motivation for finding security functionalities for the Internet that are more practical than failure localization.

4.1.2 Related work

Some of this work (in particular, the results of Section 4.3 and a weaker version of Theorem 4.4.3) appeared in our earlier technical report [48]. We built on [48] in [49], where, together with Jennifer Rexford and Eran Tromer, we gave formal definitions, constructions, and lower bounds for the simpler task of *path-quality monitoring* (PQM). In a PQM protocol the sender only wishes to *detect* if a failure occurred, rather than localize the specific faulty link along the path. We use the results from Chapter 3 in Section 4.3.2 to show how a PQM protocol can be composed to obtain a statistical FL protocol, and in Section 4.4.2 to argue that FL protocols need cryptographic computations.

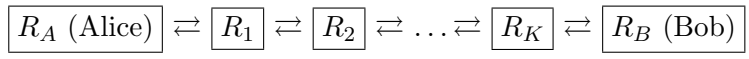


Figure 4.1: A path from Alice to Bob via K intermediate nodes.

In addition to the FL protocols from the networking literature [11,13,86,81,10,107], our work is also related to the work on secure message transmission (SMT) begun by Dolev, Dwork, Waart, and Yung in [32]. In SMT, a sender and receiver are connected by a multiple parallel wires, any of which can be corrupted by an adversary. Here, we consider a single path with a series of nodes that can be corrupted by an adversary, instead of multiple parallel paths. Furthermore, while multiple parallel paths allow SMT protocols to *prevent* failures, in our single path setting, an adversarial intermediate node can always block the communication between sender and receiver. As such, here we only consider techniques for *detecting and localizing* failures.

Subsequent to the publication of this work in [16], Zhang *et al.* [109] considered FL protocols that are similar to our per-packet FL and our sampling-based statistical FL protocols. Furthermore, Amir, Bunn, and Ostrovsky [8] consider FL in the setting of multiple paths, as in the SMT framework.

4.2 Our model

In a failure localization (FL) protocol, a sender Alice wants to know whether the packets she sends to receiver Bob arrive unmodified, and if not, to find the link along the path where the failure occurred (see Figure 4.1). We say a *failure* or *fault* occurs when a data packet that was sent by Alice fails to arrive unmodified at Bob. Following the literature, we make the somewhat strong assumption that Alice knows the identities of all the nodes of the data path. While this assumption only strengthens our lower bounds, it does limit the practicality of our protocols in settings where Alice is not sure about the paths her packets take. For more discussion on this assumption, see Chapter 1. We work in the setting where all traffic travels on symmetric paths (*i.e.,* intermediate nodes have bi-directional communication links with their neighbors, and messages that sender Alice sends to receiver Bob traverse the same path as the messages that Bob sends back to Alice). We say that messages travelling towards Alice are going *upstream*, and messages travelling towards Bob are going *downstream*. An adversary Eve can occupy any set of nodes on the path between Alice and Bob, and can add, drop, or modify messages sent on the links adjacent to any of the nodes she controls. She can also use timing information to attack the protocol.

Localizing links, not nodes. It is well known that an FL protocol can only pinpoint a *link* where a failure occurred, rather than the *node* responsible for the failure. To see why, refer to Figure 4.1, and suppose that (a) Eve controlling node R_2 becomes unresponsive by ignoring all the messages she receives from R_1. Now suppose that (b) Eve controls node R_1 and pretends that R_2 is unresponsive by dropping all communication to and from R_2. Because cases (a) and (b) are completely indistinguishable from Alice's point of view, at best Alice can localize the failure to link $(1, 2)$.

Congestion. Congestion-related packet loss is widespread on the current Internet, caused by protocols like TCP [64] that naturally drive the network into a state of congestion. Our definition accounts for congestion by assuming links can drop each message independently with some probability. One could come up with other models for congestion (*e.g.,* allowing Eve to specify the distribution of congestion-related packet loss), and for some plausible choices our

positive results will still hold. However, we use independent drops for the sake of simplicity. Furthermore, assuming that congestion is not controlled by the adversary only strengthens our lower bounds and makes our model more realistic.

4.2.1 Security definition

Let n be the security parameter. A failure localization protocol consists of an efficient initialization algorithm Init taking n uniformly random bits and generating keys for each node, and efficient node algorithms $\mathsf{Alice}, \mathsf{Bob}, R_1, \ldots, R_K$ which take in a key and communicate with each other as in Figure 4.1. We always fix $K = O(1)$ independent of n. The Alice algorithm takes in a packet that she wants to send to Bob. If communication is successful, then the Bob algorithm outputs the packet that Alice sent. Our security definitions are game-based:

Definition 4.2.1 (Security game for FL)**.** *The game begins when Eve chooses a subset of nodes $E \subseteq \{1, \ldots, K\}$ that she will occupy for the duration of the game. The Init algorithm is then used to generate keys for each node, and Eve is given the keys for the nodes $i \in E$ that she controls. We define an oracle Source that generates data packets d for the Alice algorithm to send. We allow Eve to choose the packets that the Source oracle generates, subject to the condition that she may not choose the same packet more than once during the game.*[1]

We allow Eve to add, drop, or modify any of the messages sent on the links adjacent to the nodes she occupies. We include congestion in our model by requiring that, for each message sent on each link on the path, the link goes down *or drops the message with some constant probability $\rho > 0$. Notice that this means that a failure can happen at links not adjacent to a node occupied by Eve.*

We introduce the notion of time into our model by assuming that the game proceeds in discrete time steps; in each time step, a node can take in an input and produce an output, and each link can transmit a single message. (Thus, each time step represents an event occurring on the network.) Because it is expensive to have securely synchronized clocks in a distributed system like the Internet,[2] *we do* not *allow the honest algorithms to take timing information as an input. However, to model timing attacks, we assume that Eve knows which time step that the game is in.*

Then, our per-packet security definition uses the game defined in Definition 4.2.1:

Definition 4.2.2 (Per-packet security for FL)**.** *In the per-packet security game, Eve gets to interact with the Source oracle and the "honest" node algorithms as in Definition 4.2.1, until she decides to stop. For each packet sent, Alice must output either $\surd$ (*i.e., *not raise an alarm) or a link ℓ (*i.e., *raise an alarm and localize a failure to ℓ). We assume that the game is* sequential*: Alice must output a decision for each data packet before starting to transmit the next data packet (see remarks below). We say that an FL protocol is* per-packet secure *if the following hold:*

1. *(*Secure localization*). For every packet d sent by the Source oracle that is not successfully output by Bob, then Alice outputs a link ℓ such that either (a) link ℓ is adjacent to a node occupied by Eve, or (b) link ℓ went down due to congestion for one of the messages (including FL protocol messages) associated with sending packet d from Alice to Bob.*

[1] We make this assumption because there is natural entropy in packet contents, due to TCP sequence numbers and IP ID fields [33]. One way to enforce this assumption in practice, is to require the use of ephemeral 'interval keys' which are refreshed at the end an interval, and to assume that the natural entropy in packet contents enforces the uniqueness of packets over an interval (see the further discussion in Chapter 3).

[2] Indeed, the NTP protocol used for clock synchronization on the Internet is not secure [79], and thus should not be used as an input to a secure FL protocol.

2. *(*No false positives*). For every packet d sent by the* Source *oracle that is successfully output by Bob, for which there was no congestion, and for which Eve does not deviate from the protocol, Alice outputs* $\surd$.

We need to introduce a few new concepts for our statistical security definition. First, we define an *interval* as a sequence of T packets (and associated FL protocol messages) that Alice sends to Bob.[3] Next, we use the following parameters: a false alarm threshold α, a detection threshold for the path β (where $0 < \alpha < \beta < 1$) and an error parameter $\delta \in \{0, 1\}$. Usually, we will set α such that congestion alone almost never causes the failure rate on a path to exceed the false alarm threshold.

Definition 4.2.3 ((α, β, δ)-Statistical security for FL)**.** *In the statistical security game, Eve is allowed to choose the number of* intervals *for which she wants to interact with the* Source *oracle and the honest nodes as in Definition 4.2.1. The number of packets per interval T may grow with n, but is always at least some minimum number depending α, β, δ, K. At the end of each* interval*, Alice needs to output either* $\surd$ *(*i.e., *not raise an alarm) or a link* ℓ *(*i.e., *raise an alarm and localize a link). The game is sequential; Alice must output a decision for each interval before starting the next interval. Then, an FL protocol is* statistically secure *if the following hold:*

1. *(*Secure localization*). For any interval in the security game where Eve causes the failure rate on the path to exceed the detection threshold β, then with probability $1 - \delta$ Alice raises alarm for a link ℓ that is adjacent to Eve, or a link ℓ whose failure rate exceeds $\frac{\alpha}{K+1}$.*

2. *(*Few false positives*). For any interval in the security game where Eve does not deviate from the correct algorithm R_i of any of the nodes $i \in E$ that she controls and the failure rate on each link is below the (per-link) false alarm threshold $\frac{\alpha}{K+1}$, then the probability that Alice outputs* $\surd$ *is at least* $1 - \delta$.

We now discuss some properties of our security definition.

Benign and malicious failures. Our security definitions require Alice to accurately localize failures, but these failures may be caused by Eve, or may be the result of *benign causes*, such as congestion. We do not require Alice to distinguish between benign or malicious (*i.e.,* due to Eve) failures, because Eve can always drop packets in a way that "looks like" congestion.

Sequential games. For simplicity, in our per-packet security game we required Alice to make FL decisions before she sends a new data packet. This is to capture the fact that such protocols should provide "real-time" information about the quality of the paths she uses, and so we did not allow Alice in the per-packet case to make decisions only after sending many packets (as is done in the statistical security case). We note that while our lower bounds (*i.e.,* attacks) are sequential, our positive results (*i.e.,* , protocols) do not use the assumption of sequential execution in any way, and are secure in a more general setting where Eve can choose, at each point in time, which of the previously sent packets "time-out", and then Alice needs to output FL decisions for these packets. We emphasize that the sequential assumption does *not* prevent Alice from keeping state and using information from *past* packets in order to make FL decisions. (Though none of our positive results require that Alice does this.)

Movements of the adversary. Our model does not allow Eve to move from node to node in a single security game. This assumption, which only strengthens our lower bounds, does not significantly limit the practicality of our protocols for a number of reasons. Firstly, when Eve models a Internet service provider that tries to bias the results of FL protocol for business

[3] We can think of an interval as all the packets sent in some time period (*e.g.,* approximately 10^7 packets are sent 100 msec over a 5 Gbps Internet path).

reasons, it is reasonable to assume that she may only occupy nodes owned by her business entity. Furthermore, when Eve is an external attacker or virus that compromises a router, "leaving" a router means that the legitimate owner of the router removed the attacker from the router, *e.g.,* by refreshing its keys. We model this key refresh process as a re-start of the security game. Furthermore, in practice "movements" to a new router happen infrequently, since an external attacker typically needs a different strategy each time it compromises a router owned by a different business entity.

Generalizations. All our results generalize to the setting where congestion rates, false alarm thresholds, and detection thresholds are different per link; we set them all equal here for simplicity. Our lower bounds also hold for the weaker adversary model where Eve can occupy only one node and the Source oracle generates independent (efficiently-samplable) packets from a distribution that is *not* controlled by Eve.

4.3 Protocols

We now present protocols for secure per-packet and statistical FL. Our protocols are related, though not identical to those of [10, 11, 13]. (In Appendix C.1 we show that the protocols in [10,13] do not satisfy our security definitions.)

We use the notation $[m]_k$ to denote a message m authenticated by a key k using a *message authentication code* (MAC); such schemes can be constructed from any one-way function [51,54]. We'll often use the well-known notion of an *onion report*: if every node R_i wants to transmit a report τ_i to Alice in an authenticated way, then we define inductively $\theta_{K+1} = [(K+1, \tau_{\mathsf{Bob}})]_{k_{\mathsf{Bob}}}$ and for $1 \leq i \leq K$, $\theta_i = [(i, \tau_i, \theta_{i+1})]_{k_i}$. That is, each R_i's report is appended with its downstream neighbors' reports before being authenticated and passed upstream. Onion reports prevent Eve from selectively dropping reports — if Eve occupies R_j and wants to drop the report τ_j of R_i for some $i > j$ then, under the assumption that Eve cannot forge MACs, Alice will discover that R_j tampered with the onion report. We also note that every time we send or store a packet d in acknowledgments and reports, we could save space by replacing d with an $O(n)$-length hash of d via some collision-resistant hash function, where n is the security parameter.

4.3.1 Optimistic Per-Packet FL Protocol

We assume that each node R_i shares a symmetric key k_i with Alice. For each packet that Alice sends, the protocol proceeds in two phases:

The detect phase. Alice stores each packet d that she sends to Bob. When Bob receives the packet d, he responds with an ack of the form $a = [d]_{k_B}$. Alice removes the packet d from storage when she receives a validly MAC'd corresponding ack, and raises an alarm if a valid ack is not received.[4] We also require each intermediate node to store each data packet and corresponding ack.

The localize phase. This phase is run only if Alice raises an alarm for a packet d. Alice sends an *onion report request* $q = (\mathsf{report}, d)$ downstream towards Bob. To respond to the request, each node R_i checks if he stored data packet d; if he did, R_i sets $\tau_i = (q, i, d, a)$ where a is the ack he saw corresponding to packet d, and substituting the symbol $\perp$ for d and/or a if he failed to receive that packet or an ack. R_i then creates an onion report θ_i using τ_i as described above.

[4] In practice, each packet d should be stored along with a local timeout at Alice. If the ack does not arrive before the timeout expires, then Alice should raise an alarm.

In the onion report, R_i can substitute the symbol $\theta_{i+1} = \perp$ if he fails to receive a θ_{i+1} from R_{i+1}. [5]

To localize the failure, Alice classifies the onion reports that she received in response to her onion report request q. An onion report $\theta_i = [q', i', d', a', \theta_{i+1}]_{k_i}$ is "consistent" if it is present, *i.e.,* $\theta_i \neq \perp$, and all of the following four conditions hold. Otherwise, an onion report is "inconsistent".

1. $q' = q$ sent out by Alice.
2. The MAC on θ_i is valid.
3. $d' = d$, where d is the packet queried in q.
4. a' is *not* a valid ack for packet d.

Alice localizes then localizes the upstream-most link $(i, i+1)$ where the onion reports transition from consistent to inconsistent.

Theorem 4.3.1. *The optimistic FL protocol is per-packet secure.*

Proof. Eve can win the security game by causing a failure and either (a) convincing Alice that no failure occurred, or (b) causes Alice to localize a node that is not adjacent to Eve. We show that both (a) and (b) happen with negligible probability:

Consider (a) first. Recall that the packets d Alice sends in the security game are unique, and that each ack for a packet d contains the packet d. It follows that if Eve creates a valid ack to a packet d that was dropped before it arrived at Bob, she needs to forge the MAC on a message (B, d). From the security of the MAC, she can do this with only negligible probability.

Next, consider (b). Let R_i be the upstream-most node where Eve either caused a failure or tampered with an ack. We have two cases:

- Suppose all the nodes upstream of Eve's node R_i do not deviate from the correct algorithm. Let R_j be the first honest node that is downstream of node R_i (we know such a node exists because Eve cannot occupy Bob's node). Since R_{i-1} and R_j are honest, they correctly generate their onion reports θ_{i-1}, θ_j, and these reports must have different entries in their "data" fields (if Eve tampered with the packet at node R_i), and/or different entries in their "ack" fields (if Eve tampered with the ack at R_i). Now, since all the nodes upstream of R_{i-1} behave honestly, their onion reports are all be consistent. Also, conditioned on Eve not forging R_j's MAC on the onion report, we know that R_j's onion report is inconsistent. It follows that the upstream most transition from consistent to inconsistent reports must occur on some link between R_{i-1} and R_j and Alice will output a link adjacent to Eve.

- Suppose one of the nodes upstream of Eve's node R_i does deviate from the correct algorithm. Call the upstream-most such node R_e, and observe that it must be occupied by Eve. By the way we chose R_i, we know that R_e did not cause a failure or tamper with an ack. It follows that R_e must have tampered with an onion report request or an onion report. Let R_j be the first honest node downstream of R_e. Conditioned on not forging the MAC of an honest node in the onion report, it follows that Eve at R_e must have caused a consistent/inconsistent transition at some link between R_{e-1} and R_j, and so that Alice will output a link adjacent to Eve.

[5] When each node originally receives the onion report request q from Alice, each node sets an local time-out that determines how long he should wait for his downstream neighbor to send their onion report. If the onion time-out expires, the node reports a missing onion report by setting $\theta_{i+1} = \perp$ and then proceeding to construct his own onion report as before.

Combining these two cases, we see that from the security of the MAC, (b) happens with negligible probability. □

Efficiency. We remark that the detect phase of this protocol incurs a high storage and communication overhead at each node on the path; we require the addition of at least one new $O(n)$-length message for each data packet sent, and even more egregiously, each node must store (an $O(n)$-length digest of) each packet it sends until it receives an ack or onion report request. This high overhead makes this protocol highly impractical for regular Internet traffic; however, it might be useful for specialized highly-secure networks, or for certain classes of low-volume traffic *e.g.,* network management traffic.

4.3.2 A Composition Technique for Statistical FL

We now consider statistical security protocols, that apply results from our previous work on statistical PQM Chapter 3 to obtain statistical FL protocols with much lower overhead. In a statistical PQM protocol, Alice *detects* whenever the average failure rate exceeds a threshold β (but she need not localize a link).

Here we show how to compose the lightweight PQM protocols we presented in Chapter 3 to obtain statistical FL protocols. While it is possible to give a very general composition theorem, for clarity and concreteness, we first describe how to compose the simpler *symmetric secure sampling (SSS)* protocol of Chapter 3 to obtain a protocol with storage and communication overhead that linear in (*i.e.,* a small fraction of) the number of sent packets in the interval, T. The protocol we present here requires each node to share *pairwise* keys with Alice and Bob. However, we can extend this result to the public-key setting by composing instances of the asymmetric secure sampling protocols of Chapter 3 , to obtain a protocol that requires only a single public-key cryptographic operation per interval of T sent packets. For brevity, we omit any further discussion of this protocol here.

Finally, we show how to compose the *secure sketch protocol* of Chapter 3 to obtain a more efficient FL protocol with about $O(K^2 \log T + n)$ storage overhead at each node and only two additional control messages.

A composition with that uses Secure Sampling PQM.

Symmetric Secure Sampling (SSS), a statistical PQM protocol from Chapter 3 . SSS requires Alice and Bob to securely designate a random p fraction of the data packets that Alice sends to Bob as "probes", and require that Bob send MAC'd acknowledgments for all the probes. We call p the *probe frequency.* To do this, Alice and Bob share a secret $k = (k_1, k_2)$. For each packet d that Alice sends to Bob, they use k_1 to compute a function Probe that determines whether or not a packet d is a probe and should therefore be stored, and acknowledged. To acknowledge a probe, Bob sends Alice an ack $[d]_{k_2}$ that is MAC'd using k_2. The Probe function is implemented using a pseudorandom function (PRF) f keyed with k_1, that we think of as mapping strings to integers in $[0, 2^{n-1}]$; We define

$$\begin{array}{ll} \mathsf{Probe}_{k_1}(d) = \text{Yes} & \text{if } \frac{f_{k_1}(d)}{2^n} < p, \\ \mathsf{Probe}_{k_1}(d) = \text{No} & \text{otherwise.} \end{array} \tag{4.1}$$

For each interval, Alice stores each probe packet (*i.e.,* each packet d such that $\mathsf{Probe}_{k_1}(d)$ =Yes). At the end of the interval, after T packets are sent, Alice computes V, a count of the number of stored (probe) packets for which she failed to receive a valid ack. She computes the average failure rate as $\frac{V}{pT}$.

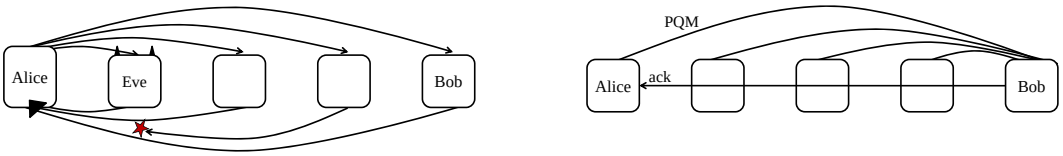


Figure 4.2: On the left an insecure composition, on the right our secure composition.

A composition that does not work. Perhaps the most natural approach to construct a statistical FL protocol is to have Alice run K simultaneous PQM protocols with each of the intermediate nodes, and use the statistics from each protocol to infer behavior at each link (similar to [86, 13, 107]). However, we now show that this composition is vulnerable to the following *timing attack*: Suppose a packet d that Alice sends to Bob is ack'd by innocent node R_j with message a. Then, if Eve occupies node R_i for $i < j - 1$, she can determine that R_j originated the ack a by counting the time steps that elapsed between the time step in which she saw d and time step in which she saw a. Then, Eve can implicate R_j by selectively dropping every ack that originates at R_j. Notice that this attack results from the structure of this composition, and cannot be prevented even when acks are encrypted.[6] In practice, this attack can be launched when isolated burst of packets triggers a separate burst of acks at each intermediate node.

Composing PQM to statistical FL. We require that every node R_i shares pairwise keys k_i^A, k_i^B with Alice and Bob respectively. Using k_i^B, each intermediate node runs a statistical PQM protocol with Bob with the following modification: whenever Bob decides to send an ack for a packet d to an intermediate node R_i, Bob will (1) always address the ack to Alice and (2) MAC the ack in onion fashion, starting with k^B_{Alice} (on the inside of the onion) and ending with k_K^B (on the outside of the onion). Each node forwards all acks upstream, and processes only the ack he expects. At the end of the interval u, Alice will send an onion report request $q = (\mathsf{report}, u)$ to all the intermediate nodes. Each intermediate node produces a MAC'd onion report $\theta_i = [q, i, V_i, \theta_{i+1}]_{k_i^A}$ where V_i is his estimate of the average failure rate on the path between himself and Bob. Letting α, β be the false alarm and detection thresholds, when Alice receives the final onion report θ_1, she computes $F_\ell = V_i - V_{i+1}$ for each link $\ell = (i, i+1)$, and outputs ℓ if $F_\ell > \frac{\alpha+\beta}{2(K+1)}$, or if $\ell = (i, i+1)$ is the upstream-most link when the onion report θ_{i+1} refers to the wrong interval, is missing, or is invalidly MAC'd.
We prove that this scheme is secure provided that the interval length T is long enough and the congestion rate ρ is small enough.

Theorem 4.3.2. *The composition of SSS described above with probe frequency p satisfies* (α, β, δ)*-strong statistical security when each interval contains at least* $T = O(\frac{K^2}{p(\beta-\alpha)^2} \ln \frac{K}{\delta})$ *packets and the congestion rate satisfies* $\beta - \alpha \gg K\rho$.

Proof. First, observe that the probability that any efficient adversary Eve successfully forges an ack for a dropped packet by forging a MAC used in SSS is negligible. As in the Optimistic Protocol, the probability that any efficient adversary Eve successfully forges the onion report of an honest node (by forging the MAC on the onion report) is negligible as well. Hence, for the rest of this proof assume that Eve does not forge an ack to a dropped packet or validly forge the onion report of an honest node. Moreover, we can assume that Eve does not tamper with the onion report, or else she will implicate a link adjacent to one of the nodes she controls. We now work within a single interval:

- Let V_i be R_i's *estimate* of the failure rate between R_i and Bob.

[6] In [107], the authors suggest randomizing the sending time of acks.

- Let D_i be a count of the number of packets that were dropped or modified on the path between R_i and Bob.
- Let C_i be the number of acks intended for *any node* that were dropped or modified on the path between Bob and R_i.
- Let $p' = \frac{p}{1-(1-p)^{K+1}}$ be the probability that a node R_i expects an ack to a packet d (*i.e.,* $\mathsf{Probe}_{k_i^B}(d) = \text{Yes}$) conditioned on there being at least one node expecting an ack to packet d (*i.e.,* $\exists j \in \{0, \ldots, K\},\ \mathsf{Probe}_{k_j^B}(d) = \text{Yes}$).[7]

Note that when R_i estimates the average failure rate on the path from R_i to Bob, she is unable to distinguish between dropped packets and dropped acks. Also, it is possible that $D_i > D_{i+1}$ or $C_i > C_{i+1}$ for two adjacent uncorrupted nodes because of congestion. In the absence of adversarial behavior at R_i, the expectation of the estimator V_i that Alice receives in the onion report is $\frac{1}{T}(D_i + \frac{p'}{p}C_i)$. Finally, notice that the average failure rate on link $(i, i+1)$ is $\frac{1}{T}(D_i - D_{i+1})$.

Set $\gamma = \frac{\beta-\alpha}{2(K+1)}$. If $T = O(\frac{K^2}{p(\beta-\alpha)^2} \ln \frac{K}{\delta})$ then we have the following lemmata:

Lemma 4.3.3 (Deviation of the estimator V_i)**.** *For each $i \notin E$ where E is the set of nodes corrupted by Eve it holds (up to negligible error) that*

$$\Pr\left[\left|V_i - \tfrac{1}{T}(D_i + \tfrac{p'}{p}C_i)\right| > \tfrac{1}{4}\gamma\right] < \tfrac{\delta}{4(K+1)}$$

Lemma 4.3.4 (Acks dropped due to congestion)**.** *For each $i, i+1 \notin E$, it holds (up to negligible error) that*

$$\Pr\left[\tfrac{p'}{p}\tfrac{C_i - C_{i+1}}{T} > \tfrac{\gamma}{2}\right] < \tfrac{\delta}{2(K+1)}$$

The proofs of these lemmata are technical, but not difficult. We defer them to Appendix C.2.1. Both proofs are applications of the Chernoff bound under the assumption that the Probe function is implemented with a truly random function; the negligible error refers the difference between a PRF and a truly random function. The proof of Lemma 4.3.3 relies on the fact that Eve cannot bias node R_i's estimate of C_i by selectively dropping acks because (1) acks destined for different nodes look identical, and they all originate at Bob (so that an adversary cannot use timing to distinguish between them), and (2) acks are onion MAC'd, so the adversary cannot selectively tamper with an ack intended for an upstream node. The proof of Lemma 4.3.4 also relies on the fact that $\beta - \alpha \gg K\rho$.

Few false positives: To prove this, we consider an interval where all the nodes on the path behave honestly, and show that, with probability at least $1 - \delta$, Alice will not raise an alarm during this "honest interval".

Consider link $\ell = (i, i+1)$ where the average failure rate is less than the false alarm threshold so $\frac{1}{T}(D_i - D_{i+1}) < \frac{\alpha}{K+1}$. We now show that Alice will not raise an alarm for this link ℓ by proving that Alice's estimate of the failure rate for ℓ, *i.e.,* $V_i - V_{i+1}$, does not exceed her alarm decision threshold, *i.e.,* $\frac{\alpha+\beta}{2(K+1)}$. We do this by proving that

$$\Pr\left[\left|(V_i - V_{i+1}) - \tfrac{1}{T}(D_i - D_{i+1})\right| > \tfrac{\alpha+\beta}{2(K+1)} - \tfrac{\alpha}{K+1} = \gamma\right] < \tfrac{\delta}{K+1} \tag{4.2}$$

Notice that "Few false positives" condition follows from (4.2) by a union bound over all $K+1$ links.

[7] This quantity is the probability that a node R_i samples an ack that was dropped between R_i and R_B, since at least one node must have sampled the corresponding packet in order for the ack to be transmitted at all.

To prove (4.2), we start with the expression below, and apply the triangle inequality, and then Lemma 4.3.3:

$$\begin{aligned}
&\Pr[|(V_i - V_{i+1}) - (\tfrac{D_i - D_{i+1}}{T} + \tfrac{p'}{p}\tfrac{C_i - C_{i+1}}{T})| > \gamma/2] \\
&\quad \leq \Pr[|V_i - \tfrac{1}{T}(D_i + \tfrac{p'}{p}C_i)| > \gamma/4] + \Pr[|V_{i+1} - \tfrac{1}{T}(D_{i+1} + \tfrac{p'}{p}C_{i+1})| > \gamma/4] \\
&\quad \leq \tfrac{\delta}{2(K+1)} \qquad (4.3)
\end{aligned}$$

Next, from Lemma 4.3.4 we know that $\Pr[\frac{p'}{p}\frac{C_i - C_{i+1}}{T} > \gamma/2] \leq \frac{\delta}{2(K+1)}$, and so a union bound over this expression and (4.3) proves (4.2).

Secure localization: We now show that if Eve drops more than a β fraction of packets in any interval, then Alice will catch her with probability at least $1 - \delta$. Since the actual failure rate on the path is $\frac{1}{T}D_A > \beta$, we start by applying Lemma 4.3.3 to find that Alice's estimate of the failure rate is $V_A > \beta - \frac{\gamma}{4}$ with probability at least $1 - \frac{\delta}{4(K+1)}$. We now use an averaging argument to claim that there exists some link $\ell = (i, i+1)$ such that $V_i - V_{i+1} > \frac{\alpha+\beta}{2(K+1)}$. To see why, suppose for the sake of contradiction that for all i we had $V_i - V_{i+1} \leq \frac{\alpha+\beta}{2(K+1)}$. Then, it follows that

$$V_A = \sum_{i=0}^{K}(V_i - V_{i+1}) \leq \sum_{\ell} \tfrac{\alpha+\beta}{2(K+1)} = \tfrac{\alpha+\beta}{2} < \beta - \tfrac{\gamma}{4}$$

where $V_{K+1} = 0$ (Bob's estimate of drops to himself is 0). But this contradicts our condition that $V_A > \beta - \frac{\gamma}{4}$, so there is at least one link $\ell = (i, i+1)$ with $V_i - V_{i+1} > \frac{\alpha+\beta}{2(K+1)}$ so that Alice raises an alarm.

Next, recall that we assume that for any link where the true failure rate due to congestion less than $\frac{\alpha}{K+1}$, we have from our proof of the "Few false positives" condition that with probability $\frac{\delta}{K+1}$, Alice does not raise an alarm for link ℓ between two honest nodes. Then, Alice must have raised the alarm for a link adjacent to Eve with probability at least $1 - \delta$ (by a union bound) or a link with actual failure rate larger than $\frac{\alpha}{K+1}$, and secure localization follows. □

Efficiency. We remark that this protocol requires Alice and each intermediate node to store tags of length $O(n)$ for a p-fraction of the packets that they send. The communication overhead of the protocols is similarly a p-fraction of $O(n)$-length tags. Notice that, under the assumption that the interval T is long enough, we can take p to be arbitrarily small.

A composition that uses Secure Sketch PQM

Secure Sketch, a statistical PQM protocol from Chapter 3 . In Secure Sketch PQM, Alice and Bob to securely aggregate information about all the traffic that Alice sends to Bob in short hash-based data-structure called a sketch. At the end of the interval, Alice and Bob exchange their sketches using in MAC'd control messages, and use the sketches to estimate the failure rate on the path. To do this, Alice and Bob share a secret $k = (k_1, k_2)$. The key k_1 is used to key a PRF that is used to at the beginning of interval u by both parties to derive the interval key k_u as $k_u = f_{k_1}(u)$. For each data packet d, Alice and Bob use the interval key k_u to compute a hash $f_{k_1}(d)$ of the packet. The output of the hash function is a length N-vector that added to a vector of N counters, each of length b, called the sketch. After T packets are sent and the interval ends, Alice sends Bob control message that contains her sketch and the next interval number, and is MAC'd with k_2. Bob responds by subtracting Alice's sketch from his own, and replying with a MAC'd control message containing the interval number and difference

between the two sketches. Alice then obtains an estimate of the failure rate V by computing some function g on the difference between the two sketches.

The security of secure sketch PQM. Briefly, the secure sketch protocol works because it correctly estimates the p^{th}-moment of a packet stream (for some $p \geq 1$). That is, consider the stream of T packets that Alice sends to Bob during the interval, where each packet is chosen from a universe U (*e.g.,* if packets are 1500bytes, then $|U| \approx 2^{1500 \cdot 8}$). Let $\mathbf{v}_A$ be the characteristic vector of this stream, a U-dimensional vector that has c in the position corresponding to packet x if packet x was sent c times during the interval. Similarly, let $\mathbf{v}_B$ be the characteristic vector of the stream of packets received by Bob. Then, the sketches allow Alice to estimate $\|\mathbf{v}_A - \mathbf{v}_B\|_p$. In particular, we say that a sketching protocol (ε, δ)-estimates the p^{th}-moment of the characteristic vector $\mathbf{v}_A - \mathbf{v}_B$ if

$$\Pr\left[\left|V - \|\mathbf{v}_A - \mathbf{v}_B\|_p\right| \leq \varepsilon \|\mathbf{v}_A - \mathbf{v}_B\|_p\right] < 1 - \delta \tag{4.4}$$

where the probability is taken over the randomly chosen key k_u used to key the packet-hash function f. In Chapter 3 , we discuss exactly how to choose the hash function f, and how this choice affects p, the norm estimated by the sketch, and $N \times b$, the size of the sketch. For our purposes we shall simply note that if the hash function f is an appropriately-chosen PRF, then we can use sketch of size $N \times b$ where $N = O(\frac{1}{\varepsilon^2}\log(\frac{1}{\delta}))$ and $b = O(\log(T))$.

We no longer have timing attacks. In secure sketch PQM, Alice and Bob exchange only a pair of control messages at the end of the interval; no other communication between them is required. Because the *timing* of these control messages do *not* leak any information, the timing attack we mentioned in Section 4.3.2 is no longer an issue. Our composition of secure sketch PQM to statistical FL will have Alice run K simultaneous PQM protocols with each of the intermediate nodes as in Figure 4.2, and use the statistics from each protocol to infer behavior at each link.

A simpler composition. We require that every node R_i shares pairwise keys k_i with Alice only (*c.f.,* with our SSS-based composition, where nodes need to share keys with Bob as well). Using k_i, each intermediate node runs a secure sketch PQM protocol with Alice, so that Alice will keep a sketch $\mathbf{w}_i^A$ for every $i \in [K]$ and every other node R_i will keep a single sketch $\mathbf{w}_i$. However, instead of sending individual control messages to each node at the end of interval u, Alice will now send a single onion-MAC'd interval-end message containing all her sketches as

$$q = [(u, \mathbf{w}_1^A)[(u, \mathbf{w}_2^A)...[(u, \mathbf{w}_B^A)]_{k_B}...]_{k_2}]_{k_1}$$

to all the intermediate nodes. Upon receiving a validly-MAC'd interval-end message, intermediate node R_i extracts the sketch w_i^A, and passes the interval-end message to R_{i+1}. (R_i drops the interval-end message if the MAC is invalid.) Finally, as in the usual composition, each node R_i produces a MAC'd onion report $\theta_i = [u, i, V_i, \theta_{i+1}]_{k_i}$. Here, V_i is node R_i's estimate of $\|\mathbf{v}_i - \mathbf{v}_A\|_p$, which is computed by applying the function g to the difference sketch $\mathbf{w}_i - \mathbf{w}_i^A$. (Recall that v_A is the characteristic vector of the stream of packets that Alice sends, and $\mathbf{v}_i$ is the characteristic vector of the stream of packets that R_i receives.) Letting α, β be the false alarm and detection thresholds, when Alice receives the final onion report θ_1, computes $F_\ell = V_i - V_{i+1}$ for each link $\ell = (i, i+1)$, and outputs ℓ if $\ell = (i, i+1)$ is the upstream-most link where $F_\ell > \frac{T}{K+1}\frac{\beta(2\alpha+\beta)}{\alpha+2\beta} = \Gamma$, or the onion report θ_{i+1} refers to the wrong interval, is missing, or is invalidly MAC'd. If there is no such link, she outputs $\surd$.

Limiting the number of nodes occupied by Eve. To prove that this scheme is secure, we need to assume that interval length T is long enough, the sketches are big enough, and the congestion rate ρ is small enough. Our proof also relies on limiting the number of links occupied by Eve to $\approx \sqrt{K}$. However, we conjecture that it may be possible to weaken this assumption,

as we have not been able to find any attacks on the protocol when Eve occupies more than $\sqrt{K}$ links. For more discussion, see the remarks in Appendix C.2.4.

Theorem 4.3.5. *The composition of secure sketch PQM described above satisfies (α, β, δ)-statistical security if the congestion rate satisfies $\rho K^2 \leq \beta$, Eve occupies $M \leq \sqrt{(K+1)(1-\frac{\rho}{\beta}K^2)}$ links, each interval contains at least $T > \frac{K+1}{\alpha}$ packets, and for each $i \in [K]$, sketches $\mathbf{w}_i, \mathbf{w}_i^A$ have size*

$$N_i \times b = O\left(i^2 \left(\tfrac{2\beta+\alpha}{\beta-\alpha}\right)^2 \log(\tfrac{K}{\delta})\right) \times O(\log T) \tag{4.5}$$

Proof. First, the probability that any efficient adversary Eve successfully forges the interval end message or the onion report of an honest node (by forging the MAC) is negligible. Hence, for the rest of this proof assume that Eve does not validly forge the onion report of an honest node. Moreover, we can assume that Eve does not tamper with the interval-end message of the onion report, or else she will implicate a link adjacent to one of the nodes she controls. We now work within a single interval, and use the following definitions:

- Let $\mathbf{v}_A$ is the characteristic vector of the stream of packets that Alice sends and $\mathbf{v}_i$ for $i \in [K+1]$ to be the characteristic vector of the stream of data packets that R_i receives.
- Let $\mathbf{x}_i = \mathbf{v}_i - \mathbf{v}_A$. We can decompose any $\mathbf{x}_i$ into two vectors $\mathbf{x}_i = \mathbf{d}_i + \mathbf{a}_i$. The vector $\mathbf{d}_i$ is the characteristic vector of packets *dropped* on the path from Alice to R_i, and contains the non-negative components of $\mathbf{x}_i$. The vector $\mathbf{a}$ is the characteristic vector of packets *added* on the path from Alice to R_i, and contains the non-positive components of $\mathbf{x}_i$. Also notice that the non-zero coordinates of $\mathbf{d}$ and $\mathbf{a}$ are disjoint.
- Let V_i be R_i's *estimate* of $\|\mathbf{x}_i\|_p^p$.
- Let D_i be a count of the number of failures that occurred on the path between Alice and R_i.

Our proof also makes use of the following identity

$$\|\mathbf{x}_i\|_p^p = \|\mathbf{d}_i\|_p^p + \|\mathbf{a}_i\|_p^p = D_i + \|\mathbf{a}_i\|_p^p \tag{4.6}$$

The first equality follows because the non-zero coordinates of $\mathbf{d}$ and $\mathbf{a}$ are disjoint. The second equality follows because every packet that Alice send is unique so that that $\mathbf{d}$ is a $\{0,1\}$-vector for every $i \in [K+1]$. In Chapter 3 we show that if interval key is refreshed at the end of each interval, then if each sketch has $N_i \times b = O(\frac{1}{\varepsilon_i}\log\frac{1}{\delta'})$ then it follows that each estimate V_i (ε_i, δ')-approximates $\|\mathbf{x}_i\|_p^p$ as per (4.4). Also, we will require that $\frac{\alpha}{K+1}T > 1$ (which gives us the bound on T, the number of packets in the interval), and prove the following lemma in Appendix C.2.3:

Lemma 4.3.6. *Let $\Gamma = \frac{T}{K+1}\frac{\beta(2\alpha+\beta)}{\alpha+2\beta}$ and $\varepsilon_i = \frac{1}{2i}\frac{\beta-\alpha}{2\beta+\alpha}$. For every $i \in [K]$, assume that R_i computes an estimate V_i that (ε_i, δ')-estimates $\|\mathbf{x}_i\|_p^p$. Suppose also that $\|\mathbf{x}_i\|_p^p \leq \frac{\beta i}{K+1}$. Then with probability at least $1-2\delta'$ it follows that:*

1. *If "link $(i, i+1)$ is good" so that $\|\mathbf{x}_{i+1}\|_p^p - \|\mathbf{x}_i\|_p^p \leq \frac{\alpha}{K+1}T$ then $V_{i+1} - V_i \leq \Gamma$.*
2. *If "link $(i, i+1)$ is bad" so that $\|\mathbf{x}_{i+1}\|_p^p - \|\mathbf{x}_i\|_p^p \geq \frac{\beta}{K+1}T$ then $V_{i+1} - V_i \geq \Gamma$.*

We use Lemma 4.3.6 to prove the "few false positives" and "secure localization" conditions.

Few false positives: To prove this, we consider an interval where all the nodes on the path behave honestly. During this interval, we know that no packets were added anywhere on the path (so that $\|\mathbf{a}_i\|_p^p = 0$ for each $i \in [K+1]$) and less than $\frac{\alpha}{K+1}$ packets were dropped at each link. We can apply identity (4.6) to find that for each link $(i, i+1)$ we have

$$\|\mathbf{x}_{i+1}\|_p^p - \|\mathbf{x}_i\|_p^p = D_{i+1} - D_i + 0 + 0 \leq \tfrac{\alpha}{K+1} \tag{4.7}$$

and the telescoping nature of (4.7) gives us that

$$\|\mathbf{x}_i\|_p^p = (\|\mathbf{x}_i\|_p^p - \|\mathbf{x}_{i-1}\|_p^p) + ... + (\|\mathbf{x}_2\|_p^p - \|\mathbf{x}_1\|_p^p) + \|\mathbf{x}_1\|_p^p \leq \tfrac{\alpha i}{K+1} \tag{4.8}$$

We can now apply Lemma 4.3.6 to show that, with probability at least $1 - 2\delta'$ we have that $V_{i+1} - V_i \leq \Gamma$ so that Alice will not output link $(i, i+1)$. A union bound over the $K+1$ links gives us that Alice will output $\surd$ during this interval with probability at least $1 - 2(K+1)\delta_i$.

Secure localization: We now show that if Eve causes more than a β fraction of failures in the interval, then with probability at least $1 - \delta$, Alice will either catch Eve or output a link with more than $\frac{\alpha}{K+1}$ failures. Recall that Alice outputs the upstream-most link $\ell = (i, i+1)$ for which there is an "alarm", *i.e.,* where $V_{i+1} - V_i \geq \Gamma$. We need the following simple observation:

Lemma 4.3.7. *Define event E_i as the event that $\|\mathbf{x}_i\|_p^p \leq \frac{\beta i}{K+1}$ .For each $i \in [K+1]$, if Alice does not raise an alarm for any link upstream of link i, then E_i holds with probability $1 - 2i\delta'$.*

Proof. Suppose that Alice does *not* raise an alarm for all the links upstream of node R_i. It follows from Lemma 4.3.6 that $\|\mathbf{x}_{j+1}\|_p^p - \|\mathbf{x}_j\|_p^p \leq \frac{\beta}{K+1}$ with probability $1 - 2\delta'$, for each link $(j, j+1)$ where $j \in [i-1]$. The lemma follows by taking a union bound over all these links and using a telescoping sum as in (4.8). □

First we show that the with high probability Alice will not output an honest link. Let link $(i, i+1)$ is be "honest", *i.e.,* have a fewer than $\frac{\alpha}{K+1}$ failures, and assume that Alice does not raise alarm for any links upstream of R_i. Now, Lemma 4.3.6 shows that, conditioned on E_i, Alice will not raise an alarm for link $(i, i+1)$ with probability at least $1 - 2\delta'$. Since Alice does not alarm for any links upstream of R_i, we can apply Lemma 4.3.7 to remove the conditioning on E_i. It follows that Alice will not output honest link $(i, i+1)$ with probability at least $1 - 2(i+1)\delta'$. Taking a union bound over all honest links gives that Alice will not alarm for any honest link with probability at least $1 - 2(K+1)^2\delta'$.

Next, we need to show that Alice either will raise an alarm for a link adjacent to Eve or link with more than $\frac{\alpha}{K+1}$ failures. The most interesting part of this proof is the following technical lemma, which we prove in Appendix C.2.4:

Lemma 4.3.8. *If Eve occupies $M \leq \sqrt{(K+1)(1 - \frac{\rho}{\beta}K^2)}$ links and causes a β-fraction of failures in the interval, then there must be a link $(i, i+1)$ that is adjacent to Eve with*

$$\|\mathbf{x}_{i+1}\|_p^p - \|\mathbf{x}_i\|_p^p \geq \tfrac{\beta}{K+1}T$$

Now let link $(i, i+1)$ be the upstream-most link that is adjacent to Eve and has $\|\mathbf{x}_{i+1}\|_p^p - \|\mathbf{x}_i\|_p^p \geq \frac{\beta}{K+1}T$. (Lemma 4.3.8 guarantees the existence of such a link.) We have two cases:

- Suppose Alice did not raise an alarm for a link upstream of R_i. Combining Lemma 4.3.7 and Lemma 4.3.6 it follows that Alice will alarm for link $(i, i+1)$ adjacent to Eve with probability $1 - 2(i+1)\delta'$.

- Suppose Alice did raise an alarm for a link upstream of R_i. It follows from Lemma 4.3.6 that there is some link $(j, j+1)$ for $j \leq [i-1]$ where, with probability $1 - 2\delta'$,

$$\frac{\alpha}{K+1} \leq \|\mathbf{x}_{j+1}\|_p^p - \|\mathbf{x}_j\|_p^p = D_{j+1} - D_j + \|\mathbf{a}_{j+1}\|_p^p - \|\mathbf{a}_j\|_p^p$$

  where the equality comes from applying identity (4.6). Now if link $(j, j+1)$ is adjacent to Eve, it follows that Alice alarms for a link adjacent to Eve, and we are done. Thus, suppose that link $(j, j+1)$ is *not* adjacent to Eve. Then, it follows that no new packets could have been added to this link, and so we have that $\|\mathbf{a}_{j+1}\|_p^p = \|\mathbf{a}_j\|_p^p$. Thus, if link $(j, j+1)$ is *not* adjacent to Eve, then Alice must have raised an alarm for a link with $D_{j+1} - D_j \geq \frac{\alpha}{K+1}$ failures, as required.

Combining these cases, we see that with probability at least $1 - 2(K+1)\delta'$, Alice will either raise an alarm for a link that is either (a) adjacent to Eve, or (b) has more than $\frac{\alpha}{K+1}$ failures.

Sizing the sketches. Finally, to ensure that (α, β, δ)-statistical security holds, it suffices to take $\delta' = \delta/4(K+1)^2$. Next, recall that Lemma 4.3.6 requires sketches that (ε_i, δ')-estimate the p^{th} moment with $\varepsilon_i = \frac{1}{2i}\frac{\beta-\alpha}{2\beta+\alpha}$. For $i \in [K+1]$ it suffices to take sketches $\mathbf{w}_i$, $\mathbf{w}_i^A$ of size $N_i \times b$ where $N_i = O(\frac{1}{\varepsilon_i^2}\log(\frac{1}{\delta'}))$ and $b = O(\log(T))$. Substituting in the values for ε_i, δ' gives us (4.5) as required. □

Efficiency. We remark that, for a given interval of length T, this protocol requires $O(K^2 \log T + n)$ storage overhead at Bob and each intermediate node, while the storage overhead at Alice is $O(K^3 \log T + Kn)$. The communication overhead of the protocol is two control messages of length $O(K^3 \log T + Kn)$ each for every T packets sent.

4.4 Lower bounds

We now argue that in any secure per-packet FL scheme Alice requires shared keys with Bob and the intermediate nodes, and Alice, Bob and each intermediate node must perform cryptographic operations. We only argue for intermediate nodes $R_2, \ldots, R_K$; R_1 is a border case which requires neither keys nor crypto because we assume Alice is always honest.

4.4.1 Failure Localization Needs Keys at Each Node

Since FL provides strictly stronger security guarantees than path-quality monitoring, it follows from the results in Chapter 3 that in any secure FL protocol, Alice and Bob must have shared keys. We also have the following theorem that proves that in any secure FL protocol, *each* intermediate node must share keys with some Alice:

Theorem 4.4.1. *Suppose* Init *generates some auxiliary information* aux_i *for each node* R_i *for* $i = 1, ..., K, \mathsf{Alice}, \mathsf{Bob}$. *A FL protocol cannot be (per-packet or statistical) secure if there is any node* $i \in \{2, \ldots, K\}$ *such that* $(\mathsf{aux}_{\mathsf{Alice}}, \mathsf{aux}_1, \ldots, \mathsf{aux}_{i-1})$ *and* aux_i *are independent.*

Proof. Suppose R_i has aux_i that is independent of $(\mathsf{aux}_{\mathsf{Alice}}, \ldots, \mathsf{aux}_{i-1})$. Then, the following two cases are indistinguishable from Alice's view: (a) Node $R_{i+1}$is malicious and blocks communication on link $(i, i+1)$, and (b) Eve occupies node R_{i-1}, and drops packets while simulating case (a) by picking an independent aux_i' and running $R_i(\mathsf{aux}_i')$ while pretending as if $(i, i+1)$ is down. These two cases are indistinguishable because aux_i is independent of $(\mathsf{aux}_{\mathsf{Alice}}, \ldots, \mathsf{aux}_{i-1})$, and so Alice will localize the failure to the same link in both case (a) and (b). But this breaks security, since R_{i+1}, R_{i-1} do not share a common link. □

4.4.2 Failure Localization Needs Crypto at Each Node

In Chapter 3 , we give a reduction from one-way functions to secure PQM, proving:

Theorem 4.4.2 (From Chapter 3)**.** *The existence of a per-packet secure PQM protocol implies the existence of an infinitely-often one-way function (i.o.-OWF).*

Since one-way functions are equivalent to many cryptographic primitives (in the sense that these primitives exist if and only if one-way functions exist [61]), this result can be interpreted to mean that nodes participating in any secure PQM protocol must perform cryptographic computations. Since FL gives a strictly stronger security guarantee than PQM, we also have that in any FL protocol, some node on the data path must perform cryptography. However, Theorem 4.4.2 only implies that the *entire system* performs cryptography. We want to prove that any secure FL protocol requires *each intermediate node* $R_1, \ldots, R_K$ to perform cryptography. Because it is not clear even how to formalize this in full generality, we instead apply the methodology of Impagliazzo and Rudich [62] to do this for *black-box* constructions of FL protocols from a random oracle RO. We model "performing cryptography" as querying the random oracle, and show that in such a secure FL protocol *each node* must query the RO.

In [62], Impagliazzo and Rudich showed that there can be no secure black-box construction of key agreement (KA) from a random oracle. They argued that if any such KA construction is secure, then it must also be secure in a relativized world where every party has access to a random oracle RO, and a PSPACE oracle. (A PSPACE oracle solves any 'PSPACE-complete problem, *e.g.,* True Quantified Boolean Formulae (TQBF).) Intuitively, in this $(\mathsf{PSPACE}, \mathsf{RO})$ world, every computation is easy to invert *except* for those computed by the RO. They obtain their result by showing, for every possible black-box construction of KA from a random oracle, that there exists an efficient algorithm (relative to $(\mathsf{PSPACE}, \mathsf{RO})$) that breaks the security of KA. Using the same reasoning, any secure black-box FL protocol constructed from a RO must remain secure even relative to a $(\mathsf{RO}, \mathsf{PSPACE})$ oracle. Then, to obtain our result, it suffices to exhibit an efficient algorithm (relative to $(\mathsf{PSPACE}, \mathsf{RO})$) that breaks security of any black-box FL protocol where one node does not call RO. We do this below.

We will use the notion of an *exchange* to denote a data packet and all the FL-protocol-related messages associated with that packet. Because our game is sequential (see Section 4.2), Alice's must decide to localize a link ℓ or output $\surd$ before the next exchange begins. Let $\langle R_{i-1}, R_i \rangle_j$ denote the distribution of all messages sent and received along link $(i-1, i)$ during the j'th exchange. We sometimes refer to these messages as a transcript for the j'th exchange. Because we allow the nodes to keep state, this distribution may depend on what happened in all previous exchanges, $\langle R_{i-1}, R_i \rangle_1, \ldots, \langle R_{i-1}, R_i \rangle_{j-1}$. We now prove that a per-packet FL protocol with $2r = O(\log n)$ messages per exchange must invoke the random oracle at every node. We assume that the number of messages per exchange is even, and that odd messages go from R_{i-1} to R_i and even messages go from R_i to R_{i-1}.We note that protocols where number of messages per packet grows with n are impractical and so "practical" protocols should use $2r = O(1)$ messages per exchange. (See Remark 4.4.7 below on the possibility of extending this result to statistical security and/or protocols with $\omega(\log n)$ messages per exchange.)

Theorem 4.4.3. *Fix a fully black-box per-packet FL protocol that uses access to a random oracle* RO*, where at least one node* R_i *for* $i \in \{2, \ldots, I\}$ *never calls the* RO *and where the maximum number of messages per exchange is* $O(\log n)$*. Then there exists an efficient algorithm relative to* $(\mathsf{PSPACE}, \mathsf{RO})$ *that breaks the security of the scheme with non-negligible probability over the randomness of* RO *and the internal randomness of the algorithm.*

The proof of Theorem 4.4.3 is quite technical and is deferred to Appendix C.3. We sketch

the proof, which resembles that of Theorem 4.4.1. Eve controls node R_{i-1} and impersonates R_i, but now aux_i is secret, so Eve must first *learn* aux_i:

1. *Learning to impersonate.* Sitting at R_{i-1}, Eve observes t exchanges (t is polynomial in n), where Eve asks Source to transmit a uniformly random data packet. She then uses the learning algorithm of Naor and Rothblum [83] to obtain a pair of impersonator algorithms A', B', whose interaction generates a distribution over transcripts for the $t+1$'th exchange. A' impersonates nodes $\mathsf{Alice}, R_1, \ldots, R_{i-1}$ and B' impersonates nodes $R_i, \ldots, R_K, \mathsf{Bob}$.

2. *Dropping and impersonating.* On the $t+1$'th exchange, for each message m_j going from R_{i-1} to R_i, Eve computes a response herself m_{j+1} using algorithm B' and returns m_{j+1} to R_{i-1}; she does not send any messages to R_i. (More precisely, B' samples m_{j+1} according to the conditional distribution $\langle A', B'\rangle^{j+1} \mid \langle A', B'\rangle^j = (m_1, ..., m_j)$. Here $\langle A', B'\rangle^j$ denotes the first j messages of $\langle A', B'\rangle$. Note that this sampling is efficient in the presence of a PSPACE oracle.)

Now, Eve at R_{i-1} will break security if she manages to use B' to impersonate an *honest* exchange during which link $(i, i+1)$ is down. (This breaks security since link $(i, i+1)$ is not adjacent to R_{i-1}.) The crucial observation is that here, Eve need only impersonate node R_i, and that R_i does not "protect" its secret keys by calling the RO. Intuitively, Eve should be able to impersonate R_i since any computations that R_i does are easy to invert in the $(\mathsf{PSPACE}, \mathsf{RO})$ world. To prove the theorem, we shall show that with non-negligible probability $> (10/\rho)^r = 1/\mathrm{poly}(n)$, the following are 1/100-indistinguishable: (a) Alice's view when link $(i, i+1)$ is down and (b) Alice's view when R_{i-1} drops a packet but impersonates link $(i, i+1)$ being down using B'.

In the following, we define the statistical distance between two random variables X, Y as $\Delta(X, Y) = \frac{1}{2}\sum_{x \in U} |\Pr[X = x] - \Pr[Y = x]|$ where U is the union of the supports of X and Y (for more background on statistical distance, see *e.g.,* [50]).

Recall (Section 4.2) that Alice is allowed to use information from past exchanges to help her decide how to send messages in new exchanges. Fortunately, the algorithm of Naor and Rothblum [83] is specifically designed to deal with this, and guarantees the following:

Lemma 4.4.4 (Based on [83])**.** *Relative to a* $(\mathsf{PSPACE}, \mathsf{RO})$*-oracle, there exists an efficient algorithm that observes at most* $t = O(\frac{n}{\varepsilon^4})$ *honest exchanges* $\langle R_{i-1}, R_i\rangle_{1,...,t}$ *and then, with probability* $> 1 - \varepsilon$*, outputs efficient impersonator algorithms* $R'_0, \ldots, R'_{K+1}$ *such that that an impersonated transcript* $\langle R'_{i-1}, R'_i\rangle_{t+1}$ *(given by simulating the interaction of all the impersonator algorithms) for the exchange* $t+1$ *is distributed* ε*-close in statistical distance to the honest transcript* $\langle R_{i-1}, R_i\rangle_{t+1}$ *for exchange* $t+1$*.*

Suppose Eve obtained an A', B' where we let A' be the collection of algorithms $R'_0, \ldots, R'_{i-1}$ and B' be the collection $R'_i, \ldots, R'_{K+1}$that satisfy the guarantee above. Our first challenge is that the Naor-Rothblum algorithm does *not* guarantee that A', B' generates an impersonated transcript that is statistically close to the "honest" transcript of messages on $(i-1, i)$ when *the observer has access to the* RO. (The "honest" transcript of messages on the link $(i-1, i)$ is generated by interactions of honest $\mathsf{Alice}, R_1, ..., R_K, \mathsf{Bob}$.) Fortunately, with probability ρ^r all the messages sent from R_i to R_{i-1} are computed without access the RO. This happens when *congestion* causes link $(i, i+1)$ to go down for the duration of an exchange (so that R_i, who never calls the RO, has to compute all his upstream messages on his own).

Our next challenge is that Eve has no control, or even knowledge, of when congestion causes this event to occur. Indeed, the distribution generated by A', B' is only guaranteed to be close to the honest transcript overall; there is no guarantee that it is close to the honest transcript

conditioned on congestion on $(i, i+1)$.[8] Fortunately, we can show that with probability ρ^r, A', B' will generate a "useful" impersonated transcript that is ε/ρ^r-statistically close to the honest transcripts conditioned on the event that link $(i, i+1)$ is down. Eve does not necessarily know *when* she impersonates a useful transcript; she simply has to hope that she is lucky enough for this to happen.

The last challenge is that even when Eve is lucky enough to obtain a useful transcript, we still need a guarantee that (a) conditioned on B' generating a useful transcript, using B' to *interact* with the honest algorithm R_{i-1} results in a transcript that is statistically close to (b) the transcript between honest algorithms R_{i-1} and R_i conditioned on link $(i, i+1)$ being down. Unfortunately, the Naor-Rothblum algorithm does not give any guarantees when an honest algorithm *interacts* with an impersonated algorithm for more than 1 round. Thus, we prove that, with probability at least $(\rho/2)^r$, the impersonator algorithm B' interacting with honest $\mathsf{Alice}, ...R_{i-1}$ still generates a useful transcript such that the statistical distance between (a) and (b) is at most 1/100. (This assumes we take ε small enough; $\varepsilon = (\rho/10)^{4r} = 1/\mathrm{poly}(n)$ suffices.)

We address these challenges in the next lemma, which we prove in Appendix C.3. We state a general version of the lemma here, for which we first need a few definitions:

- Let A, B and A', B' be two (different) pairs of algorithms such that the statistical difference between the transcripts $\langle A, B\rangle$ and $\langle A', B'\rangle$ is bounded by ε. We assume *a priori* that A, B can share randomness, say by accessing a common random oracle, and so can A', B'.

- Let $(\langle A, B\rangle, \mathsf{view}_A(\langle A, B\rangle))$ be the joint distribution of transcripts $\langle A, B\rangle$ and the internal randomness of party A, which we call view_A (which includes both randomness that is shared with B and independent randomness). For a fixed τ, we will let $\mathsf{view}_A(\tau)$ be the distribution of the internal randomness of A conditioned on outputting the transcript τ.

- To deal with interaction, we let $\langle A, B'\rangle = (m_1, \ldots, m_r)$ be the distribution over transcripts where for each message m_j sent by A is computed honestly, while each m_j sent by B' is computed by *pretending that the partial transcript so far* $\sigma_i = (m_1, \ldots, m_{j-1})$ *came from the distribution* $\langle A', B'\rangle$ *and sampling the next message* m_j *consistent with* $\langle A', B'\rangle$; more formally B' samples m_j according to the conditional distribution $(\langle A', B'\rangle^j \mid \langle A', B'\rangle^{j-1} = \sigma_{j-1})$.[9] Here $\langle A', B'\rangle^j$ denotes the first j messages of $\langle A', B'\rangle$.

We are finally ready for the statement of the Lemma.

Lemma 4.4.5. *Suppose that* $\Delta(\langle A, B\rangle, \langle A', B'\rangle) \le \varepsilon$*, and there exist events* $E_1, \ldots, E_r$ *over the internal randomness of* A, B *such that (1)* $\forall j$*, conditioned on* E_j*, the first* j *messages from* B *to* A *are independent of* A*'s internal randomness, and (2)* $\Pr[E_j \mid E_{j-1}] \ge \rho$*. Set* $\varepsilon = (\rho/10)^{4r}$*, Then there exist* $\eta \ge (\rho/2)^r$*, and distributions over the transcripts* Y, Z *such that* $\langle A, B'\rangle$ *is a convex combination* $\eta Y + (1-\eta)Z$ *and*

$$\Delta((Y, \mathsf{view}_A(Y)), (\langle A, B\rangle, \mathsf{view}_A(\langle A, B\rangle) \mid E_r)) \le 1/100$$

Lemma 4.4.5 tells us that, with probability η, $\langle A, B'\rangle$ will generate a "useful" transcript Y that is $\sqrt{\varepsilon}(10/\rho)^r$-statistically close to the honest transcript $\langle A, B\rangle$ conditioned on event E_r occurring. (Z is the "not useful" transcript that is generated with probability $1-\eta$.) We can now apply Lemma 4.4.5 by setting:

- A to be honest algorithms $R_0, R_1, \ldots, R_{i-1}$.

[8] For this reason, Eve cannot simply use R'_i (instead of $R'_i, ..., R'_K, \mathsf{Bob}'$) to impersonate the honest R_i conditioned on link $(i, i+1)$ being down.

[9] In general this is not efficient, but it is efficient because in our setting Eve has access to a PSPACE oracle.

- B to be honest algorithms $R_i, \ldots, R_{K+1}$.
- A' to be the impersonator algorithms $R'_0, \ldots, R'_{i-1}$ given by Lemma 4.4.5.
- B' to be the impersonator algorithms $R'_i \ldots, R'_{K+1}$ given by Lemma 4.4.5.
- E_j to be the event that link $(i, i+1)$ is congested for the messages $1, \ldots, j$ that are sent downstream from B to A. (Then, E_r is the event that link $(i, i+1)$ is down for the duration of an exchange of length $r = O(\log n)$ messages.)

Now, notice that since R_i does not query the random oracle, conditioned on E_j the first j messages of B are independent of A because they are computed by R_i only. Next, note that $\Pr[E_j \mid E_{j-1}] = \rho$ because each message is lost to congestion independently.

To combine everything, set $\varepsilon = (\rho/10)^{4r}$ and apply Lemma 4.4.4 to find that with probability at least $\geq (1 - \varepsilon)$ we get A', B' that is ε-close to $\langle A, B \rangle$ (notice that Eve is efficient with this setting of ε). Conditioned on this happening, by Lemma 4.4.5 we get that with probability $(\rho/2)^r = 1/\text{poly}(n)$, Eve is lucky enough to generate a useful transcript such that (a) the view of Alice when Eve drops a packet at R_{i-1} and impersonates using $R'_i, \ldots, R'_{K+1}$ is 1/100-indistinguishable from the situation (b) where link $(i, i+1)$ is completely down for the duration of an exchange. Since Alice should localize the same link in case (a) and (b) for all but a 1/100 fraction of the time, this breaks security since link $(i, i+1)$ is not adjacent to Eve at R_{i-1}.

Statistical security.

Our lower bounds in the statistical setting are more subtle. First of all, from Chapter 3 the analog of Theorem 4.4.2 also holds, showing that the *entire system* needs to "perform cryptography".

Theorem 4.4.6 (From Chapter 3)**.** *The existence of a (α, β, δ)-statistically secure failure detection scheme for constants α, β, δ implies the existence of an infinitely-often one-way function (i.o.-OWF).*

However, we run into trouble when we try to show that cryptography is required at *each intermediate node*. It turns out that Definition 4.2.3 does *not* inherently require complexity-based cryptography at intermediate nodes. We sketch a statistically secure FL protocol where the intermediate nodes $R_1, \ldots, R_K$ use only information-theoretically secure primitives (although Alice and Bob still use regular MAC's). While this protocol is completely impractical in terms of communication and storage overhead, we present it here to demonstrate the subtleties of Definition 4.2.3.[10]

Remark 4.4.7 (Impractical "crypto-free" statistical FL protocol.)**.** *The protocol uses one-time MACs (OTMAC), information-theoretic objects that have the same properties as regular MACs except that they can only be used a single time. (OTMACs and can be constructed from universal hash functions [23].) Each node R_i shares pairwise keys with Alice. All the intermediate nodes and Bob store each packet that Alice sends to Bob. For each packet, Bob replies with an ack signed using a regular MAC. At the end of the interval, Alice counts the number of acks that she either fails to receive, or are invalid. The first time this count exceeds a β-fraction, Alice sends a "report request" message that is signed using a OTMAC to $R_1, \ldots, R_K, R_{K+1}$. Each node $R_1, \ldots, R_K$*

[10] In concurrent work, Wong et al. [107] propose a statistical FL scheme where no cryptography is performed *during an interval*. Instead, they precompute shared secrets that are appended to packets over the course of an interval and are used guarantee security. The secrets must refreshed periodically, which requires cryptographic participation by the intermediate nodes. This contrasts with the impractical scheme we describe here, which truly *never* requires any intermediate node to perform crypto.

responds with a report of every single packet they have witnessed, that is "onion signed" using the OTMAC (as in Section 4.3.1). Alice uses these reports in the usual way to localize link ℓ adjacent to Eve. From this point onwards Alice simply counts valid acknowledgments from Bob, and blames link ℓ each time the count exceeds a β fraction.

The protocol satisfies Definition 4.2.3 because the probability that the failure rate at any link exceeds β by congestion alone is negligible. Since we do not allow Eve to move during the security game, if Alice successfully localizes Eve to link ℓ once, it means it must have been Eve's fault, and so from then on Alice can always blame all failures on link ℓ. As noted above, similar "impractical" protocols exist for per-packet protocols with $\omega(\log n)$ additional messages per packet (since all $\omega(\log n)$ messages are lost to congestion with only negligible probability), except that we replace the idea of "exceeding β fraction of failures" with "losing an entire exchange due to congestion". We may interpret this as follows:

1. It is unreasonable to assume that the failure rate at a link exceeds β only due to adversarial behavior (*i.e.,* Eve). For example, occasionally congestion might spike, or a router might malfunction or go down due maintenance, causing more than a β-fraction of packets to be dropped. If we assume such events happen with non-negligible probability, we can adapt the proof of Theorem 4.4.2 to show that cryptography is necessary at intermediate nodes for statistical security. As a corollary, if Eve can control congestion at links she does not occupy, then we need cryptography at every intermediate node. Our FL protocols remain secure even under the strongest such definition, where the failure rate on a link not occupied by Eve can exceed β.

2. We can take this issue outside of our model. If we say that it is reasonable that Eve cannot move during the security game, and that the failure rate cannot exceed β on a link that Eve does not control, then, as we showed above, there exist protocols where the intermediate nodes do not use complexity-based cryptography. However, we must be cognizant that in the real world there can be multiple adversaries that we would like to localize correctly, or the adversary may be able to move from one link to another. If protocols that do not use cryptography at intermediate nodes are to remain secure after Eve moves (and learns the key of previous nodes she occupied), then the keys at each node should be refreshed periodically. This key refresh process would require each intermediate node to use cryptography.

4.5 Open problems

We gave lower bounds on the key-management and cryptographic overhead of secure FL protocols. While our statistical FL protocol based on sketching requires fairly small storage overhead, the interesting problem of bounding the *storage* requirements in an FL protocol is still open. Furthermore, our results here only apply to FL on *single symmetric paths* between a single sender-receiver pair. An interesting question would be to consider FL for *asymmetric paths*, where the packets Bob sends back to Alice may take a different path than the packets that Alice sends to Bob. Another direction is to consider FL in networks where packets can travel simultaneously on *multiple paths*, as in the SMT framework [32]. Recently, Amir *et al.* [8] presented a protocol for this setting, that optimizes for low *communication overhead*. Designing such protocols that optimize for low *storage and computational overhead* remains an interesting open question.

References

[1] Bad ISPs that cause trouble for BitTorrent clients. `http://www.azureuswiki.com/index.php/Bad_ISPs`.

[2] RIPE certification task force. `http://www.ripe.net/ripe/tf/certification/`.

[3] Keynote launches new SLA services, June 2001. `http://investor.keynote.com/phoenix.zhtml?c=78522&p=irol-newsArticle_Print&ID=183745`.

[4] Traceroute. *Available: http://costard.lbl.gov/cgi-bin/man/man2html?traceroute+8*, 2001.

[5] D. Achlioptas. Database-friendly random projections. In *PODS*, pages 274–281, 2001.

[6] R. Ahlswede and A. Winter. Strong converse for identification via quantum channels. *IEEE Trans. IT*, 48(3):569–579, 2002.

[7] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *STOC*, pages 20–29, 1996.

[8] Y. Amir, P. Bunn, and R. Ostrovsky. Authenticated adversarial routing. In *Theory of Cryptography Conference, TCC*, 2009.

[9] D. Angulin and L. Valiant. Fast probabilistic algorithms for hamiltonian circuits and matchings. *J. of Computer and System Sciences*, 19:155–193, 1979.

[10] K. Argyraki, P. Maniatis, O. Irzak, A. Subramanian, and S. Shenker. Loss and delay accountability for the Internet. *ICNP*, 2007.

[11] I. Avramopoulos, H. Kobayashi, R. Wang, and A. Krishnamurthy. Highly secure and efficient routing. In *IEEE INFOCOM*, 2004.

[12] I. Avramopoulos and J. Rexford. Stealth probing: Data-plane security for IP routing. *USENIX*, 2006.

[13] B. Awerbuch, D. Holmer, C. Nita-Rotaru, and H. Rubens. An on-demand secure routing protocol resilient to byzantine failures. In *ACM WiSE*, 2002.

[14] H. Ballani, P. Francis, and X. Zhang. A study of prefix hijacking and interception in the Internet. In *ACM SIGCOMM*, 2007.

[15] S. Balon and G. Leduc. Can forwarding loops appear when activating iBGP multipath load sharing? In *AINTEC*, 2007.

[16] B. Barak, S. Goldberg, and D. Xiao. Protocols and lower bounds for failure localization in the Internet. In *IACR EUROCRYPT*, 2008.

[17] F. Bergadano, D. Cavagnino, and B. Crispo. Chained stream authentication. In *Workshop on Selected Areas in Cryptography*, pages 144–157, 2000.

[18] D. J. Bernstein. Polynomial evaluation and message authentication. Technical report, http://cr.yp.to/ Document ID: b1ef3f2d385a926123e1517392e20f8c., October 2007.

[19] S. Bradner. Key words for use in RFCs to indicate requirement levels. RFC 2119, March 1997.

[20] B. Briscoe. FLAMeS, Tech. Rep., BT Research, 2000. `http://www.labs.bt.com/people/briscorj/papers.html`.

[21] K. Butler, T. Farley, P. McDaniel, and J. Rexford. A survey of BGP security issues and solutions. Technical report, ATT Labs-Research, 2004.

[22] M. Caesar and J. Rexford. BGP routing policies in ISP networks. *IEEE Network Magazine, special issue on Interdomain Routing*, Nov/Dec 2005.

[23] J. L. Carter and M. N. Wegman. Universal classes of hash functions. *JCSS*, 18(2):143–154, 1979.

[24] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. *Theoretical Computer Science*, 312(1):3–15, 2004.

[25] S. Cheung. An efficient message authentication scheme for link state routing. In *Annual Computer Security Applications Conference*, pages 90–98, 1997.

[26] D. Clark, J. Wroclawski, K. Sollins, and R. Braden. Tussle in cyberspace: defining tomorrow's Internet. *IEEE/ACM Transactions on Networking*, 13(5):462– 475, June 2005.

[27] G. Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms*, 55(1), 2005.

[28] M. Crovella and B. Krishnamurthy. *Internet Measurement*. Wiley, 2006.

[29] J. Daemen and V. Rijmen. A new MAC construction ALRED and a specific instance ALPHA-MAC. In *FSE: Fast Software Encryption*, volume 3557, pages 1–17. Springer, 2005.

[30] R. R. Dakdouk, S. Salihoglu, H. Wang, H. Xie, and Y. R. Yang. Interdomain routing as social choice. In *Incentive-Based Computing (IBC)*, 2006.

[31] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346 (Proposed Standard), 2006.

[32] D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly secure message transmission. *J. of the ACM*, 40(1), 1993.

[33] N. G. Duffield and M. Grossglauser. Trajectory sampling for direct traffic observation. *IEEE/ACM Trans. Networking*, 9(3), 2001.

[34] N. Feamster, R. Johari, and H. Balakrishnan. Implications of autonomy for the expressiveness of policy routing. In *ACM SIGCOMM*, 2005.

[35] J. Feigenbaum, D. R. Karger, V. Mirrokni, and R. Sami. Subjective-cost policy routing. In X. Deng and Y. Ye, editors, *First Workshop on Internet and Network Economics*, 2005.

[36] J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker. A BGP-based mechanism for lowest-cost routing. *Distributed Computing*, 18(1), July 2005.

[37] J. Feigenbaum, V. Ramachandran, and M. Schapira. Incentive-compatible interdomain routing. In *Conference on Electronic Commerce*, pages 130 – 139, 2006.

[38] J. Feigenbaum, R. Sami, and S. Shenker. Mechanism design for policy routing. *Distributed Computing*, 18(4):293–305, 2006.

[39] J. Feigenbaum, M. Schapira, and S. Shenker. *Algorithmic Game Theory*, chapter Distributed Algorithmic Mechanism Design. Cambridge University Press, 2007.

[40] M. Feldman, J. Chuang, I. Stoica, and S. Shenker. Hidden-action in multi-hop routing. In *EC '05: Proceedings of the 6th ACM conference on Electronic commerce*, pages 117–126, 2005.

[41] L. Gao, T. Griffin, and R. Rexford. Inherently safe backup routing with BGP. *IEEE Infocomm*, 2001.

[42] L. Gao and R. Rexford. Stable Internet routing without global coordination. *IEEE/ACM Trans. on Network.*, 2001.

[43] R. Gao, C. Dovrolis, and E. Zegura. Interdomain ingress traffic engineering through optimized AS-path prepending. In *IFIP Networking*, 2005.

[44] V. Gill, J. Heasley, and D. Meyer. The generalized TTL security mechanism (gtsm). RFC 3682, 2004.

[45] S. Goldberg and S. Halevi. Rational ASes and traffic attraction: Incentives for honestly announcing paths in BGP. Technical Report TR-813-08, Princeton University, Dept. of Computer Science, Feb. 2008.

[46] S. Goldberg, S. Halevi, A. D. Jaggard, V. Ramachandran, and R. N. Wright. Rationality and traffic attraction: Incentives for honest path announcements in BGP. In *ACM SIGCOMM*, 2008.

[47] S. Goldberg and J. Rexford. Security vulnerabilities and solutions for packet sampling. *IEEE Sarnoff Symposium*, 2007.

[48] S. Goldberg, D. Xiao, B. Barak, and J. Rexford. A cryptographic study of secure fault detection in the internet. *Princeton University, Department of Computer Science, Technical Report*, 2007.

[49] S. Goldberg, D. Xiao, E. Tromer, B. Barak, and J. Rexford. Path quality monitoring in the presence of adversaries. In *SIGMETRICS*, June 2008.

[50] O. Goldreich. *Foundations of Cryptography.* Cambridge University Press, 2007.

[51] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. of the ACM*, 33(4):210–217, 1986.

[52] G. Goodell, W. Aiello, T. Griffin, J. Ioannidis, P. McDaniel, and A. Rubin. Working around BGP: An incremental approach to improving security and accuracy of interdomain routing. In *Network and Distributed System Security Symposium*, 2003.

[53] T. Griffin, F. B. Shepherd, and G. Wilfong. The stable paths problem and interdomain routing. *IEEE/ACM Trans. on Network.*, April 2002.

[54] J. Hastad, R. Impagliazzo, L. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM J. of Computing*, 1999.

[55] J. He and J. Rexford. Towards Internet-wide multipath routing. *IEEE Network Magazine Special Issue on Scalablity*, March 2008.

[56] A. Heffernan. Protection of BGP sessions via the TCP MD5 signature option. RFC 2385, 1998.

[57] K. J. Houle and G. M. Weaver. Trends in denial of service attack technology. Technical report, CERT Coordination Center, October 2001.

[58] G. Huston. Interconnection, peering, and settlements. In *Internet Global Summit (INET)*, June 1999.

[59] IETF. Packet sampling working group. *http://www.ietf.org/html.charters/psamp-charter.html*.

[60] IETF. Working Group on IP Performance Metrics. *http://www.ietf.org/html.charters/ippm-charter.html*.

[61] R. Impagliazzo and M. Luby. One-way functions are essential for complexity based cryptography. *FOCS*, 1989.

[62] R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *STOC*, 1989.

[63] P. Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. of the ACM*, 53(3):307–323, 2006.

[64] V. Jacobson. Congestion avoidance and control. *ACM SIGCOMM*, 18(4), 1988.

[65] A. D. Jaggard, V. Ramachandran, and R. N. Wright. Towards a realistic model of incentives in interdomain routing: Decoupling forwarding from signaling. Technical Report 2008-02, DIMACS, Apr. 2008.

[66] S. Kent, C. Lynn, and K. Seo. Secure border gateway protocol (S-BGP). *J. Selected Areas in Communications*, 18(4):582–592, April 2000.

[67] H. Krawczyk, M. Bellare, and R. Canetti. HMAC : Keyed-Hashing for Message Authentication. RFC 2104, 1997.

[68] T. D. Krovetz. *Software-optimized universal hashing and message authentication.* PhD thesis, University of California, Davis, 2000.

[69] P. Laskowski and J. Chuang. Network monitors and contracting systems: competition and innovation. In *ACM SIGCOMM*, 2006.

[70] R. Lavi and N. Nisan. Online ascending auctions for gradually expiring items. In *ACM-SIAM Symp. on Discrete Algorithms, SODA*, 2005.

[71] K. Levchenko. Chernoff bound. Online. www-cse.ucsd.edu/ klevchen/techniques/chernoff.pdf.

[72] H. Levin, M. Schapira, and A. Zohar. The strategic justification for BGP. Technical report, Hebrew University of Jerusalem, 2006.

[73] H. Levin, M. Schapira, and A. Zohar. Interdomain routing and games. In *ACM STOC*, May 2008.

[74] X. Liu, X. Yang, D. Wetherall, and T. Anderson. Efficient and secure source authentication with packet passports. In *SRUTI'06: Steps to Reducing Unwanted Traffic on the Internet*. USENIX, 2006.

[75] M. Luckie, K. Cho, and B. Owens. Inferring and debugging path MTU discovery failures. In *Internet Measurement Conference*, 2005.

[76] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson. User-level Internet path diagnosis. *SOSP*, 2003.

[77] Z. Mao, J. Rexford, J.Wang, and R. H. Katz. Towards an accurate AS-level traceroure tool. In *ACM SIGCOMM*, 2003.

[78] D. A. McGrew and J. Viega. The security and performance of the galois/counter mode (GCM) of operation. In *INDOCRYPT*, pages 343–355. Springer-Verlag, 2004.

[79] D. Mills, A. Thyagarajan, and B. Huffman. Internet timekeeping around the globe. *Proc. PTTI*, pages 365–371, 1997.

[80] I. Mironov, M. Naor, and G. Segev. Sketching in adversarial environments. In *STOC*, 2008.

[81] A. T. Mizrak, Y.-C. Cheng, K. Marzullo, and S. Savage. Detecting and isolating malicious routers. *IEEE Transactions on Dependable and Secure Computing*, 3(3):230–244, 2006.

[82] M. Motiwala, A. Bavier, and N. Feamster. Network troubleshooting: An in-band approach (poster). *NSDI*, 2007.

[83] M. Naor and G. N. Rothblum. Learning to impersonate. In *International Conf. on Machine Learning*, 2006.

[84] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35(1-2):166–196, 2001.

[85] A. Nucci. Skype detection: Traffic classification in the dark, 2006. `http://www.narus.com/_pdf/news/Converge-Skype%20Detection.pdf`.

[86] V. Padmanabhan and D. Simon. Secure traceroute to detect faulty or malicious routing. *HotNets-I*, 2002.

[87] D. C. Parkes and J. Shneidman. Specification faithfulness in networks with rational nodes. In *ACM PODC*, 2004.

[88] V. Paxson. End-to-end Internet packet dynamics. *IEEE Trans. on Networking*, 7(3):277–292, 1999.

[89] R. Perlman. *Network Layer Protocols with Byzantine Robustness*. PhD thesis, MIT, 1988.

[90] A. Perrig, R. Canetti, D. Song, and J. D. Tygar. Efficient authentication and signing of multicast streams over lossy channels. In *IEEE Security and Privacy Symposium*, 2000.

[91] A. Ramachandran and N. Feamster. Understanding the network-level behavior of spammers. *ACM SIGCOMM*, 2006.

[92] Y. Rekhter, T. Li, and S. Hares. A border gateway protocol 4 BGP-4. RFC 4271, January 2006.

[93] P. Rogaway. Formalizing human ignorance: Collision-resistant hashing without the keys. In *Vietcrypt*, volume 4341, pages 211–228. Springer, 2006.

[94] M. Roughan. Fundamental bounds on the accuracy of network performance measurements. In *ACM SIGMETRICS*, 2005.

[95] J. Sommers, P. Barford, N. Duffield, and A. Ron. Improving accuracy in end-to-end packet loss measurement. In *ACM SIGCOMM*, 2005.

[96] J. Sommers, P. Barford, N. Duffield, and A. Ron. Accurate and efficient SLA compliance monitoring. In *ACM SIGCOMM*, 2007.

[97] D. Song, D. Wagner, and X. Tian. Timing analysis of keystrokes and SSH timing attacks. In *10th USENIX Security Symposium*, 2001.

[98] J. Stone and C. Partridge. When the CRC and TCP checksum disagree. In *ACM SIGCOMM*, 2000.

[99] L. Subramanian, V. Roth, I. Stoica, S. Shenker, and R. H. Katz. Listen and Whisper: Security mechanisms for BGP. In *NSDI*, 2004.

[100] M. Thorup and Y. Zhang. Tabulation based 4-universal hashing with applications to second moment estimation. In *SODA*, pages 615–624, 2004.

[101] F. Wang and L. Gao. On inferring and characterizing Internet routing policies. In *ACM IMC '03*, pages 15–26. ACM, 2003.

[102] H. Wang, R. K. Chang, D.-M. Chiu, and J. C. Lui. Characterizing the performance and stability issues of the AS path prepending method. In *ACM SIGCOMM Asia Workshop*, 2005.

[103] H. Wang, H. Xie, Y. R. Yang, L. E. Li, Y. Liu, and A. Silberschatz. On the stability of rational, heterogeneous interdomain route selection. In *ICNP*, 2005.

[104] M. N. Wegman and J. L. Carter. New hash functions and their use in authentication and set equality. *JCSS*, 22:265–279, 1981.

[105] D. Wendlandt, I. Avramopoulos, D. G. Andersen, and J. Rexford. Don't secure routing protocols, secure data delivery. In *ACM SIGCOMM HotNets-V*, 2006.

[106] R. White. Deployment considerations for secure origin BGP (soBGP). draft-white-sobgp-bgp-deployment-01.txt, June 2003, expired.

[107] E. L. Wong, P. Balasubramanian, L. Alvisi, M. G. Gouda, and V. Shmatikov. Truth in advertising: Lightweight verification of route integrity. In *PODC*, 2007.

[108] J. Xu. Tutorial on network data streaming. In *ACM SIGMETRICS*, 2008.

[109] X. Zhang, A. Jain, and A. Perrig. Packet-dropping adversary identification for data plane security. In *CoNEXT: Conference on emerging Networking EXperiments and Technologies*, December 2008.

Appendix A

Honest Path Announcements in BGP

A.1 Formalizing "No Incentive to Lie"

As we mentioned several times in the text, the formal notion of "no incentive to lie" that we use for some of our positive results is different from "incentive compatibility in ex-post Nash equilibrium" that was used in prior work; see [87]. Here we explain this difference in more detail.

A.1.1 Ex-Post Nash

The notion of ex-post Nash equilibrium expands upon the usual Nash equilibrium to distributed settings, where players may not have full information on each other's preferences. Below we let θ_i denote the private information of node i. (In our setting, this consists of the node's valuation and attraction functions.)

Let $s_i(\theta_i)$ be a strategy for node i; which takes as input i's private information and then describes the actions that node i takes in each round of the game. (For example, a BGP-compliant strategy was described in Definition 2.2.1.) A *strategy profile* $\mathbf{s} = (s_1, s_2, \ldots, s_k)$ is a tuple consisting of one strategy s_i for each node i. Together with the private inputs θ of all nodes and a particular schedule t, such a strategy profile $\mathbf{s}$ determines a particular execution of the interdomain routing game. Below we denote by $g_t(\mathbf{s}(\theta))$ the outcome of this execution. (This notation assumes that the execution converges to a stable outcome; otherwise we arbitrarily define the outcome as the first non-transient global state in this execution.)

We say that the strategy profile $\mathbf{s}$ is an *ex-post Nash equilibrium* if for each node i, every possible alternate strategy s_i' that i could have, every fair schedule t, and for all possible values of the private information $\theta = (\theta_1 \ldots \theta_k)$, it holds that

$$\begin{aligned} & u_i(g_t(s_1(\theta_1), \ldots, s_i(\theta_i), \ldots, s_k(\theta_k))) \\ & \quad \geq \quad u_i(g_t(s_1(\theta_1), \ldots, s_i'(\theta_i), \ldots, s_k(\theta_k))), \end{aligned}$$

In other words, a strategy profile $\mathbf{s}$ is in ex-post Nash equilibrium if, regardless of the underlying private information of all other nodes, each node i obtains at least as great a utility by executing strategy s_i contained in s rather than some other strategy s_i'. This is much stronger than a Nash equilibrium, in which nodes are assumed to know the private information of other nodes, and weaker than a dominant-strategy equilibrium, in which nodes have a single strategy that is best to execute regardless of the other players' strategies (and not just their private information). Dominant-strategy equilibrium appeared in some of the initial work on mechanism

design and routing [84, 36]. Ex-post Nash equilibrium, as in [37, 39, 73], can be used to capture *rational specification faithfulness*. If we let the strategy profile s contain the strategies that nodes "follow a protocol as specified," then showing that s is an ex-post Nash equilibrium amounts to showing that nodes have no incentive to unilaterally deviate from following the protocol.

We note that ex-post Nash equilibrium does not address deviations by more than one node, although the topic of collusion-proof ex-post Nash equilibrium is addressed in [39, 73].

A.1.2 Partially-Specified Strategies

As defined above, ex-post Nash equilibrium requires that all nodes follow a fully-specified strategy profile. In our setting, this means in particular that all the actions of the nodes (including their filtering policies) must be spelled out in this strategy profile. We stress that this requirement goes well beyond requiring that all nodes comply with the BGP specification [92]. In particular, a BGP-compliant implementation allows node to use arbitrary ingress and egress filtering (as long as the select paths based on their ranking functions), but such arbitrary filtering is not consistent with the strategies in prior work [37, 39, 73].

Insisting that all nodes follow a fully-specified strategy-profile may not be realistic in large distributed systems, where protocols are only partially specified and many options are left for the individual implementations. (Indeed, avoiding over-specification is crucial for RFCs; see [19, §6].) We therefore describe BGP-compliance in Definition 2.2.1 as a *property* of a strategy (or, equivalently, as a "set of allowed strategies").

A.1.3 Solution Concepts

Extending the formal treatment to a set of strategy allows one to define a variety of solution concepts. Below we mention two such concepts that are used in our paper.

Ideally, one would have wanted to augment the notion of ex-post Nash, allowing also part of the *strategy* itself (*e.g.,* the export rules) and not just the valuation and attraction functions to be treated as private inputs. Namely, we would have liked to have a single (fully specified) strategy profile, such that every node i has an incentive to follow its strategy even when other nodes do not follow theirs, as long as all nodes follow "allowed strategies". Hence, this notion lies somewhere in between ex-post Nash and a dominant-strategy (and in particular it implies the standard ex-post Nash concept). We note that our positive result for traffic-volume attraction in Theorem 2.4.1 actually meets this strong solution concept. (The positive result for customer attraction in Theorem 2.6.1 achieves a similar concept, but that result is significantly weaker since it only addresses stable outcomes.)

Unfortunately, for the case of "generic attractions" in Theorem 2.5.1 we are not able to achieve this strong solution concept. In fact, for that case we cannot even show a standard ex-post Nash result. Instead, we settle for a very weak notion of solution, showing only that for every node there exists an "allowed strategy" that is optimal. Following Lavi and Nisan [70], this concept can be called *Set ex-post Nash*, and is defined thus:

A set profile $\mathbf{S} = (S_1, \ldots, S_k)$ (one set for every player) is *Set ex-post Nash equilibrium* if for every node i and every profile of fully specified strategies for the other nodes $s_1 \ldots s_{i-1}, s_{i+1} \ldots s_k$ (with $s_j \in S_j$ for all j), there exists $s_i^* \in S_i$ such that

$$\begin{aligned} & u_i(g_t(s_1(\theta_1), \ldots, s_i^*(\theta_i), \ldots, s_k(\theta_k))) \\ \geq \;& u_i(g_t(s_1(\theta_1), \ldots, s_i'(\theta_i), \ldots, s_k(\theta_k))), \end{aligned}$$

for every possible alternate strategy s_i' that i could have, every fair schedule t, and for all possible values of the private information $\theta = (\theta_1 \ldots \theta_k)$.

We emphasize that this solution concept only states that the "optimal" strategy s_i^* for node i *exists* in S_i, without specifying exactly how to find it. Furthermore, this condition does not necessarily yield a single (fully-specified) strategy profile $\mathbf{s}$ that is an ex-post Nash equilibrium, since the optimal strategy s_i^* for node i may change depending of the strategies of the other players.

A.2 Proofs: Useful Lemmas

Lemma A.2.1 (False path lemma)**.** *Consider an execution of the routing protocol where all the nodes in the AS graph except perhaps a single manipulator node m follow BGP-compliant strategies, and assume that this execution converges to a persistent outcome M. If any node $n \neq m$ announces a false path P in M (*i.e., *P differs from the data-plane path that n uses in M), then P must be of the form $P = nRmQd$ where nRm a true path and mQd is a false path.*

Proof. Denote the path that n announces by $n = a_r a_{r-1} \dots a_1 a_0 = d$. Let a_i be the closest node to n on this path that announces to a_{i+1} something other than $a_i a_{i-1} P$ where $a_{i-1}P$ is the announcement that a_i receives from a_{i-1}. Since this is not consistent with a BGP-compliant strategy, we conclude that necessarily $a_i = m$. Hence m must be on the path that n announces in this execution. Let i^* be the last occurrence of m on this path (namely $m = a_{i^*}$ and $m \neq a_j$ for $j > i^*$). Then for every $j > i^*$, a_j uses a BGP-compliant strategy so it follows that a_j announces to a_{j+1} the path $a_j a_{j-1} \dots a_0$, and moreover a_j uses a_{j-1} as its next-hop in the data-plane path in T. It follows that the data-plane path of n begins with $n = a_r a_{r-1} \dots a_{i^*} = m$. Thus, denoting $R = a_{r-1} \dots a_{i^*+1}$ and $Q = a_{i^*+1} \dots a_0$, we have that nRm is a true path, and since by assumption n announces a false path it follows that mQd must therefore be a false path. □

Next, we define a useful concept, called permitted path. Informally, a permitted path is a path that is not (ingress or egress) filtered by any node on that path.

Definition A.2.2 (Permitted paths)**.** Consider an AS graph where all nodes use BGP compliant strategies. We say that a path P is *permitted* if it is admitted at all the nodes in it, and moreover every node in it exports it to the next node.

Note that if all nodes use BGP compliant strategies then any data-plane path must also be a permitted path.

Our proofs rely heavily on the following lemma, due to Feigenbaum *et al.* [39].

Lemma A.2.3 ([39, Lemma 14.8])**.** *Consider an AS graph where all nodes use BGP-compliant strategies that obey consistent export, and where the ranking functions of all nodes are policy-consistent and contain no dispute wheels.*

Then there is a unique globally stable outcome T that the protocol must converge to, and moreover T is locally optimal at all nodes in terms of the ranking functions. Namely: for any permitted path nSd in the network, the node n is assigned in T a data-plane path nRd such that $r_n(nRd) \geq r_n(nSd)$.

For self-containment, we re-prove this lemma here.

Proof. Since the ranking contain no dispute wheel and all nodes use BGP compliant strategies, it follows from [53] that there exists a unique globally stable outcome T to which the protocol converges. It remains to show that T is locally optimal at all nodes.

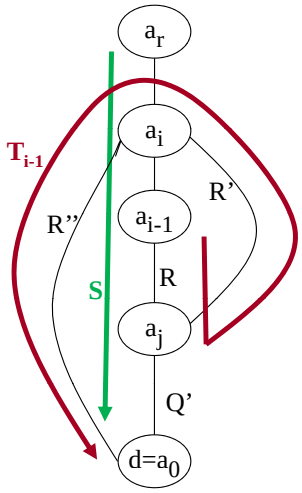


Figure A.1: Case 2 of the induction step in the proof of Lemma A.2.3.

Let $a_r \to a_{r-1} \to \ldots a_0 = d$ be any permitted path in the graph, and for every node a_i on this path we denote by S_i the sub-path $a_i \to \ldots a_0$. We will prove by induction over i, that each node a_i is assigned in T a path which is ranked at least as high as S_i.

Base case. The case $i = 0$ is trivially true, because the only path for $a_0 = d$ is the empty one.

Induction step. Assume that for all $j < i$ it holds that the path assigned to a_j in T (which we denote T_j) is ranked at least as high as S_j, namely $r_{a_j}(T_j) \geq r_{a_j}(S_j)$. (This implies in particular that a_j is assigned some path in T.) We now prove for a_i.

Note that a_{i-1} is willing to export S_{i-1} to a_i (since we said that S was permitted), and therefore it must also announce T_{i-1} to a_i because of consistent export. We have two cases: either the path T_{i-1} goes through a_i, or it does not.

1. If T_{i-1} does not go through a_i then from policy consistency and $r_{a_{i-i}}(T_{i-1}) \geq r_{a_{i-i}}(S_{i-1})$ we get that also
$$r_{a_i}(a_i T_{i-1}) \geq r_{a_i}(a_i S_{i-1}) = r_{a_i}(S_i)$$
Hence a_i has an available path that is ranked at least as high as S_i, and therefore must choose one such highly-ranked path in T.

2. Assume now that the path T_{i-1} does go through a_i. We depict this case in Figure A.1.

   Denote the longest common prefix of the paths T_{i-1} and S_{i-1} by $Ra_j = (a_{i-1} \ldots a_{j+1})a_j$ (note that R may be empty). Namely, we have $T_{i-1} = Ra_jQ$, $S_{i-1} = Ra_jQ'$, and the first nodes in Q, Q' differ. (In other words, the node a_j is the first node on the path S_{i-1} that uses a different next-hop in S_{i-1} and T_{i-1}.) Since T_{i-1} goes through a_i but S_{i-1} does not, it means that a_i must be somewhere on the sub-path Q, so we can re-write T_{i-1} as $T_{i-1} = Ra_jR'a_iR''d$, where $T_j = a_jR'a_iR''d$ is the path assigned to a_j in T (and $T_i = a_iR''d$ is assigned to a_i in T).

   By the induction hypothesis we have that $r_{a_j}(T_j) \geq r_{a_j}(S_j)$, but since a_j uses different next-hops in T_j, S_j then the inequality must be strict. It must therefore be the case that $r_{a_i}(T_i) \geq r_{a_i}(S_i)$, or else we have a (2-pivot) dispute-wheel between a_i and a_j: a_i prefers $S_i = a_i \ldots a_j \ldots a_1 d$ over $T_i = a_iR''d$, and a_j prefers $T_j = a_jR'a_iR''d$ over $S_j = a_j \ldots a_1 d$.

□

A.3 Proofs: Volume Attractions

We now prove Theorem 2.4.1.

THEOREM 2.4.1 *Consider an AS graph where the valuation functions contain no dispute wheels. Suppose that all nodes, except a single manipulator node m, use BGP-compliant strategies and set their ranking equal to their valuations ($r_n(\cdot) \equiv v_n(\cdot)$ for every node n). Suppose further that m has a traffic-volume attraction function, and that at least one of the following two conditions hold:*

a. The valuations function of all nodes are next-hop and the export functions of all the nodes but m obey all-or-nothing export; or

b. The valuations function of all nodes are policy consistent, the export functions of all the nodes but m obey consistent export, and the network has path verification.

Then there is a BGP compliant strategy for m that sets $r_m(\cdot) \equiv v_m(\cdot)$ and obeys all-or-nothing export (and therefore also consistent export), such that this strategy is optimal for m. In particular setting $r_m(\cdot) \equiv v_m(\cdot)$ and using export-all rule is one optimal strategy.

Proof. Consider an arbitrary strategy for m and denote by M any persistent outcome of the protocol (which need not be globally stable, see Section 2.3.1). We assume that $u_m(M) > -\infty$ (or else any BGP-compliant strategy for m will do).

Now consider a BGP compliant strategy for m where $r_m(\cdot) \equiv v_m(\cdot)$, and m exports-all on every edge on which it announces a simple path in M. The rest of m's export policy can be arbitrary, as long as it complies with consistent export. Clearly this strategy is BGP compliant and obeys consistent export, and moreover when m uses this strategy then the ranking functions of all nodes are policy-consistent and contain no dispute wheels (since they are set equal to the valuation functions). We can therefore apply Lemma A.2.3 to conclude that there is a unique globally stable outcome T, which is locally optimal at all nodes with respect to the ranking functions. We now prove that the utility of m in T is at least as high as in M. A crucial observation (that we prove in Lemma A.3.1 below), is that for every node n, the data-plane path of n in T has valuation at least as high as any control-plane announcement that n receives in M. We can now show that $u_m(T) \geq u_m(M)$.

- From the crucial observation Lemma A.3.1, we know that the valuation of m in T is at least as high as in M (since m routes in M on some path that was announced to it). Thus $v_m(M) \leq v_m(T)$.

- Next we show that every node routing through m in M must also route through it in T, and so $\alpha_m(M) \leq \alpha_m(T)$. To do this, fix some path $R = (n_r n_{r-1} \ldots n_0 = d)$ that *does not go through* m in T. We prove by induction on i that each of the nodes n_i use the same path also in M. The base case $n_0 = d$ this is trivial. For the induction step, assume now that every n_j with $j < i$ uses the same path in T and M. We prove this is also the case for n_i. Denote the path that n_{i-1} uses in T and M by R_{i-1}. Since $n_{i-1} \neq m$ then we know that n_{i-1} exports the path R_{i-1} to n_i also in M. From the crucial observation Lemma A.3.1, we also know that R_{i-1} is at least as good as any path which is announced to n_i in M (since n_i is in a persistent state). Further, R_{i-1} must be strictly better for n_i than any path that does not have next-hop n_{i-1}. Hence n_i will choose the path $n_{i-1}R_{i-1}d$ in M as well, and we have completed the induction step.

Thus, since $u_m(\cdot) = v_m(\cdot) + \alpha_m(\cdot)$, we have that $u_m(T) \geq u_m(M)$, and Theorem 2.4.1 follows. □

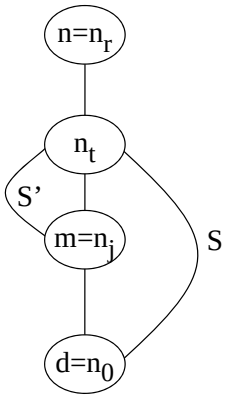


Figure A.2: The proof of Theorem 2.4.1

Lemma A.3.1 (Crucial Observation)**.** *Consider an AS graph where the valuation functions contain no dispute wheels, where one node m uses an arbitrary strategy and all other nodes use some BGP-compliant strategies with $r_n(\cdot) \equiv v_n(\cdot)$. Let M denote an outcome of the routing protocol in this network and assume that $u_m(M) > -\infty$ (M is a globally persistent outcome, but need not be globally stable).*

Consider further a BGP-compliant strategy for m where $r_m(\cdot) \equiv v_m(\cdot)$ and m exports-all on every edge on which it announces a simple path in M. The rest of m's export policy can be arbitrary, as long as it complies with consistent export. Let T denote the unique globally stable outcome of the protocol in this modified network.

Finally, assume that at least one of the following two conditions hold:

a. *The valuations function of all nodes are next-hop and the export functions of all the nodes but m obey all-or-nothing rule; or*

b. *The valuations function of all nodes are policy consistent, the export functions of all the nodes but m obey consistent export, and the network uses path verification.*

Then for every node n in the network, $v_n(T)$ is at least as high as the valuation of any path announcement that n receives in M.

Proof. Let R be a path announcement that a node n receives in M, and assume that $v_n(nR) > -\infty$ (otherwise there is nothing to prove). This means that nR is a simple path that reaches the destination, so we can denote it by $R = n_{r-1} \dots n_1 n_0$ with $n_0 = d$ (and we also denote $n = n_r$). In the rest of this proof, we show that there must exists a path nS which is permitted in the network where m uses the BGP-compliant strategy above, such that $v_n(nS) \geq v_n(nR)$. Then, if we apply Lemma A.2.3 to the permitted path nS, it follows that the path assigned to n in T has valuation at least as high as $v_n(nS) \geq v_n(nR)$ and Lemma A.3.1 follows.

First, notice that if the manipulator m is *not on* R then the path nR itself is permitted in the "BGP compliant network" and we are done. Now assume that $m = n_j$ for some $j \leq r-1$. Since we assumed that $u_m(M) > -\infty$ then m has some data-plane path to the destination in M, and we denote this path by mQ. Note that mQ is a data-plane path that includes only honest nodes, so it must be permitted in the "BGP compliant network". We now consider separately the two cases in the lemma statement.

Case a: next-hop policy and all-or-nothing export. There are two sub-cases: either mQ goes through n, or it does not.

- Suppose mQ does not go through n. Let t be the highest index ($j \leq t < r$) such that the path mQ goes through n_t, and denote the portion of mQ from n_t and on by n_tS. Thus S is a data-plane path that does not go through $n_r = n$ and does not go through $n_j = m$. (See Figure A.2.) Hence $n_r \dots n_tS$ is a simple path, and by next-hop policy it holds that $v_n(n_r \dots n_tS) = v_n(n_r \dots n_t \dots n_0) = v_n(n_rR)$. Thus we have proved that the path $n_r \dots n_tS$ is ranked at least as high as nR. It remains to prove that it is permitted. We have two sub-cases: either $m = n_t$ or not.

  $m = n_t$. In this case, we have $t = j$ and $Q = S$. Then all the nodes $n_{j+1} \dots n_{r-1}$ must be honest and since n_r receives the announcement $n_{r-1} \dots n_1n_0$ then m must have announced something to n_{j+1} in M. By construction, m must export all on this link in its BGP compliant strategy. Also the path mS is admitted at m (since m has ranking more than $-\infty$), and so $mS = nR$ is a permitted path as required.

  $m \neq n_t$. In this case m is not on the path $n_r \dots n_tS$. We prove by induction that each honest node n_i admits and exports the path $n_in_{i-1}...S$ in M.

  As a base case, n_t uses the data-plane path n_tS by construction, and thus n_tS must be permitted. Furthermore, since n_t exports a path to n_{t+1} in M, from all-or-nothing export we have that n_t is willing to export n_tS also in M. For the induction step, suppose that n_{i-1} admits and exports $n_{i-1}...n_tS$ to n_i. Since n_i uses next-hop policy, we have that $v_{n_{i+1}}(n_in_{i-1}...n_tS) = v_{n_{i+1}}(n_in_{i-1}...n_t...d)$. Since n_i exported a path to n_{i-1} in T, from all-or-nothing export we have that n_i is willing to export $n_in_{i-1}...n_tS$ also in M.

  Thus our induction has shown that the path $nn_{r-1}...n_tS$ in M is permitted (since all the nodes on that path admit it and are willing to export it), and moreover that $n_rn_{r-1}...n_tS$ is ranked at least as high as $n_rn_{r-1}...n_1d = nR$ as required.

- Suppose mQ does go through n. Then denote mQ as $mS'nS$. Now nS is permitted since it is a data-plane path, and nS must have higher ranking than nR since (because n is in a persistent state) n received the announcement R but is routing in the data-plane over nS.

This concludes the proof for the setting of next-hop policy and all-or-nothing export.

Case b: policy-consistency and path verification. Due to path verification, we know that the path R is admitted and exported by all the "honest nodes" $n_i \neq m$ and therefore these nodes admit it and export it also in T. Also, by the way that we defined the ranking and export functions of m we know that IF $v_m(mn_{j-1} \dots n_0) > -\infty$ then also m will admit and export this path in T (and again we have that nR is permitted).

It is left to consider the case that $v_m(mn_{j-1} \dots n_0) = -\infty$, namely the case where m announces in M a path that is not admitted by its valuation function. Again, let t be the highest index ($j \leq t \leq r$) such that the data-plane path mQ that m uses in M goes through n_t, and denote the portion of mQ from n_t and on by n_tS (so S does not go through $n_j = m$). (See Figure A.2.) We now show that the valuation $v_{n_t}(n_tS)$ must be at least as high as $v_{n_t}(n_tn_{t-1} \dots n_0)$.

- If $n_t = n_j = m$ (so mQ and n_tS is the same path) then this follows from the fact that $v_m(mQ) > -\infty = v_m(mn_{j-1} \dots n_1d)$.

- If $m \neq n_t$ then we re-write the path mQ as $mS'n_tS$, and notice that we must have $v_{n_t}(n_tS) \geq v_{n_t}(n_t \dots mn_{j-1} \dots n_0)$, or else we have a dispute wheel between n_t and m (since $v_m(mS'n_tS) > v_m(mn_{j-1} \dots n_0) = -\infty$).

Now consider the path $n_rn_{r-1} \dots n_tS$. This is a simple path, and we just showed that $v_{n_t}(n_tS) \geq v_{n_t}(n_tn_{t-1} \dots n_0)$. From policy consistency it follows that also for each n_i, $t+1 \leq i \leq r$, the path

$n_i \dots n_t S$ has ranking at least as high as $n_i n_{i-1} \dots n_1$ (and therefore also valuation at least as high), since each n_i exports the path $n_i n_{i-1} \dots n_1$ to n_{i+1} in T it follows from consistent export that n_i exports $n_i \dots n_t S$ in M. Hence $n_r n_{r-1} \dots n_t S$ is a permitted path with valuation in n at least as high as nR, as needed. This concludes the proof for the setting of policy consistency and path verification. □ □

A.4 Proofs: Generic Attractions

THEOREM 2.5.1 *Consider an AS graph where the valuation functions are next-hop and contain no dispute wheel. Suppose that all nodes, except a single manipulator node m, use BGP-compliant strategies where they set their ranking equal to their valuations ($r_n(\cdot) \equiv v_n(\cdot)$ for every node n), and obey all-or-nothing export. Suppose further that the network uses either loop verification or path verification. Then there exists a BGP compliant strategy for m that uses $r_m(\cdot) \equiv v_m(\cdot)$ and obeys all-or-nothing export, which obtains the best possible* globally stable outcome *in terms of the utility function of m.*

Proof. Let M be a globally stable outcome that is obtained by an arbitrary (possibly cheating) strategy for m. We again assume that $u_m(M) > -\infty$, or else there is nothing to prove. In particular this implies that m has a data-plane path to d in M. Also, by the discussion in Section 2.2.3 we can assume without loss of generality that m has a single outgoing link in M.

Consider a BGP compliant strategy for m where $r_m \equiv v_m$ and m exports-all on every edge on which it announces a simple path in M, and exports nothing on every other edge. Clearly this strategy is BGP compliant and obeys all-or-nothing export, and moreover when m uses this strategy then the ranking functions of all nodes are next-hop (and therefore also policy-consistent) and contain no dispute wheel (since they are set equal to the valuations). This is exactly the setting of Case b of the crucial observation Lemma A.3.1, so we know that there is a unique globally stable outcome T such that for every node n in the network, the path assignment of n in T has valuation at least as high as any path-announcement that n receives in M. In particular, it follows that $v_m(T) \geq v_m(M)$ (because m routes in M on some path that was announced to it). Since $u_m(\cdot) = v_m(\cdot) + \alpha_m(\cdot)$, it only remains to show that $\alpha_m(T) \geq \alpha_m(M)$.

Assume to the contrary that we have $\alpha_m(T) < \alpha_m(M)$. We prove a sequence of statements that imply that some other node b must have raised an alarm, because it receives a path announcement of the form QbR where b did not announce the path R, and where m is on path Q. This contradicts either path verification (since b receive an announcement containing a path through b that b did not announce) or loop verification (where the utility of m is set to $-\infty$ when such an alarm is raised).

Claim A.4.1. *There is a node c that (1) routes through m in M, (2) uses a different outgoing edge in M than in T, (3) every node that routes through c in M uses the same outgoing link in T and M.*

Proof. We assumed towards contradiction that m gained an attraction in M, $\alpha_m(M) > \alpha_m(T)$, which implies that the subtree of m in M cannot be contained in the subtree of m in T, namely $M(m) \not\subseteq T(m)$. Hence, there exists some node that routes through m in M and uses a different next hop in M than in T.

Denoting $m = c_0$, we continue to find nodes $c_i (i \geq 1)$ as follows: For each node c_i, if there are nodes that route through c_i in M and use a different next-hop in M than in T, then we let c_{i+1} be one such node. We repeat this process until we reach a "last node" c such that every node that routes through c in M uses the same next-hop in T and in M.

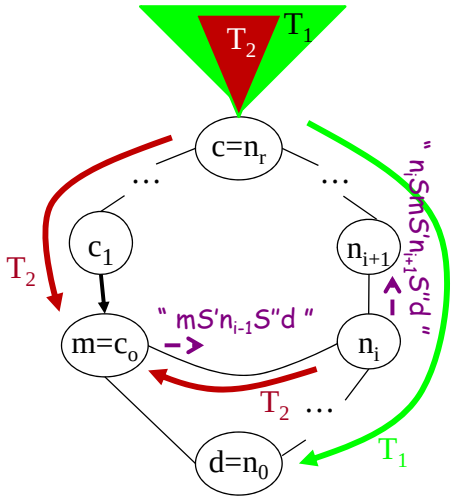


Figure A.3: The proof of Theorem 2.5.1

Observe that we must reach such "last node" since otherwise we will eventually repeat a node, say node c_r. But since each c_i routes through c_{i-1} then repeating a node means that we have a routing loop in M, and since all these nodes route through m and all of them (including m) have just one outgoing link, it follows that m is part of this routing loop, so in particular m does not have a path to the destination in M and $u_m(M) = -\infty$.

It follows by definition that this "last node" c satisfies items (1) through (3) in the claim assertion. □

Claim A.4.2. *Node c has a data-plane path to d in T.*

Proof. We again use the crucial observation Lemma A.3.1 to establish that the path assignment of c in T is ranked at least as high as any announcement that node received in M. In particular c is routing through m so it must have received an announcement with rank higher than $-\infty$ in M, so it must have a path with rank higher than $-\infty$ also in T. □

Denote the data-plane path of c to d in T by $n_r \dots n_1 n_0$ (with $c = n_r, d = n_0$), and we distinguish two cases: either n_{r-1} has a data-plane path to d also in M or it does not.

Case 1: n_{r-1} has a data-plane path to d in M. Observe that n_{r-1} does not route through $n_r = c$ in M, since it does not route though c in T, and we chose c such that $M(c) \subseteq T(c)$ (*i.e.,* every node that routes through it in M uses the same next-hop in T as in M).

Next we claim that n_{r-1} announces some simple path to n_r in M. Observe that n_{r-1} exports some path to n_r in T. If $n_{r-1} = m$, then by construction it only exports paths in T on edges on which it announces some simple path in M, so we know that it must have announced some simple path to n_r in M. On the other hand, if $n_{r-1} \neq m$ then it uses all-or-nothing export rule, and since we assume that it has a path in M and we know that it exports a path in T, it follows that it must export some path also in M (which must be simple since only simple paths are announced by BGP-compliant strategies).

Let $n_{r-1}Rd$ be the path that n_{r-1} announces to $n_r = c$ in M. Next, we claim that the path $n_r n_{r-1} Rd$ contains a loop. Suppose it did not. Then by next-hop ranking we would get that $r_{n_r}(n_r n_{r-1} Rd) = r_{n_r}(n_r n_{r-1} \dots n_0)$. But we know that the path $n_r n_{r-1} \dots n_0$ is the T path of $n_r = c$, so from the crucial observation Lemma A.3.1 we know that $n_r n_{r-1} Rd$ must be ranked at least as high as any announcement that c received in M. By construction c *uses a different*

next-hop than n_{r-1} in M, and thus it follows that the path the that c uses in M is ranked (strictly) lower than the path $n_r n_{r-1} Rd$. Now, since we assume that $c = n_r$ is stable in M, it follows that $c = n_r$ would have chosen to route through n_{r-1} also in M. This contradicts the fact that c indeed chose a different next-hop than n_{r-1} in M, and hence we conclude that the path $n_r n_{r-1} Rd$ contains a loop.

However, we argued above that the path $n_{r-1}Rd$ is simple. Thus, only way that $n_r n_{r-1} Rd$ could contain a loop is if $c = n_r$ itself appears somewhere on the path $n_{r-1}Rd$. But we argued above that n_{r-1} does not route through $c = n_r$ in T, so the path $n_{r-1}Rd$ is a false path. By the false-path lemma (Lemma A.2.1) it follows that this announced path has the form $n_{r-1} S m S' n_r S'' d$ (since from the false path lemma S is a true path and $mS'n_rS''d$ is a false path, and $c = n_r$ must appear on the false path).

Next, observe that the S'' portion of the announced path cannot include m (since m appears before $c = n_r$ and $n_{r-1} S m S' n_r S'' d$ is a simple path). But $c = n_r$ routes through m in M, and so invoking the false path lemma again implies that c must have announced some path that goes through m. It follows that $c = n_r$ did not announce the path $n_r S'' d$, and so upon obtaining the announced path $mS'n_rS''d$ from n_{r-1}, $c = n_r$ would detect a false loop and raises an alarm.

Case 2: n_{r-1} has no path to d in M. Here we denote by n_i the node closest to $c = n_r$ on the T path (but not c itself) that does have a data-plane path to d also in M. We know that such n_i exists, since in particular d has the empty path to d in M. By definition of n_i, we have that n_{i+1} *does not have any data-plane path* to the destination in M. This implies (1) that $n_{i+1} \neq m$ (since m has a path to d in M), (2) that n_{i+1} does not use the same next-hop in M as it does in T, and (3) that n_i does not route through n_{i+1} in M.

Again, we argue that n_i must announce a simple path to n_{i+1} in M, since it announces some path to n_{i+1} in T. The argument is the same as in the previous case: either $n_i = m$ where this follows by construction, or $n_i \neq m$ where it follows from the all-or-nothing export and the fact that n_i has a data-plane path in M.

Also, we denote the path that n_i announces to n_{i+1} by $n_i Rd$, and again argue that although this is a simple path, the path $n_{i+1} n_i Rd$ must include a loop, or else n_{i+1} would have chosen it in M rather than having no data-plane path at all. (This follows because any path with next-hop n_i must be admitted at n_{i-1} due to next-hop policy, and from the assumption that n_{i+1} is stable in M.)

As in the previous case, we conclude that the announcement $n_i Rd$ must include n_{i+1}. However, we argued above that n_i does not route through n_{i+1} in the data plane. Thus, we have that $n_i Rd$ is a false path, and so combining this observation with the false-path lemma Lemma A.2.1 tells us that it is of the form $n_i S m S' n_{i-1} S'' d$. But n_{i-1} did not announce the path $n_{i-1} S'' d$ (since it has no data-plane path in M, and so it does not announce anything in M). Hence, n_{i+1} must raise an alarm upon receiving the announcement $n_i Rd$ from n_i. □ □

A.5 Proofs: Gao-Rexford Networks

Before we start, we need the following useful concept:

Transitive customers. A node b is a *strict transitive customer* of node c if b is connected to c via a path consisting of only customer-provider links as in the right half of Figure A.4. We also restate here a simple, useful lemma of the Gao-Rexford conditions proved by Gao, Griffin and Rexford in [41].

Lemma A.5.1 (Transitive customers [41, Theorem VII.4])**.** *If either the path $P = abRc$ or the path $P' = cR'ba$ is permitted, and if node a is not a customer of node b, then node c is a strict transitive customer of node b over the permitted path.*

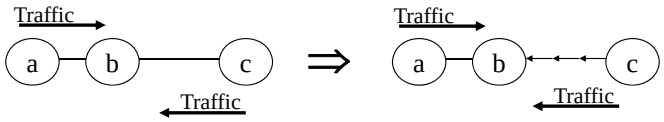


Figure A.4: Lemma A.5.1.

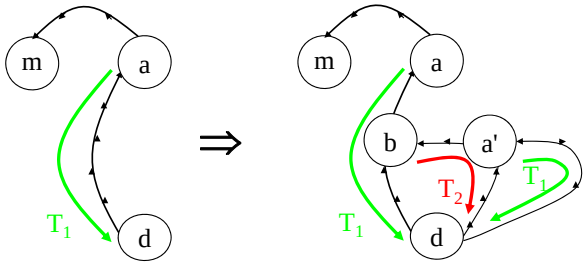


Figure A.5: Proof of Lemma A.5.2

We remark that even if not all the nodes in the AS graph use BGP-compliant strategies, Lemma A.5.1 still holds as long as all the nodes on the permitted path (except perhaps the last one, closest to the destination) use BGP-compliant strategies that obey the Gao-Rexford conditions.

We now prove the following helper lemma that we use to derive a contradiction in Theorem 2.6.1:

Lemma A.5.2. *Consider an AS graph (that obeys GR1) where all nodes, except perhaps a single manipulator node m, use BGP-compliant strategies that obey the Gao-Rexford conditions (*i.e., *rankings obey GR3, export obeys GR2) Let T be the unique globally stable outcome when m follows some BGP-compliant strategy that obeys the Gao-Rexford conditions, and let M be a globally stable outcome that results from some other arbitrary strategy of m.*

If there is a node a in the network such that (1) a is a strict transitive customer of the manipulator m, (2) a uses a different path in M than in T, and (3) the destination d is a strict transitive customer of a along a's path in T. Then there is a different node $a' \neq a$ which is a strict transitive customer of a, such that a' also satisfies the conditions (1)-(3).

Proof. Since a is a strict transitive customer of m, and the destination d is a strict transitive customer of a on a's T path, then the Topology condition GR1 implies that m cannot be on the path of a in T. Denote by b the node closest to the destination along a_i's path in T that uses a different path in M than in T (we know that such a b exists since in particular node a is such a node), and denote the paths of b in T and M by bQ_1d and bQ_2d, respectively.

Since all the nodes on the path Q_1d are honest and they all use that path in M, it follows that b must have received an announcement Q_1d from the first hop on that path in M, (and since M is a persistent outcome) and yet it chose a different path in M. We conclude that b's ranking has $r_b(M) > r_b(T)$. And since b's next hop in T is a customer, the Preferences condition GR3 implies that b's next hop in M must also be a customer. Applying Lemma A.5.1 we get that (a) node m cannot be on the path bQ_2d, or else it would have to be a strict transitive customer of b and we would have a customer-provider loop; and (b) since m is not on bQ_2d then the destination is a strict transitive customer of b along this path.

Let node a' be the node closest to the destination along the path bQ_2d that uses a different path in M than in T (again, we know it exists since b is one such node). Denote the paths of a' in T and M by $a'R_1d$ and $a'R_2d$, respectively. It follows that the path R_2d is also in the path assignment T. Notice that a' is also a strict transitive customer of the manipulator m, and that destination d is a strict transitive customer of a' along the path R_2d. Since all the nodes on the path R_2d uses this path also in T, and since a' received an announcement for this path in M (because it uses this path in M) then a' must have received an announcement R_2d in T also (since T is a globally stable outcome). Yet a' chose a different path in T. We conclude that the ranking of a' has $r_{a'}(T) > r_{a'}(M)$, which also implies that $a' \neq b$.

Since $r_{a'}(T) > r_{a'}(M)$ and since the next hop after a' on the path $a'R_2d$ in M is a customer of a', the Preferences condition GR3 implies that the next hop after a' on the path $a'R_1d$ in T must also be a customer. Then, we can apply Lemma A.5.1 to find that the destination is a strict transitive customer of a' along the path $a'R_1d$ in T.

We established that a' satisfies the conditions (1)-(3), and we also know that b is a transitive customer of a (or a itself), a' is a strict transitive customer of b, and $a' \neq b$. It follows that $a' \neq a$, since otherwise we would have a customer-provider loop in the graph. □

We are now ready to prove the main result of this section.

THEOREM 2.6.1 *Consider an AS graph where the valuations are policy consistent and contain no dispute wheels, and the valuations and attraction functions of all nodes obey the Gao-Rexford conditions and AT4, and all attractees use next-hop policy with their providers and peers. Suppose that all nodes, except a single manipulator node m, use BGP-compliant strategies that obey consistent export and GR2 export, and moreover set their ranking equal to their valuations ($r_n(\cdot) \equiv v_n(\cdot)$ for every node n). Suppose further that the network has path or loop verification.*

Then there exists a BGP compliant strategy for m that uses $r_m(\cdot) \equiv v_m(\cdot)$ and obeys GR2 and consistent export, which obtains the best possible globally stable *outcome in terms of the utility function of m. In particular, setting $r_m(\cdot) \equiv v_m(\cdot)$ and exporting all paths to customers and no paths to providers and peers is one optimal strategy.*

Proof. Let M be a globally stable outcome that results from some arbitrary strategy for m. We assume M that $u_m(M) > -\infty$ (or else any BGP compliant strategy for m will do).

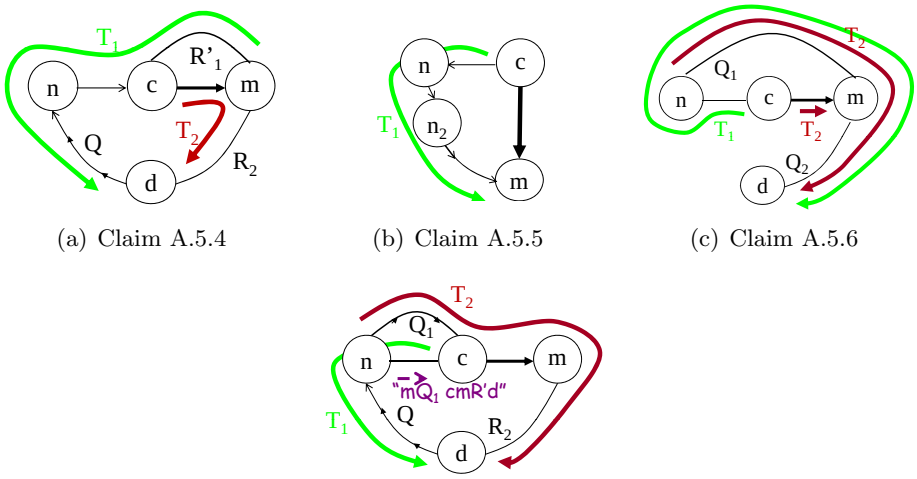
Now fix a BGP compliant strategy for m where $r_m \equiv v_m$, and where m (i) exports all paths to every *customer* that routes through it in M and (ii) exports no paths to nodes that are not its customers. (Note that this export rule obeys GR2.) The rest of m's export policy can be arbitrary, as long as it complies with consistent export and with GR2.

Clearly this strategy is BGP compliant, and when m uses this strategy then the ranking functions of all nodes contain no dispute wheels (since they are set equal to the valuation functions). The results of Griffin *et al.* [53] imply that the protocol converges to a unique globally stable outcome T. We prove next that the utility of m in T is at least as high as in M.

Our proof is by contradiction. We assume that $u_m(M) > u_m(T)$, and prove a sequence of claims that together imply that the conditions of Lemma A.5.2 must hold in this graph. We then repeatedly apply Lemma A.5.2 to show that the graph contains a customer-provider cycle, and thus violates the Topology condition GR1.

Denote the data-plane paths of m to the destination in T and M by mR_1 and mR_2, respectively.

Claim A.5.3. *The is a node c that is an attractee of m that routes directly through m in M but not in T.*



(a) Claim A.5.4 (b) Claim A.5.5 (c) Claim A.5.6

(d) Claim A.5.8

Figure A.6: Pictorial representation of the proof of Theorem 2.6.1

Proof. Since the data plane path R_2 used by m in M is permitted at all nodes on R_2, and since all these nodes are honest (otherwise mR_2 would not be a simple path, and $u_m(M) = -\infty$) know that mR_2 is permitted also in T. Note that T satisfies all the conditions of Lemma A.2.3, since all nodes use consistent export and set their ranking equal to their valuations (so the rankings have no dispute wheel and are policy consistent). So we know that T is locally optimal everywhere. In particular, since the data-plane path of m in M is permitted also in T (since it only goes through honest nodes) then $v_m(T) \geq v_m(M)$. But we assumed that $u_m(M) > u_m(T)$, so we must have $\alpha_m(M) > \alpha_m(T)$, which means that m gained AT4 attraction in M that it did not have in T. □

Claim A.5.4. *Node c has a data-plane path to the destination in T, and moreover $r_c(T) > r_c(M)$.*

(Note that this claim *does not follow* from Lemma A.2.3, since there could be paths that are "permitted" in M but not in T: recall that m's export policy in T dictates that it does not announce anything to its providers and peers, whereas it is possible that m did announce something to them in M.)

Proof. Assume toward contradiction that $r_c(T) \leq r_c(M)$. Since c was defined as a node that uses m as next-hop in M but not in T, then the inequality has to be strict. Since c is an attractee of m (and therefore its customer), then c must use next-hop policy with m. Since c is a customer that routes through m in M, then the export policy of m in T includes exporting all to c. Since m is honest in T, we know that m announces to c the path mR_1 that it uses in T.

If mR_1 was a simple path, then from next-hop policy we have that $r_c(cmR_1) = r_c(cmR_2) > r_c(T)$, which contradicts the fact that c is stable in T (it should have chosen the better available path cmR_1). So we know that mR_1 must have a loop in it, but mR_1 is a simple *path* (being the data-plane path of m), so it must be that c appears on that *path* (which in particular implies that c has a data-plane path in T). We can re-write the path that m takes in T as $R_1 = R'_1cnQ$, as depicted in Figure A.6(a).

Since c is a customer of m, it follows from the Topology condition GR1 that m cannot be a strict transitive customer of c along the path mR'_1c. Hence there are adjacent nodes

between m and c on the path R'_1 (call them a, b) such that a is not a customer of b. Since the path mR'_1cnQd is permitted (because it is the data plane path in T) and since all nodes behave honestly in T, we can apply Lemma A.5.1 to conclude that d is a transitive customer of b along this path. In particular it means that n is a customer of c. (Notice that this is true even if $n = d$.) But this violates the Preferences condition GR3, since we assumed that $r_c(M) = r_c(cmR_2) \geq r_c(cnQd) = r_c(T)$ where m is a provider of c and n is its customer. □

From now on, let us denote the path of c to the destination in T by $n_0n_1 \dots n_t$ (where $c = n_0$ and $d = n_t$), and remember that c uses m as a next-hop in M but not in T, so $n_1 \neq m$.

From Claim A.5.4 we can also conclude that $n_1 \neq d$: Otherwise ($d = n \neq m$), the T-path dc would be available to c also in M, and so c would take it (since we just proved that the T path is ranked higher than then M path of c) and this would contradict the stability of c in outcome M. Next we prove that m is not on the T-path of c.

Claim A.5.5. *c does not route through m in T.*

Proof. For the sake of contradiction, suppose that m is on the T-path of c, namely $m = n_j$ for some $1 \leq j \leq t$. This means in particular that $m = n_j$ exports some path to n_{j-1} in T, so n_{j-1} is a customer of m. (Recall that m only export paths in T to its customers.) Applying Lemma A.5.1 we find that c is a strict transitive customer of m along c's path in T. In particular, $c = n_0$ is a customer of n_1 and n_1 is a customer of n_2. Now since the valuations of n_1 obey GR3, we deduce that $v_{n_1}(n_1n_2 \dots d) < v_{n_1}(n_1c \dots d)$. However, from Claim A.5.4 and the fact that c uses next hop policy with all its providers, we have $v_c(cn_1 \dots d) \geq v_c(cm \dots d)$. Furthermore, the inequality is strict, since $m \neq n_1$. Hence there is a (2-pivot) dispute wheel between c and n and we have arrived at a contradiction. □

Claim A.5.6. *The node n_1 uses a different (data-plane) path for its traffic in M than in T.*

Proof. Assume toward contradiction that n_1 uses the T-path $n_1n_2 \dots n_t = d$ also in M. Below we also denote this path by n_1Q. From Claim A.5.4 we know that $r_c(cmR_2) < r_c(cn_1Q)$, so we know that n_1 does not announce n_1Q to $c = n_0$ in M (or else c would have used this path). But we know that n_1 exports the path n_1Q to c in T, and that n_1 is honest, so it would have exported this path to c in M if it had chosen it. We deduce that n_1 had chosen a different path in the control plane in M (even though it actually routes on n_1Q in the data plane). In other words, n had chosen a false path in M. From the false path lemma (Lemma A.2.1), we have that both the false-path in the control plane and the data-plane path must include m. But this is a contradiction, since we assume that n uses the same data-plane path in both M and T, and from Claim A.5.5 we know that m is not on the data-plane path of n_1 in T. □

Claim A.5.7. *Node n_1 announces a path to $c = n_0$ in M.*

Proof. For every node n_i on the T-path $n_1 \dots n_{t-1}n_t$, we denote the control-plane path that n_i chooses in M (if any) by n_iQ_i. We now show by backward induction over $i = t \dots 2$ that (i) node n_i ranks n_iQ_i at least as high as $n_in_{i+1} \dots n_t$, and (ii) n_i announces the path n_iQ_i to n_{i-1}. For the proof below, recall that $n_i \neq m$ for all i (due to Claim A.5.5), so all the n_i's use policy-consistent ranking and consistent export also in M.

The base case $n_t = d$ is obvious. For the induction case, assume that the two conditions above hold for n_{i+1} and we prove for n_i. We have two cases: either $n_{i+1}Q_{i+1}$ goes through n_i or it does not.

- If $n_{i+1}Q_{i+1}$ does not go through n_i, then from policy consistency (and since n_{i+1} prefers this path to $n_{i+1}\dots n_t$) we have that also n_i must prefer $n_in_{i+1}Q_{i+1}$ over $n_in_{i+1}\dots n_t$. Moreover, since the path $n_in_{i+1}Q_{i+1}$ is available to n_i in M (as we assume that n_{i+1} announces it), and since M is a globally stable outcome, then n_i must choose a control-plane path in M that is ranked at least as high. We conclude that $r_{n_i}(n_iQ_i) \geq r_{n_i}(n_in_{i+1}Q_{i+1}) \geq r_{n_i}(n_in_{i+1}\dots n_t)$.

- Suppose that $n_{i+1}Q_{i+1}$ does go through n_i. Then rewrite this path as $n_{i+1}Q_{i+1} = n_{i+1}R_{i+1}n_iQ'_i$. By the induction hypothesis, n_{i+1} announces this path to n_i, and also prefers it over $n_{i+1}\dots n_t$. Since n_i is honest and the network uses loop verification, it must be the case that n_i actually announces the path $n_iQ'_i$ (or else n_i would have raised an alarm, which would have set the utility of m in this outcome to $-\infty$). Hence n_i must have chosen $n_iQ'_i$ in the control plane in M, in other words we have $Q'_i = Q_i$.

  We claim that n_i must prefer n_iQ_i over $n_in_{i+1}\dots n_t$; otherwise we would have a dispute wheel between n_i and n_{i+1}, since n_{i+1} prefers $n_{i+1}R_{i+1}n_iQ_i$ over $n_{i+1}\dots n_t$.

In either case, we know that n_i prefers n_iQ_i over $n_in_{i+1}\dots n_t$. Since n_i uses consistent export, and since it announces $n_in_{i+1}\dots n_t$ to n_{i-1} in T, then it has to announce also n_iQ_i to n_{i-1} in M. □

Claim A.5.8. *The node n_1 is a strict transitive customer of m, and the destination d is a strict transitive customer of n_1 over the data-plane path of n_1 in T.*

Proof. Recall that we denote the data-plane path of n_1 in T by n_1Q. If n_1 is a direct customer of c then the first part of the lemma follows trivially (since c is a customer of m), and the second part follows by applying Lemma A.5.1 to the permitted path cn_1Q in T.

If n_1 is not a customer of c, then c must use next hop policy with n_1. From Claim A.5.7, we know that n_1 announces a path to c in M. Let n_1Q' be that path that n_1 announces to c in the manipulated outcome M. If the path n_1Q' does not go through c, then we have

$$r_c(cn_1Q') = r_c(cn_1Q) > r_c(cmR_2)$$

where the equality follows from next-hop policy and the inequality is from Claim A.5.4. But this is impossible, since if this was the case then c would have chosen n_1 as its next-hop also in M. Thus, the path n_1Q' must go through c.

Next denote by cmR' the control-plane path that c chooses in M. By loop-verification, it must be the case that cmR' is a suffix of n_1Q' (or else c would have raised an alarm and the utility of m would be set to $-\infty$). So re-write n_1Q' as $n_1Q'_1cmR'$. The path Q'_1 does not include m, or else n_1 wouldn't have chosen this path since it would contain a routing loop through m. Hence the partial path $n_1Q'_1cm$ must be the data-plane path that is used in M (and in particular it must be a permitted path). Since c is a customer of m, then we can apply Lemma A.5.1 to conclude that n_1 is a strict transitive customer of c (and therefore also of m).

Moreover, since n_1 is a strict transitive customer of c then the Topology condition GR1 says that it cannot be a provider of c. We assumed that n_1 is also not a customer of c, so they must be peers. We can now apply Lemma A.5.1 to the permitted T path cn_1Q, to conclude that the destination d is a strict transitive customer of n_1 over this path. □

Claims A.5.6 and A.5.8 established the existence of a node $a_0 = n_1$ which is (1) a strict transitive customer of the manipulator m, and where (2) a_0 uses a different path in M than in T, and (3) the destination d is a strict transitive customer of a_0 along its data-plane path in

T. Lemma A.5.2 asserts that there must be another node $a_1 \neq a_0$ which is a strict transitive customer of a_0, where a_1 also satisfies the conditions (1)-(3). Repeated applications of this lemma thus give us a sequence of nodes $a_1, a_2, \ldots$ such that for all i $a_i \neq a_{i-1}$ and a_i is a strict transitive customer of a_{i-1} (and they all satisfy the same conditions). Since there are a finite number of nodes in the AS graph, eventually one of the nodes in the sequence will repeat, resulting in a customer-provider cycle and violating the Topology condition GR1.

We see that our assumption that $u_m(M) > u_m(T)$ leads to a contradiction, thus concluding the proof of Theorem 2.6.1. □ □

Appendix B

Failure Detection

B.1 Fast cryptographic hashing

We now consider efficient pseudorandom functions (PRFs) for packet hashing. As we discussed in Section 3.3, the PQM adversary is presented with an online problem: to break security, Eve must break the secret key within a small time interval until the key is refreshed, based on the limited number of examples she sees during that interval. (Indeed, in some of our protocols we voluntarily send the key in plaintext once the interval is over, see Section 3.4.2).

For general hashing of variable-length packets (lengths upto, say 1500B) with a PRF, we suggest a construction based on ε_h-almost universal hash functions, as discussed in *e.g.,* [104] [68, Sec. 2.8.3] [18, 78]. Namely, for an interval key $k_u = (\kappa_1, \kappa_2)$ set

$$h_{k_u}(x) = E_{\kappa_1}(g_{\kappa_2}(x)) \tag{B.1}$$

where E is a block cipher taking n-bit inputs to n bit outputs (*e.g.,* AES), and g is an ε_g-almost universal hash function producing n-bit outputs (as defined in equation (3.1) of Section 3.3). See *e.g.,* [18] for a nice survey of various ε_h-almost universal hash functions that can be used with this construction. [1] For long packets (length $\gg n$-bits), the performance of this hash function is dominated by the performance of the universal hash function g, which can extremely fast. For shorter packets (of lengths $\approx n$-bits), performance is limited by the block cipher; fortunately, this construction amounts to using a single invocation of the block cipher (*c.f.,* with traditional PRFs like HMAC [67] that require ℓ/n invocations of the block-cipher for packets of length ℓ-bits.) Performance can be further improved by the replacing full-fledged block-cipher like AES with a weaker block cipher such as DES or with a small number of rounds of AES [29]; it would still require enormous resources to break the security of the PRF within the time limit (100ms) imposed in our online setting.

Unfortunately, most of the literature focuses on the construction of fast universal-hash based *MACs*, rather than PRFs. Thus, below, we shall show that the PRF construction presented in (B.1) is a secure PRF as long as the ε_g universal hash function g used in the construction has

$$\varepsilon_g \leq 2^{-(2\log_2 q + k)} \tag{B.2}$$

and the PRF is used no more than q times before the interval key is refreshed. In our setting, $q = O(T)$, where T is the number of packets per interval[2], and k is the security parameter. In

[1] The original universal-hash-based MACs require an an extra nonce r which must be unique for each invocation, and define $h_k(x, r) = E_{\kappa_1}(r) + g_{\kappa_2}(m)$ or $h_k(x, r) = E_{\kappa_1}(r) \oplus g_{\kappa_2}(m)$. Because, in our application, we need only hash a small number of packets $\approx T = 10^7$ before changing the interval key, this nonce is not required.

our online setting, k need not be very large. As an example, putting $q = 10T$ and $T = 10^7$ and requiring $k > 32$, it follows that we require $\varepsilon_g \leq 2^{-86}$. Suppose now we use GHASH [78] with block lengths n as the universal hash function used in the construction. Then for packets of length upto 1500B bytes, GHASH has $\varepsilon_g = 1500/m2^{-n}$ where m is the block length and n is the length of the output. Thus, we find it suffices to run GHASH with output length $n = 96$ and block lengths of $m = 128$ bits. (A general rule of thumb is that hashing is faster with smaller block lengths. Note also that in Section 3.5.4, we show that for the 'secure sketch' protocol, it actually suffices to use GHASH with block lengths of about $m = 64$ bits!)

We now obtain (B.2). We say that a function h is (q, ε)-secure PRF if an algorithm, given an oracle for a function F, has advantage at most ε in distinguishing if F is either (1) h keyed with some randomly chosen secret key k_u, or (2) a truly random function with the appropriate domain and range [50]. To show that the construction in (B.1) is a secure PRF, we prove the following theorem:

Theorem B.1.1. *The function $h_\kappa(x) = f(g_\kappa(x))$ is a $(q, q^2\varepsilon_g)$-secure PRF if f is a* truly random function *and g is an ε_g-almost universal hash function keyed with randomly-chosen key κ.*

Then the security of the construction follows if we assume that the block cipher E produces (pseudorandom) outputs that are indistinguishable from the outputs of a truly random function. Thus, we require $q^2\varepsilon_g \leq 2^{-k}$, where k is the security parameter for the PRF, and (B.2) follows.

Proof of Theorem B.1.1. The algorithm makes q queries $x_1, ..., x_q$ to the oracle for F; without loss of generality, assume these are all distinct. (If the adversary repeats a query, he can compute the answer without consulting the oracle.) Let E_{distinct} be the event that κ was chosen such that that $g_\kappa(x_1), ..., g_\kappa(x_q)$ are all distinct. Since f is a truly random function taking in distinct inputs, it follows that if E_{distinct} is true, the adversary has no advantage in distinguishing between h and a truly random function. Thus, the adversary's distinguishing advantage is at most $\Pr[\neg E_{\text{distinct}}]$. Now, for every distinct $x_1, ..., x_q$, we have $\Pr[\neg E_{\text{distinct}}] = \binom{q}{2}\varepsilon_g \leq q^2\varepsilon_g$ where ε_g is the collision probability of g (taken over the choice of κ) as in equation (3.1). The theorem follows. □

B.2 Interval synchronization

B.2.1 Symmetric-key protocols

In our secure sketch (Section 3.5) and symmetric secure sampling protocols (Section 3.4.1), we assume that Alice and Bob agree on the set of packets belonging to a particular interval, and process these packets using the same interval key k_u (and, in Section 3.5, map the same set of packets to the same sketch.) For these protocols, the best way to achieve this is to have Alice send Bob a special 'Interval End' message each time she ends an interval and begins a new one. The 'Interval End' message should contain the interval number u, and be authenticated with a MAC keyed with (some portion of) the master secret key. When Bob receive this packet, he knows he should derive a fresh interval key (and, in Section 3.5, a fresh sketch). This approach

[2] In the secure sampling protocol, q, the number of queries made to the packet-hashing PRF is the sum of the number of packets that Alice sends and the number of packets that Bob receives. Notice that an adversary can exceed the bound of $q = O(T)$ by adding many packets to path, and potentially use the information it sees (*i.e.,* the ACKs) to learn how to break the PRF. To avoid such problems, we suggest that Bob counts the number of packets he receives in an interval, and stops acknowledging them once more than q packets have been received.

does not make any synchronization assumptions about Alice and Bob's local clocks; it also works even if the path between Alice and Bob is subject to variable latency.

The effect of packet reordering on interval synchronization. Of course, in the benign case, out-of-order arrival could cause packets in an interval u to arrive after the interval marker packet for u (and thus be interpreted by Bob as part of interval $u+1$). Fortunately, out-of-order arrival should not cause any false alarms as long as the number of packets arriving out of order before the interval marker is a small fraction of αT, where α the false-alarm threshold. (Note that because we focus on PQM protocols that operate at the network layer, at this layer TCP retransmissions do not look like out-of-order packets.)

Indeed, out-of-order arrive limits the choice of αT. To see how, consider an ordered stream of packets transmitted by a sender (*e.g.,* 1,2,3,4,5,6,7,8). Let a "reordered packet" be some packet that arrives at the receiver later than the packets after it in the ordered stream sent by the sender (*e.g.,* . in received stream 1,2,4,5,6,7,3,8, packet 3 is the reordered packet). Then, define the packet lag as the number of packets that were sent by the sender after the reordered packet, but were received at the receiver earlier than the reordered packet itself (*e.g.,* in received stream 1,2,4,5,6,7,3,8, packet 3 is the reordered packet and packet lag is 4).

To ensure natural packet reordering on the link does not cause a loss of interval synchronization between the sender and receiver, a good rule of thumb is to ensure that $\alpha T \geq 99^{th}$ percentile of packet lag. The packet lag depends on the the class of packets monitored by the PQM protocol. For instance:

- If the PQM protocol is in setting where no load balancing is used, (*i.e.,* packets sent by Alice to Bob are sent over a single physical path through the network, rather than split over multiple paths) then packet lag is typically very small, *i.e.,* about 10's of packet [88, Sec. III.A]. Thus, ensuring $\alpha T \geq 100$ is sufficient in this case.

- If the PQM protocol is used to monitor a single "layer-3 flow", *i.e.,* a set of IP packets with same (Source IP, Destination IP, Source Port, Destination Port, Protocol Numbuer), then we assume that packet lag is less than 128 packets. (This is the assumption made in IPSec). Thus, it suffices to take $\alpha T > 1280$.

- If the PQM protocol simultaneously monitors multiple layer 3 flows, then packet lag can be quite high. This is because different flows may be routed on different paths through the network; if there is a significant time delay between the different paths used by the different flows, then packet lag can be very high. The best way to determine packet lag in this setting is to measure it directly; however, we conjecture that even if there is a 10ms difference between the "fast path" used by one group of flows and the "slow path" used by another group of flows, for 1 Gbps flow of traffic, packet lag should be on the order of 10^9 bps / 64 bytes/packet * .01 sec $= 1.6x10^5$ packets, so we can use $\alpha T > 1.6x10^6$.

Also, if the link has many out-of-order packets even in the benign case, we can enforce interval synchronization by marking packets with a single bit denoting the parity of the interval number (note that if the adversary tampers with this mark, she only increases the likelihood that Alice will raise an alarm)

Finally, notice that if Eve drops or delays the marker packet for interval u, then she only increases the changes that Alice raises an alarm (since doing is equivalent to adding many packets to interval u and dropping many packets in interval $u+1$).

B.2.2 Asymmetric-key protocols

Our asymmetric secure sampling protocols (Section 3.4.2) use a different approach for interval synchronization. Here, the end of the interval is determined when the server sends out the 'salt release message'. Thus, there is no need to have the client send the server an 'interval marker packet'. We do, however, require Alice to be coarsely synchronized to Bob's clock, so that an adversary cannot replay old salt release messages (and use the old salt to form ACKs that trick the client into accepting an interval for which she should have raised an alarm).

In settings where the sender and receiver do not share a clock, the following simple protocol can be used to securely synchronize Alice's clock to Bob's clock to within 1.5 round trip times (RTT) (*e.g.,* $\tau = 150$ ms). Notably, this protocol does not require either Alice or Bob to keep any state beyond their keys and local clocks. The protocol also does not require Alice and Bob to trust one another, and does not affect Alice's global clock that is used when interacting with other parties.

Simple synchronization protocol. Suppose Alice has some local secret key k_A (she does not need to shared this key with anyone).

1. At time t_A (on Alice's clock) Alice sends Bob the message $\text{MAC}_{k_A}(t_A)$.
2. Bob receives this message at time t_B (on Bob's clock) and responds with digitally signed message
   $\xi = \text{Sign}_{SK_B}(t_B, \text{MAC}_{k_A}(t_A))$.[3]
3. Alice accepts Bob's message ξ if $\text{Verify}_{PK_B}(\xi)$ returns $(t_B, \text{MAC}_{k_A}(t_A))$, the MAC is correct, and Alice's current local time t'_A fulfills $t'_A < t_A + \tau$. If Alice accepted Bob's message, she computes $\Delta_B = \tau - t'_A$, and from now on, whenever interacting with Bob she offsets her clock by a factor of Δ_B.

If, after many attempts, Alice fails to receive a valid response to her synchronization message, then she decides to raise an alarm. After Alice accepts, her local clock (after being offset by Δ_B) is within τ seconds from Bob's regardless of Eve's actions. Indeed, a sufficient condition is that any accepted message ξ was sent by Bob when his local time was t'_B and Alice's local time was after t'_A. Violating either of these would contradict the security of the digital signature and MAC schemes.

B.3 Secure sampling needs PRFs

To give an example of why non-cryptographic hash functions are not insufficient in our sampling protocols, suppose that the Probe function of equation (3.3) was implemented using a CRC keyed with a secret modulus, as in [33], instead of with a PRF. Approximate the CRC function as $h_k(x) = x \mod k$, and consider the following attack: Eve starts by observing the interactions on the channel, and records the list of packets that were not acknowledged. Then, whenever she sees a new packet that is within a small additive distance of old packet that was not acknowledged, she drops the packet. Thus, Eve can drop non-probe packets with high probability, and she can bias the estimate V well below the true failure rate.

[3] While computing and verifying digital signatures typically takes on the order of 3ms, and is thus insignificant as compared to the 150ms interval consider here. Furthermore, this time delay is constant and known and can be subtracted from Δ_B.

B.4 Prehashing packets

As discussed in Section 3.5.4, arguably the most expensive part of our sketching protocol is the computation of the per-packet hash. We now show how to reduce the cost of this computation by (1) first mapping packets from from U to a short n_1-bit string using the efficient ε_g-almost universal hash function, and (2) then using a PRF (or 4-wise independent) hash to map from n_1-bit to the sketch. Our approached is based on that of Thorup and Zhang [100].

Preliminaries. Recall that U is the universe of all possible packets, and $\mathbf{v}$ is the characteristic vector of the stream of packets. Let $g : U \to \{0,1\}^{n_1}$ be an ε_g-almost universal hash function, as defined in Section 3.3. The hash function g maps the packet stream containing elements in U to a new 'intermediate' stream of n_1-bit strings. Now, we let $\mathbf{u}$ be an 'intermediate vector' which is the characteristic vector of this new stream of n_1-bit strings. And finally, recall that $\mathbf{w}$ is the sketch vector of length N.

Thus, our approach amounts to using the ε_g-almost universal hash g to hash $\mathbf{v}$ the 'intermediate vector' $\mathbf{u}$, and then using a a second-moment estimation scheme to hash $\mathbf{u}$ down to the sketch $\mathbf{w}$. Thus, the second-moment estimation scheme estimates the second moment of $\mathbf{u}$, rather than the real characteristic vector $\mathbf{v}$! We now show that, if ε_g is sufficiently small, this does very little damage, since $\|\mathbf{u}\|_2 \approx \|\mathbf{v}\|_2$.

Theorem B.4.1. *Given a vector $\mathbf{v} \in 2^{|U|}$ and $\mathbf{u} \in \mathbb{R}^{2^{n_1}}$. Then if $g : U \to \{0,1\}^n$ is an ε_g-almost 2-wise independent hash function per equation (3.1), is used to map $\mathbf{v}$ to $\mathbf{u}$ according to the algorithm $u_{g(x)} += v_x$ (*i.e., *$\forall\, x \in \mathbf{v}$ the $g(x)^{th}$ counter in $\mathbf{u}$ is incremented with value v_x) then*

$$\Pr\left[\,|\, \|\mathbf{u}\|_2 - \|\mathbf{v}\|_2 \,| > \delta_1 \|\mathbf{v}\|_2\right] < \delta_2 \tag{B.3}$$

as long as

$$|\mathbf{v}|_1 > \frac{\delta_1 \delta_2}{\varepsilon_g} \tag{B.4}$$

Recall that $|\mathbf{v}|_1 = A + D$. Returning the proof of Theorem 3.5.1, for (α, β, δ)-secure PQM we would like (B.3) to hold particularly when $D = \alpha T$ and $D = \beta T$, with $\delta_1 \ll \varepsilon = \frac{\beta - \alpha}{\alpha + \beta}$. To be more conservative, we will take $|\mathbf{v}|_1 = T$, and $\delta_1 = \frac{\varepsilon}{10}$. We'll also set $\delta_2 = \frac{\delta}{100}$. Then (α, β, δ)-secure PQM require the hash function g to have ε_g as :

$$\varepsilon_g < \frac{\varepsilon\delta}{10^3 T} = \frac{\delta}{10^3 T} \frac{\beta - \alpha}{\alpha + \beta} \tag{B.5}$$

Proof of Theorem B.4.1. Let v_a be the a^{th} entry of characteristic vector $\mathbf{v}$. Now, start with the observation that

$$\|\mathbf{u}\|_2^2 = \sum_{g(a)=g(b)} v_a v_b \tag{B.6}$$

$$= \sum_a v_a^2 + \sum_{a \neq b, g(a)=g(b)} v_a v_b \tag{B.7}$$

$$= \|\mathbf{v}\|_2^2 + \sum_{a \neq b} v_a v_b Y_{a,b} \tag{B.8}$$

where we define the random variable $Y_{a,b}$ as

$$Y_{a,b} = \begin{cases} 1 & \text{if } g(a) = g(b), a \neq b, \\ 0 & \text{else.} \end{cases} \tag{B.9}$$

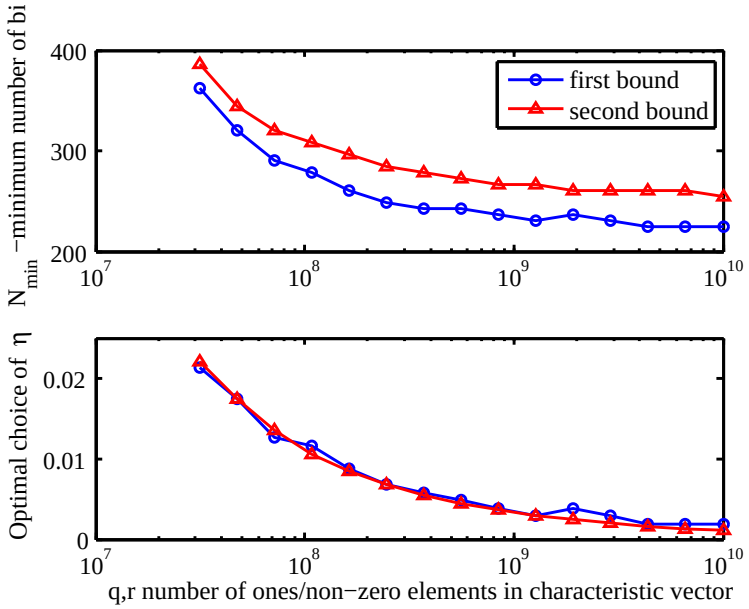


Figure B.1: An example of how to use Theorem B.5.4.

and from (B.8) we take the expectation over the randomness in g and find that

$$\mathbb{E}[\ |\ \|\mathbf{u}\|_2^2 - \|\mathbf{v}\|_2^2\ |\ ] \leq \sum_{a,b} |v_a v_b| \mathbb{E}[|Y_{a,b}|] \tag{B.10}$$

$$\leq \sum_{a,b} |v_a v_b| \cdot \varepsilon_g \tag{B.11}$$

$$= (|\mathbf{v}|_1^2 - \|\mathbf{v}\|_2^2) \cdot \varepsilon_g \tag{B.12}$$

where the first inequality follows from (B.8), the second inequality follows because per equation (3.1) the collision probability of g is ε_g.

Now, we would like to ensure that $\|\mathbf{u}\|_2$ provides a good estimate of $\|\mathbf{v}\|_2$. That is, we would like to satisfy (B.3). Using Markov's inequality, we have

$$\Pr\left[\left|\|\mathbf{u}\|_2^2 - \|\mathbf{v}\|_2^2\right| > \delta_1 \|\mathbf{v}\|_2^2\right] \leq \frac{\mathbb{E}[\ |\ \|\mathbf{u}\|_2^2 - \|\mathbf{v}\|_2^2\ |\ ]}{\delta \|\mathbf{v}^2\|_2} \tag{B.13}$$

$$\leq \frac{(|\mathbf{v}|_1^2 - \|\mathbf{v}\|_2^2)}{\|\mathbf{v}\|_2^2} \frac{\varepsilon_g}{\delta_1} \tag{B.14}$$

$$\leq |\mathbf{v}|_1 \frac{\varepsilon_g}{\delta_1} \tag{B.15}$$

And rearranging the last inequality we know that (B.3) holds as long as (B.4) holds, which completes the proof. □

B.5 Second-moment estimation with CCF

We start by restating a result of Thorup and Zhang from [100].

Theorem B.5.1. *For any characteristic vector* $\mathbf{v} \in \mathbb{R}^{|U|}$*, if we construct sketch* $\mathbf{w} \in \mathbb{Z}^N$ *according to the CCF-algorithm with two independent 4-universal hash functions* $h : U \to [N]$ *and* $s : U \to \{-1, 1\}$ *as* $w_{h(a)} += s(a)v_a$ *then we have*

$$\mathbb{E}[\|\mathbf{w}\|_2^2] = \|\mathbf{v}\|_2^2 \tag{B.16}$$

$$\mathbb{VAR}[\|\mathbf{w}\|_2^2] = \tfrac{2}{N}(\|\mathbf{v}\|_2^4 - \|\mathbf{v}\|_4^4) \tag{B.17}$$

B.5.1 CCF with 4-wise independent hashes

To show that the number of bins required for CCF with a 4-wise independent hash is $N > \frac{2}{\varepsilon^2\delta}$ we prove the following theorem.

Theorem B.5.2. *For any vector* $\mathbf{v} \in \mathbb{R}^U$ *sketched according to the CCF-algorithm with a 4-wise independent hash to obtain* $\mathbf{w} \in \mathbb{Z}^N$*, then for any* $\varepsilon \in (0, 1)$ *we have*

- *If* $\|\mathbf{v}\|_2^2 \le \alpha T$ *and* $\alpha T > 1$ *then*

$$\Pr\left[\|\mathbf{w}\|_2^2 < (1+\varepsilon)\alpha T\right] \le \tfrac{2}{N\varepsilon^2}(1 - \tfrac{1}{\alpha T}) \tag{B.18}$$

- *If* $\|\mathbf{v}\|_2^2 \ge \beta T$ *and* $\beta T > 1$ *then*

$$\Pr\left[\|\mathbf{w}\|_2^2 > (1-\varepsilon)\beta T\right] \le \tfrac{2}{N\varepsilon^2} \tag{B.19}$$

Then, recall that we set our threshold as $\Gamma = (1-\varepsilon)\beta T = (1+\varepsilon)\alpha T$ for $\varepsilon = \frac{\beta-\alpha}{\beta+\alpha}$. Following the approach in the proof of Theorem 3.5.1, we can show that the scheme is (α, β, δ)-secure PQM protocol per Definition 3.2.1 if if we take a sketch with at least $N > \frac{2}{\delta\varepsilon^2} = \frac{2}{\delta}(\frac{\beta+\alpha}{\beta-\alpha})^2$ bins.

Proof of Theorem B.5.2. We start with the first item and write:

$$\Pr\left[\|\mathbf{w}\|_2^2 < (1+\varepsilon)\alpha T\right] \le \frac{\mathbb{VAR}[\|\mathbf{w}\|_2^2]}{\left((1+\varepsilon)\alpha T - \mathbb{E}[\|\mathbf{w}\|_2^2]\right)^2} \tag{B.20}$$

$$\le \frac{2}{N}\frac{(\|\mathbf{v}\|_2^4 - \|\mathbf{v}\|_4^4)}{\left((1+\varepsilon)\alpha T - \|\mathbf{v}\|_2^2\right)^2} \tag{B.21}$$

$$\le \frac{2}{N}\frac{(\|\mathbf{v}\|_2^4 - \|\mathbf{v}\|_2^2)}{\left((1+\varepsilon)\alpha T - \|\mathbf{v}\|_2^2\right)^2} \tag{B.22}$$

$$\le \frac{2}{N}\frac{(\alpha T)^2 - \alpha T}{(\varepsilon\alpha T)^2} \tag{B.23}$$

$$= \frac{2}{N\varepsilon^2}(1 - \tfrac{1}{\alpha T}) \tag{B.24}$$

where the first inequality follows from the Chebyshev bound, the first equality follows from applying Theorem B.5.1, and second inequality follows because for any $\mathbf{v} \in \mathbb{Z}^N$ we have $\|\mathbf{v}\|_2^2 \le \|\mathbf{v}\|_4^4$ (this holds because $\mathbf{v}$ has integer valued entries), and the third inequality follows from the fact that $\|\mathbf{v}\|_2^2 \le \alpha T$ and $\alpha T > 1$.

Next, consider the second item and write:

$$\Pr\left[\|\mathbf{w}\|_2^2 < (1-\varepsilon)\beta T\right] \le \frac{\mathbb{VAR}[\|\mathbf{w}\|_2^2]}{\left(\mathbb{E}[\|\mathbf{w}\|_2^2] - (1-\varepsilon)\beta T\right)^2} \tag{B.25}$$

$$\le \frac{2}{N}\frac{(\|\mathbf{v}\|_2^4 - \|\mathbf{v}\|_4^4)}{\left(\mathbb{E}[\|\mathbf{w}\|_2^2] - (1-\varepsilon)\beta T\right)^2} \tag{B.26}$$

$$\le \frac{2}{N}\frac{\|\mathbf{v}\|_2^4}{\left(\|\mathbf{v}\|_2^2 - (1-\varepsilon)\beta T\right)^2} \tag{B.27}$$

where again the first inequality follows from the Chebyshev bound, the first equality follows from applying Theorem B.5.1, and second inequality follows because $\|\mathbf{v}\|_4^4 \geq 0$ since $\beta T > 1$. To bound (B.27), we use the following claim.

Claim B.5.3. *For any $a > 0$, and function $f(x) = \frac{x}{x-a}$ decreases with x when $x > a$.*

Proof. Follows because $\frac{df}{dx} = -\frac{a}{(x-a)^2} < 0$ and there are no singularities in $f(x)$ for all $0 < a < x$. □

To apply Claim B.5.3, let $x = \|\mathbf{v}\|_2^2$ and $a = (1-\varepsilon)\beta T$ and recall from the statement of the theorem that $x \geq \beta T > a$. From Claim B.5.3 we note that (B.27) takes on its largest value when $x = \beta T$, so we can write (B.27) as

$$\Pr\left[\|\mathbf{w}\|_2^2 < (1-\varepsilon)\beta T\right] \leq \frac{2}{N}\frac{(\beta T)^2}{(\beta T - (1-\varepsilon)\beta T)^2} \tag{B.28}$$

$$= \tfrac{2}{N\varepsilon^2} \tag{B.29}$$

which completes the proof. □

B.5.2 CCF with PRFs

First, we prove a precise version of Theorem B.5.4.

Theorem B.5.4. *For any vector $\mathbf{v} \in \mathbb{Z}^U$, choosing the $N \times U$ matrix S uniformly from $\mathcal{S}_{\text{CCF}}$ and setting $\mathbf{w} = S\mathbf{v}$, we have that for all $\varepsilon \in [0,1)$ and all $q, r > N$*

1. *If $\mathbf{v} \in \{-1,0,1\}^U$, and $\|v\|_2^2 \leq q$, then for $\eta \in [0, \frac{1}{2}\sqrt{\varepsilon^2+10\varepsilon+9} - \frac{1}{2}(\varepsilon+3))$ and $y \doteq \frac{(1+\varepsilon)(1-\eta)}{(1+\eta)^2} - 1$:*

$$\Pr\left[\|\mathbf{w}\|_2^2 > (1+\varepsilon)q\right] \leq 2Ne^{-\frac{\eta^2 q}{3N}} + e^{-\frac{N}{2}(y^2/2 - y^3/3)} \tag{B.30}$$

2. *If the number of* non-zero *entries in $\mathbf{v}$ is r, then for $\eta \in \left(0, \frac{1}{2-\varepsilon}(3-2\varepsilon-\sqrt{5\varepsilon^2-14\varepsilon+9})\right)$ and $y \doteq \frac{(1-\eta)^2}{1+\eta}(1-\frac{\varepsilon}{2}) - (1-\varepsilon)$ it follows that*

$$\Pr\left[\|\mathbf{w}\|_2^2 < (1-\varepsilon)r\right] \leq 2Ne^{-\frac{\eta^2 r}{3N}} + e^{-N\frac{\varepsilon}{3(1+\eta)}y} \tag{B.31}$$

For a fixed ε, suppose we want to choose values for q, r, N that ensure that both the first item and the second item occur with probability at most δ. To use Theorem B.5.4, we need to choose a value of η (within the appropriate range) and plug it into (B.30)-(B.31) to obtain q, r and N. This is not as simple as it initially appears: the first term ($2Ne^{-\frac{\eta^2 q}{3N}}$) in equations (B.30)-(B.31), *increases* in η, which the second term *decreases* in η. Thus, we must tradeoff between these two terms in order to find an optimal choice of η, *i.e.,* one that minimizes N for a given choice of $q, r,$. This optimization can be messy, and so we do it in MATLAB. For example, in Figure B.4 we set $\varepsilon = \frac{1}{3}$ and for $q = r$ we require that both the first item and the second item occur with probability at most $\delta = \frac{1}{100}$. For each value of q, we show the choice of η that minimizes N for both the first item and the second item in Theorem B.5.4.

Proof of Theorem B.5.4. The main observation we make is that, with high probability, the ± 1 entries of $\mathbf{v}$ are distributed evenly among the coordinates of $\mathbf{w}$. Conditioned on this happening, we can then apply the analysis of Achlioptas [5].

Definitions. We need the following definitions.

- We write v_x for the x^{th} element in $\mathbf{v}$.
- Define for $i \in [N]$ the set $Q_i = \{x \in U \mid h(x) = i\}$ where h is the pseudorandom hash function.
- Define D_i as the number of non-zero entries in $\mathbf{v}$ that hash to the i^{th} bin the sketch $\mathbf{w}$. That is $D_i = |\{v_x | v_x \neq 0, x \in Q_i\}|$.
- Define Y_x as an unbiased ± 1 random variable for each $x \in U$.

Our proof proceeds as follows. We first obtain a bound on D_i for each i. When then use the bounds on D_i to prove the first item (B.30), and then use them to prove the second item (B.31).

Bounding D_i. Let E_i denote the event that $\exists i \in [N]$ such that $D_i > (1+\eta)q/N$ or $D_i < (1-\eta)q/N$. Then, for $\eta \in [0,1)$, we have that

$$\Pr[E_1] \leq N\left(\Pr[D_i > (1+\eta)\tfrac{q}{N}] + \Pr[D_i < (1-\eta)\tfrac{q}{N}]\right) \leq N\left(e^{-\frac{\eta^2}{3}\frac{q}{N}} + e^{-\frac{\eta^2}{2}\frac{q}{N}}\right) \tag{B.32}$$

which is a straightforward application of a union bound followed by the Chernoff bound.[4]

Bounding the first item. Now we condition on $\neg E_1$. Let $\gamma = \frac{1+\varepsilon}{(1+\eta)^2}$ and write:

$$\begin{aligned}\Pr[\|\mathbf{w}\|_2^2 > (1+\varepsilon)q \mid \neg E_1] &= \Pr[\sum_{i=1}^{N} D_i^2 \left(\tfrac{1}{D_i}\sum_{x\in Q_i} Y_x v_x\right)^2 > (1+\varepsilon)q \mid \neg E_1] \\ &= \Pr[\sum_{i=1}^{N} \left(\tfrac{1}{D_i}\sum_{x\in Q_i} Y_x v_x\right)^2 > \gamma\tfrac{N^2}{q} \mid \neg E_1]\end{aligned}$$

where first equality comes from expanding $\mathbf{w}$ as $\mathcal{S}\mathbf{v}$ and then multiplying by $\frac{D_i}{D_i}$, and the second equality follows from the fact that conditioning on $\neg E_i$ implies that $D_i \leq (1+\eta)q/N$. Next, set $\mathbf{Y}_i$ to be the vector of all Y_x for each $v_x \in -1, 1, x \in Q_i$. Set $\mathbf{u}_i$ the vector with entries $\frac{v_x}{\sqrt{D_i}}$ for each $v_x \in \{-1,1\}, x \in Q_i$. Notice that both $\mathbf{Y_i}$ and $\mathbf{u_i}$ have length D_i, and $\|\mathbf{u}_i\|_2^2 = 1$ so $\mathbf{u_i}$ is a unit vector. Now we write

$$\begin{aligned}&= \Pr[e^{t\sum_{i=1}^{N}\langle\frac{\mathbf{Y}_i}{\sqrt{D_i}},\mathbf{u}_i\rangle^2} > e^{t\gamma\frac{N^2}{q}} \mid \neg E_1] \\ &\leq e^{-t\gamma\frac{N^2}{q}} \prod_{i=1}^{N} \mathbb{E}[e^{t\langle\frac{\mathbf{Y}_i}{\sqrt{D_i}},\mathbf{u}_i\rangle^2} \mid \neg E_1]\end{aligned}$$

[4] We use the following Chernoff bounds. Let X_i be i.i.d indicator variables with mean μ, and let

$$\Pr\left[\sum_{i=1}^{n} X_i \leq (1-\gamma)N\mu\right] \leq e^{-\gamma^2 N\mu/C_1} \tag{B.33}$$

$$\Pr\left[\sum_{i=1}^{n} X_i \geq (1+\gamma)N\mu\right] \leq e^{-\gamma^2 N\mu/C_2} \tag{B.34}$$

If $0 < \gamma < 1$ then [9, Fact 4] gives $C_1 = 2$ and $C_2 = 3$. If $0 < \gamma < \frac{1}{2}$ then [6, Thm. 19] gives $C_1 = C_2 = 2\ln 2$.

where the inequality follows from the Markov bound. Now we are ready to apply the result of Achlioptas. We restate equation (2) and Lemma 5.2 of [5] here, using our own terminology.

Lemma B.5.5 (From [5])**.** *For $t \in [0, D_i/2]$, unit vector $\mathbf{u}_i$ (*i.e., *$\|\mathbf{u}_i\|_2^2 = 1$) and $\mathbf{Y}_i$ chosen uniformly from $\{1,-1\}^{D_i}$ we have that*

$$\mathbb{E}[e^{t\langle \frac{\mathbf{Y}_i}{\sqrt{D_i}}, \mathbf{u}_i\rangle^2}] \leq \frac{1}{\sqrt{1-2t/D_i}} \tag{B.35}$$

$$\mathbb{E}[\langle \tfrac{\mathbf{Y}_i}{\sqrt{D_i}}, \mathbf{u}_i\rangle^2] = \frac{1}{D_i} \tag{B.36}$$

$$\mathbb{E}[\langle \tfrac{\mathbf{Y}_i}{\sqrt{D_i}}, \mathbf{u}_i\rangle^4] = \frac{3}{D_i^2} \tag{B.37}$$

Now, using Achlioptas's result in (B.35) we write

$$\begin{aligned}
\Pr[\|\mathbf{w}\|_2^2 > (1+\varepsilon)q \mid \neg E_1] &\leq e^{-t\gamma\frac{N^2}{q}} \prod_{i=1}^{N} \mathbb{E}[e^{t\langle \frac{\mathbf{Y}_i}{\sqrt{D_i}}, \mathbf{u}_i\rangle^2} \mid \neg E_1] \\
&\leq e^{-t\gamma\frac{N^2}{q}} \prod_{i=1}^{N} \frac{1}{\sqrt{1-2t/D_i}} \\
&\leq e^{-t\gamma\frac{N^2}{q}} (1 - \tfrac{2t}{(1-\eta)q/N})^{-\frac{N}{2}} \doteq v(t) \qquad \text{(B.38)}
\end{aligned}$$

where the last inequality (B.38) follows from conditioning on $\neg E_i$ which implies that $(1-\eta)q/N < D_i$ for all $i \in [N]$. Note that result of Achlioptas in (B.35) to hold, we must have $0 \leq t < D_i/2 \leq \frac{(1+\eta)q}{2N}$ where the last inequality here follows from the fact that $\neg E_i$ implies that $D_i < (1+\eta)q/N$.

Optimizing and bounding t. Next, we optimize $v(t)$ in (B.38), by finding t such that $\frac{dv(t)}{dt} = 0$.

$$\begin{aligned}
\tfrac{dv(t)}{dt} = -\tfrac{\gamma N^2}{q} v(t) + (-\tfrac{N}{2})(-\tfrac{2}{(1-\eta)q/N})(1 - \tfrac{2t}{(1-\eta)q/N})^{-1} v(t) &= 0 \\
\tfrac{\gamma N^2}{q}(1 - \tfrac{2t}{(1-\eta)q/N}) &= \tfrac{N^2}{(1-\eta)q} \\
t &= \tfrac{q}{2N}\left((1-\eta) - \tfrac{(1+\eta)^2}{1+\varepsilon}\right) \qquad \text{(B.39)}
\end{aligned}$$

where the last equality uses the fact that $\gamma \doteq \frac{1+\varepsilon}{(1+\eta)^2}$. Now recall that for Achlioptas's result in (B.35) to hold, we need to ensure that $0 \leq t < \frac{(1+\eta)q}{2N}$. Using (B.39), we write

$$\begin{aligned}
0 &\leq t \\
0 &\leq \tfrac{q}{2N}\left((1-\eta) - \tfrac{(1+\eta)^2}{1+\varepsilon}\right) \\
\tfrac{(1+\eta)^2}{1+\varepsilon} &\leq (1-\eta) \\
\tfrac{(\eta^2+3\eta)}{1-\eta} &\leq \varepsilon \qquad \text{(B.40)}
\end{aligned}$$

and we also need

$$\begin{aligned}
t &< \tfrac{(1+\eta)q}{2N} \\
\tfrac{q}{2N}\left((1-\eta) - \tfrac{(1+\eta)^2}{1+\varepsilon}\right) &< \tfrac{(1+\eta)q}{2N} \\
-\tfrac{(1+\eta)^2}{1+\varepsilon} &< 2\eta \\
-\left(1 + \tfrac{(1+\eta)^2}{2\eta}\right) &< \varepsilon \qquad \text{(B.41)}
\end{aligned}$$

Now, (B.41) holds for any $\eta \in [0,1)$. But, we will need to ensure that our choice of $\eta \in [0,1)$ satisfies (B.40).

Returning now to (B.38), plug (B.39) into (B.38) to get

$$\Pr[\|\mathbf{w}\|_2^2 > (1+\varepsilon)q \mid \neg E_1] \leq \left(e^{-y}(1+y)\right)^{\frac{N}{2}} \tag{B.42}$$

where we define

$$y \doteq \frac{(1+\varepsilon)(1-\eta)}{(1+\eta)^2} - 1 \tag{B.43}$$

and solving inequality (B.40), we find that (B.42) holds as long as $\eta \in [0,1)$ satisfies

$$0 < \eta < \tfrac{1}{2}\left(\sqrt{\varepsilon^2 + 10\varepsilon + 9} - (\varepsilon + 3)\right) \tag{B.44}$$

Notice from (B.43) that the bound in (B.44) this implies that (B.42) holds for the region $y \in [0,\varepsilon)$. Now, Achlioptas observes that $e^{-y}(1+y) \leq e^{(-y^2/2+y^3/3)}$ for any $y \in (0,1)$. Since for us $y \in (0,\varepsilon)$, and $\varepsilon < 1$ we finally have

$$\Pr[\|\mathbf{w}\|_2^2 > (1+\varepsilon)q \mid \neg E_1] \leq e^{-\frac{N}{2}(y^2/2 - y^3/3)} \tag{B.45}$$

which decays exponentially in N. □

Bounding the second item. Let r be the number of non-zero entries in $\mathbf{v}$. We will bound $\Pr[\|\mathbf{w}\|_2^2 < (1-\varepsilon)r]$. Define E_1 as before, only this time use r instead of q. Again we condition on $\neg E_1$.

$$\begin{aligned}
\Pr[\|\mathbf{w}\|_2^2 < (1-\varepsilon)r \mid \neg E_1] &= \Pr[\sum_{i=1}^{N} D_i^2 \left(\tfrac{1}{D_i}\sum_{x\in Q_i} Y_x v_x\right)^2 < (1-\varepsilon)r \mid \neg E_1] \\
&= \Pr[\sum_{i=1}^{N} \left(\tfrac{1}{D_i}\sum_{x\in Q_i} Y_x v_x\right)^2 < \tfrac{(1-\varepsilon)}{(1-\eta)^2}\tfrac{N^2}{r} \mid \neg E_1]
\end{aligned}$$

where first equality comes from the expanding $\|\mathbf{w}\|_2^2$and then multiplying by $\frac{D_i}{D_i}$, and the second equality follows from the fact that conditioning on $\neg E_i$ implies that $(1-\eta)r/N < D_i$. Next, we let $c_i^2 = \sum_{x\in Q_i} \frac{v_x^2}{D_i}$. Now observe that $c_i^2 = \frac{1}{D_i}\sum_{x\in Q_i} v_x^2 \geq \frac{1}{D_i} D_i = 1$ since the entries of v are integers (and D_i is the number of non-zero entries in v that are in Q_i). We now multiply by $\frac{c_i}{c_i}$:

$$\begin{aligned}
&= \Pr[\sum_{i=1}^{N} c_i^2 \left(\sum_{x\in Q_i} Y_x \tfrac{v_x}{D_i c_i}\right)^2 < \tfrac{(1-\varepsilon)}{(1-\eta)^2}\tfrac{N^2}{r} \mid \neg E_1] \\
&\leq \Pr[\sum_{i=1}^{N} \left(\sum_{x\in Q_i} Y_x \tfrac{v_x}{D_i c_i}\right)^2 < \tfrac{(1-\varepsilon)}{(1-\eta)^2}\tfrac{N^2}{r} \mid \neg E_1]
\end{aligned}$$

where the inequality follows from the fact that $c_i^2 \geq 1$. We now set $\mathbf{Y}_i$ to be the vector of all Y_x for each $v_x \neq 0, x \in Q_i$. Set $\mathbf{u}_i$ the vector with entries $\frac{v_x}{\sqrt{D_i}c_i}$ for each $v_x \neq 0, x \in Q_i$. Notice that both $\mathbf{Y}_i$ and $\mathbf{u}_i$ have length D_i, and that $\mathbf{u}_i$ is a unit vector, since $\|\mathbf{u}_i\|_2^2 = \frac{1}{D_i c_i^2}\sum_{x\in Q_i} v_x = \frac{c_i^2}{c_i^2} = 1$. We write

$$
\begin{aligned}
&= \Pr[\sum_{i=1}^{N} \langle \tfrac{\mathbf{Y}_i}{\sqrt{D_i}}, \mathbf{u}_i\rangle^2 < \tfrac{(1-\varepsilon)}{(1-\eta)^2}\tfrac{N^2}{r} \mid \neg E_1] \\
&\leq e^{t\frac{(1-\varepsilon)}{(1-\eta)^2}\frac{N^2}{r}} \prod_{i=1}^{N} \mathbb{E}[e^{-t\langle \frac{\mathbf{Y}_i}{\sqrt{D_i}}, \mathbf{u}_i\rangle^2} \mid \neg E_1]
\end{aligned}
$$

where the first inequality follows from the Markov bound, and we require that $t > 0$. We now follow that analysis in Achiloptas, and expand out the quantity inside the expectation to obtain:

$$
\leq e^{t\frac{(1-\varepsilon)}{(1-\eta)^2}\frac{N^2}{r}} \prod_{i=1}^{N} \mathbb{E}[1 - t\langle \tfrac{\mathbf{Y}_i}{\sqrt{D_i}}, \mathbf{u}_i\rangle^2 + \tfrac{t^2}{2}\langle \tfrac{\mathbf{Y}_i}{\sqrt{D_i}}, \mathbf{u}_i\rangle^4 \mid \neg E_1]
$$

Now we can apply Achiloptas's results from (B.36) and (B.37) to obtain:

$$
\leq e^{t\frac{(1-\varepsilon)}{(1-\eta)^2}\frac{N^2}{r}} \prod_{i=1}^{N} \left(1 - \tfrac{t}{D_i} + \tfrac{t^2}{2}\tfrac{3}{D_i^2}\right)
$$

and conditioning on $\neg E_1$ gives us:

$$
\leq e^{t\frac{(1-\varepsilon)}{(1-\eta)^2}\frac{N^2}{r}} \left(1 - \tfrac{1}{1+\eta}\tfrac{tN}{r} + \tfrac{3}{2(1-\eta)^2}(\tfrac{tN}{r})^2\right)^N
$$

For convience, we'll now let $\tau = \frac{tN}{r}$, and rewrite this as

$$
= \left(e^{\frac{(1-\varepsilon)}{(1-\eta)^2}\tau}\left(1 - \tfrac{1}{1+\eta}\tau + \tfrac{3}{2(1-\eta)^2}\tau^2\right)\right)^N \doteq \nu(\tau)^N \tag{B.46}
$$

Bounding equation (B.46). We now need to find a choice of $\tau > 0$ that causes (B.46) to decay with N. It will suffice to find τ that causes $\nu(\tau)$ to decay exponetially, *i.e.,* we want $\nu(\tau) \sim e^{-\chi}$ for some $\chi > 0$. To do this, we start by rewriting $\nu(\tau)$ in the following way:

$$
\nu(\tau) = e^{\frac{(1-\varepsilon)}{(1-\eta)^2}\tau}\left(1 - \tfrac{1}{1+\eta}\tau \cdot \left(1 - \tfrac{3}{2}\tfrac{(1+\eta)}{(1-\eta)^2}\cdot\tau\right)\right)
$$

Notice that $\nu(\tau)$ is the product of a polynomial and exponential with postive argument (that grows). Notice that the only way we can hope to make $\nu(\tau)$ decay, is if we require the polynomial to decay. To do this, we need to ensure that the expression $(1 - \frac{3}{2}\frac{(1+\eta)}{(1-\eta)^2}\cdot\tau)$ is positive. Thus, we shall choose $\tau = \frac{\varepsilon}{2}(\frac{3}{2}\frac{(1+\eta)}{(1-\eta)^2})^{-1}$. Subsituting in the value for τ gives us:

$$
= e^{\frac{1-\varepsilon}{1+\eta}\frac{\varepsilon}{3}}\left(1 - (\tfrac{1-\eta}{1+\eta})^2\tfrac{\varepsilon}{3}\cdot(1 - \tfrac{\varepsilon}{2})\right)
$$

The series expansion of an exponential tell us that for any non-negative x we have the identity $1 - x \leq e^{-x}$. Since the quantity $(\frac{1-\eta}{1+\eta})^2 \frac{\varepsilon}{3} \cdot (1 - \frac{\varepsilon}{2})$ is non-negative for every $\varepsilon \in (0, 1)$, we can apply this identity here:

$$\begin{aligned} &\leq \exp\left(\tfrac{1-\varepsilon}{1+\eta}\tfrac{\varepsilon}{3} - (\tfrac{1-\eta}{1+\eta})^2 \tfrac{\varepsilon}{3} \cdot (1 - \tfrac{\varepsilon}{2})\right) \\ &= e^{-\frac{\varepsilon}{3(1+\eta)}} \exp\left(\tfrac{(1-\eta)^2}{1+\eta}(1 - \tfrac{\varepsilon}{2}) - (1 - \varepsilon)\right) \end{aligned} \tag{B.47}$$

It follows from (B.47) that proving that $\nu(\tau)$ decays exponentially amounts to ensuring that

$$y(\eta, \varepsilon) \doteq \frac{(1-\eta)^2}{1+\eta}(1 - \tfrac{\varepsilon}{2}) - (1 - \varepsilon) \geq 0 \tag{B.48}$$

and, recalling that $\eta, \varepsilon \in (0, 1)$ some MATHEMATICA magic finds that (B.48) holds as long as $\eta \in (0, c(\varepsilon))$, where

$$c(\varepsilon) = \tfrac{1}{2-\varepsilon}(3 - 2\varepsilon - \sqrt{5\varepsilon^2 - 14\varepsilon + 9}) \tag{B.49}$$

This bound on η, despite being ugly, makes sense. Notice that when $\varepsilon = 0$, we have that $\eta = 0$, and when $\varepsilon = 1$, we have $c(\varepsilon) = 1$ so that $\eta \in (0, 1)$. Also, we observe that y monotonically decreases in η, ranging from $y(0, \varepsilon) = \varepsilon$ to $y((c(\varepsilon), \varepsilon) = 0$.[5] We also observe that y monotonically increase in ε, ranging from $y(\eta, 0) = y(0, 0) = 0$ (since $\eta = 0$ when $\varepsilon = 0$), and $y(\eta, 1) = \frac{1}{2}\frac{(1-\eta)^2}{1+\eta}$ (and $\eta \in (0, 1)$ when $\varepsilon = 1$). [6]

Putting everything together, we finally have that as long as $\eta \in (0, c(\varepsilon))$ where $c(\varepsilon)$ is given in (B.49), then y as given in (B.48) is such that $y > 0$. Re-writing (B.46) using (B.47) and (B.48) as

$$\Pr[\|\mathbf{w}\|_2^2 < (1 - \varepsilon)r \mid \neg E_1] \leq e^{-N\frac{\varepsilon}{3(1+\eta)}y} \tag{B.50}$$

we can see that the error decays exponentially in N, as required. □

A simpler statement of the theorem

We now prove the version of Theorem B.5.4 that appears in Section 3.5.4.

Theorem B.5.6 (Theorem 3.5.2 restated.)**.** *For any vector $\mathbf{v} \in \mathbb{Z}^U$, choose the $N \times U$ matrix S uniformly from $\mathcal{S}_{\mathrm{CCF}}$ and set $\mathbf{w} = S\mathbf{v}$. Then, for all $\varepsilon \in [0, 1)$ and η such that $\left(\frac{1-\eta}{1+\eta}\right)^2 = \max\left(\frac{1+\frac{\varepsilon}{2}}{1+\varepsilon}, \frac{1-\frac{3\varepsilon}{4}}{1-\frac{\varepsilon}{2}}\right)$, choosing*

$$N \geq \tfrac{24}{\varepsilon^2} \ln \tfrac{2}{\delta} \tag{B.51}$$

$$q, r \geq \tfrac{3N}{\eta^2} \ln \tfrac{4N}{\delta} \tag{B.52}$$

ensures that the following two items occur with probability at least $1 - \delta$:

1. *If $\mathbf{v} \in \{-1, 0, 1\}^U$, and $\|v\|_2^2 \leq q$, then $\|\mathbf{w}\|_2^2 < (1 + \varepsilon)q$.*

[5] It's easy to see that when $\eta = 0$, then $y(0, \varepsilon) = \varepsilon$, and a simple check in MATHEMATICA shows that when $\eta = c(\varepsilon)$ as in (B.49), then $y(c(\varepsilon), \varepsilon) = 0$. By inspection, it's clear that y decreases in η.

[6] First consider the case where $\varepsilon = 0$. Now when $\varepsilon = 0$, $c(\varepsilon) = 0$, and the requirement that $\eta \in (0, c(\varepsilon))$ implies that $\eta = 0$. It follows that $y = 0$. Next consider the case where $\varepsilon = 1$, which means that for $\eta \in (0, 1)$, we have that $y(\eta, 0) = \frac{1}{2}\frac{(1-\eta)^2}{1+\eta}$. Now, since the derivative $\frac{dy}{d\varepsilon} = \frac{1+\eta(4-\eta)}{1+\eta} > 0$ for any $\eta \in (0, 1)$, we know that y grow monotonically in ε.

2. The number of non-zero *entries in* $\mathbf{v}$ *is* r*, then* $\|\mathbf{w}\|_2^2 > (1-\varepsilon)r$.

Proof. We show how to obtain the Theorem 3.5.2 from Theorem B.5.4. To ensure that the error probability is at most δ in (B.30) it suffices to set

$$2Ne^{-\frac{\eta^2 q}{3N}} \leq \tfrac{\delta}{2} \tag{B.53}$$

$$e^{-\frac{N}{2}(y_1^2/2 - y_1^3/3)} \leq \tfrac{\delta}{2} \tag{B.54}$$

And to ensure that the error probability is at most δ in (B.31) we need to set

$$2Ne^{-\frac{\eta^2 r}{3N}} \leq \tfrac{\delta}{2} \tag{B.55}$$

$$e^{-N\frac{\varepsilon}{3(1+\eta)}y} \leq \tfrac{\delta}{2} \tag{B.56}$$

Bounding N. Referring to (B.54), we need to choose $N > N_{\min,1}$ where:

$$N_{\min,1} = \frac{4}{y_1^2(1 - y_1/6)} \ln \tfrac{2}{\delta} \tag{B.57}$$

Where recall that $y_1 \doteq \frac{(1+\varepsilon)(1-\eta)}{(1+\eta)^2} - 1$. It's easy to see that $y_1 \in (0, \varepsilon)$ for any $\eta, \varepsilon \in (0,1)$. To simplify (B.57), we will now require that $y_1 \geq \varepsilon/2$, which means we can write:

$$\begin{aligned} &\leq \frac{4}{y_1^2(1 - \varepsilon/6)} \ln \tfrac{2}{\delta} \\ &\leq \frac{4}{(\varepsilon/2)^2(1 - \varepsilon/6)} \ln \tfrac{2}{\delta} \\ &\leq \frac{19.2}{\varepsilon^2} \ln \tfrac{2}{\delta} \end{aligned}$$

where the first inequality follows because $y \leq \varepsilon$, the second follows from $y \geq \varepsilon/2$, and the third follows from $\varepsilon \leq 1$. Now, instead of using the "ugly" expression for $N > N_{\min,1}$ in (B.57) to bound N, we have "nicer" bound on N that clearly shows the dependence of N on ε, δ as:

$$N \geq \frac{19.2}{\varepsilon^2} \ln \tfrac{2}{\delta} \tag{B.58}$$

Next, refer to (B.56), we need to choose $N > N_{\min,2}$ where:

$$N_{\min,2} = \tfrac{3(1+\eta)}{\varepsilon y_2} \ln \tfrac{2}{\delta} \tag{B.59}$$

Where recall that $y_2 = \frac{(1-\eta)^2}{1+\eta}(1 - \frac{\varepsilon}{2}) - (1 - \varepsilon)$. It's easy to see that $y_2 \in (0, \frac{\varepsilon}{2})$ for any $\eta \in (0,1)$. To simplify (B.57), we will now require that $y_2 \geq \varepsilon/4$ which means we can write:

$$\begin{aligned} &\leq \tfrac{12(1+\eta)}{\varepsilon^2} \ln \tfrac{2}{\delta} \\ &\leq \tfrac{24}{\varepsilon^2} \ln \tfrac{2}{\delta} \end{aligned}$$

where the first inequality follows from our choice of $y_2 \geq \varepsilon/4$ and the second from $\eta \leq 1$. Now we again have "nicer" bound on N (showing it's dependence of N on ε, δ) as:

$$N \geq \tfrac{24}{\varepsilon^2} \ln \tfrac{2}{\delta} \tag{B.60}$$

Comparing equations (B.58) and (B.60) we find that it suffices to choose N satisfying (B.60).

Bounding η. These nice bounds on N does not come free. To obtain (B.58), we need to ensure that $y_1 > \varepsilon/2$. We write

$$\frac{\varepsilon}{2} \leq y_1 \doteq \frac{(1+\varepsilon)(1-\eta)}{(1+\eta)^2} - 1 \tag{B.61}$$

$$\frac{1+\frac{\varepsilon}{2}}{1+\varepsilon} \leq \frac{1-\eta}{(1+\eta)^2} \tag{B.62}$$

Now since $\frac{1+\frac{\varepsilon}{2}}{1+\varepsilon} \leq \left(\frac{1-\eta}{1+\eta}\right)^2 \leq \frac{1-\eta}{(1+\eta)^2}$ it follows that (B.62) holds if

$$\frac{1+\frac{\varepsilon}{2}}{1+\varepsilon} \leq \left(\frac{1-\eta}{1+\eta}\right)^2 \tag{B.63}$$

Next, to obtain (B.60) we need ensure that $y_2 > \varepsilon/4$, so we write

$$\frac{\varepsilon}{4} \leq y_2 \doteq \frac{(1-\eta)^2}{1+\eta}(1-\tfrac{\varepsilon}{2}) - (1-\varepsilon) \tag{B.64}$$

and a similar argument show that (B.64) holds as long as

$$\frac{1-\frac{3\varepsilon}{4}}{1-\frac{\varepsilon}{2}} \leq \left(\frac{1-\eta}{1+\eta}\right)^2 \tag{B.65}$$

Bounding q, r. Referring to (B.53) and (B.55), we observe that is suffices to choose

$$q, r \geq \frac{3N}{\eta^2} \ln \frac{4N}{\delta} \tag{B.66}$$

Notice that this bound relies on both N, and η. We bounded N in (B.60). To minimize q, r, we want to chose η as large as possible, subject to the constraints in (B.63) and (B.65). Thus, it suffices to chose η such that

$$\left(\frac{1-\eta}{1+\eta}\right)^2 = \max\left(\frac{1+\frac{\varepsilon}{2}}{1+\varepsilon}\ ,\ \frac{1-\frac{3\varepsilon}{4}}{1-\frac{\varepsilon}{2}}\right) \tag{B.67}$$

and this completes our proof Theorem 3.5.2. □

Appendix C

Failure Localization

C.1 Vulnerabilities of Other FL Protocols

We sketch why the protocols of [86, 13, 10] do not satisfy our security definition.

An On-demand Secure Routing Protocol Resilient to Byzantine Failures [13]: Awerbuch, Holmer, Nita-Rotaru and Rubens present a statistical FL protocol in which Alice and Bob run a secure *failure detection* protocol, where Bob sends out authenticated acks for each packet he receives. Once the number of detected faulty exchanges exceeds some threshold, say β, then Alice appends a encrypted list of "probed nodes" to each *new* packet that she sends out. If a node is included in the list of probed nodes, it is expected to send Alice an ack when it receives the packet containing the list. The acks are (basically) formed as in our "onion reports". To localize failures, Alice chooses probed nodes according to a binary search algorithm, until she localizes a single link.

Now, consider an adversary Eve that sits at R_i and, for every sent packet where R_i is not included in the list of probed nodes, Eve happily causes failures. Eve stops causing failures whenever R_i is included in the list of probed nodes. Alice will never be able to localize such an Eve to a single link; as long as Eve behaves herself when she is part of the list of probed nodes, Alice has no way to find her. Our protocols avoid this problem by running their "detection phases" and "localization phases" on the same set of packets.

Furthermore, care must be taken in implementing this protocol in the presence of both adversarial behaviour and benign congestion. To see why, suppose that Eve causes the protocol to enter the localization phase. In [13], the binary search algorithm proceeds by one step each time failures are detected. It is important to ensure that normal congestion (on a link that is not adjacent to Eve) cannot cause the binary search algorithm to search for Eve in the wrong part of the path. To do this, the binary search algorithm should proceed by one step only when the *failure rate* exceeds some carefully chosen false alarm threshold (related to loss rate caused by normal congestion and the length of the portion of path that is currently being searched).

Packet Obituaries [10]: Argyraki, Maniatis, Cheriton, and Shenker propose an FL protocol that is similar to our Optimistic Protocol of Section 4.3.1. Each node locally stores digests of the packets they see, and at the end of some time interval, nodes send out reports to Alice that contain these packet digests. Alice then uses the information from these reports to localize failures on the path. The designers of this protocol focused on the benign setting, but mentioned that reports should also be *individually authenticated*. However, because these reports are not formed in a onion manner (as in our Optimistic Protocol) an adversarial node can implicate a innocent downstream node by selectively dropping the innocent node's reports.

Secure Traceroute [86]: At a very high level, Padmanabhan and Simon's FL protocol

uses an approach that is very similar to that of [13]; Alice runs a failure detection protocol with Bob until she detects that more than a β fraction of her packets have been dropped. Then, on *subsequent (new) sent packets*, Alice will run a failure localization protocol, where the intermediate nodes are required to send out acks that are used to localize failures. However, this protocol is vulnerable to the same adversary as [13]'s protocol: an Eve that causes failures when Alice runs failure detection, and then behaves herself once Alice turns on failure localization. The other issue with this protocol is that acks are *individually authenticated*, rather than onionized in the localization phase.

C.2 A Composition Technique for Statistical FL

We prove Lemma 4.3.3, Lemma 4.3.4, Lemma 4.3.6 and Lemma 4.3.8.

C.2.1 Proof of Lemma 4.3.3

Lemma C.2.1 (Restatement of Lemma 4.3.3)**.** *As long as* $T = O(\frac{K^2}{p(\beta-\alpha)^2} \ln \frac{K}{\delta})$*, then the estimators in the composition of SSS satisfy: for each* $i \notin E$ *where* E *is the set of nodes corrupted by Eve it holds (up to negligible error) that*

$$\Pr\left[\left|V_i - \tfrac{1}{T}(D_i + \tfrac{p'}{p}C_i)\right| > \tfrac{1}{4}\gamma\right] < \tfrac{\delta}{4(K+1)}$$

where $\gamma = \frac{\beta-\alpha}{2(K+1)}$*,* V_i *is* R_i*'s estimate of the failure rate between* i *and* $K+1$*,* D_i *is the number of data packets dropped between* R_i *and* R_{K+1}*, and* C_i *is the number of acks (destined for any node) dropped between* R_i *and* R_{K+1}*.*

Proof. Consider the random variable V_i' which is generated as V_i is generated in SSS, except that now we assume that instead using a pseudorandom function f_{k_1} to decide if a packet is a probe, as in equation (4.1), Alice and Bob instead use shared, truly random function ϕ_i. Now consider the statistical FL protocol composed of K instances of this "truly random version of SSS". In this statistical FL protocol, it follows that regardless of how Eve (or congestion) behaves,

- Every packet that Eve (or congestion) drops is a probe for each R_i with probability p independent of Eve's actions and each other R_j for $j \neq i$.
- Every ack that Eve (or congestion) drops or tampers with is a probe for each R_i with probability p' independent of Eve's actions and each other R_j for $j \neq i$. As we argued in the proof of Theorem 4.3.2, this is because acks intended for different nodes are indistinguishable (since they are all identically onion MAC'd, and they all originate at Bob), and since the acks are onion MAC'd, Eve cannot selectively tamper with the ack intended for an upstream node.

We will show now that this means V_i' is the average of many independent random variables. Let $S_{D_i}, S_{C_i} \subseteq [T]$ denote the set of exchanges that are data-faulty and ack-faulty, respectively for node R_i (we can order the exchanges in an arbitrary way), and notice that $S_{D_i} \cap S_{C_i} = \varnothing$ since an exchange cannot be both data- and ack-faulty. We now define the random variables X_t for

$t \in [T]$. For $t \notin S_{D_i} \cup S_{C_i}$, the variable X_t is identically 0. Otherwise, X_t is defined as follows:

$$\text{For } t \in S_{D_i},\ X_t = \begin{cases} 1 & \text{w.p. } p \\ 0 & \text{w.p. } 1-p \end{cases}$$

$$\text{For } t \in S_{C_i},\ X_t = \begin{cases} 1 & \text{w.p. } p' \\ 0 & \text{w.p. } 1-p' \end{cases}$$

We claim that $V_i' = \frac{1}{pT}\sum_{t=1}^{T} X_t$ because each data packet is unique and sampled exchanges are chosen using a truly random function, each exchange will be sampled by R_i with independent probability p. Thus each data-faulty exchange in S_{D_i} was sampled by R_i with probability p, while each ack-faulty exchange in S_{C_i} was sampled by R_i with probability p' (because here we need to condition on the fact that at least one node sampled the ack-faulty exchange (so that Bob generates an ack for that exchange!)). Thus, it follows that $\mathbb{E}[V_i'] = \frac{1}{T}(D_i + \frac{p'}{p}C_i)$. We now have

$$\begin{aligned} \Pr[|V_i' - \frac{1}{T}(D_i + \frac{p'}{p}C_i)| > \gamma] &= \Pr[|pV_i' - \frac{1}{T}(pD_i + p'C_i)| > p\gamma] \\ &= \Pr[\frac{1}{T}|\sum_t X_t - (pD_i + p'C_i)| > p\gamma] \\ &= \Pr[|\frac{1}{T}\sum_t X_t - \mu| > \frac{p\gamma}{\mu}\mu] \end{aligned} \tag{C.1}$$

where we let $\mu = \frac{1}{T}(pD_i + p'C_i)$. At this point, it would be nice if we could say that $\mu = O(p)$, which would allow us to derive the conclusion, but *a priori* we can only assume that $\mu = O(p')$. Instead, in the rest of this proof we shall carefully show that the probability that $C_i > 2\frac{p}{p'}T$ is at most $\frac{\delta}{8(K+1)}$, and conditioned on $C_i \leq 2\frac{p}{p'}T$ then we also have that the probability that $|\frac{1}{T}\sum_t X_t - \mu| > p\gamma$ is bounded by $\frac{\delta}{8(K+1)}$, which gives us an overall bound of $\frac{\delta}{4(K+1)}$.

We start by letting Y denote the number of exchanges in the game that require acks for any R_i. Notice that $\mathbb{E}[Y] = \frac{p}{p'}T$ and that $C_i \leq Y$ unconditionally, simply because one can't tamper with an ack if it was never sent. Now we split the probability in Equation (C.1) as follows:

$$\begin{aligned} \Pr[|\frac{1}{T}\sum_t X_t - \mu| > \frac{p\gamma}{\mu}\mu] &\leq \Pr[|\frac{1}{T}\sum_t X_t - \mu| > \frac{p\gamma}{\mu}\mu \text{ and } C_i > 2\frac{p}{p'}T] \\ &\quad + \Pr[|\frac{1}{T}\sum_t X_t - \mu| > \frac{p\gamma}{\mu}\mu \text{ and } C_i \leq 2\frac{p}{p'}T] \\ &\leq \Pr[C_i > 2\frac{p}{p'}T] + \Pr[|\frac{1}{T}\sum_t X_t - \mu| > \frac{p\gamma}{\mu}\mu \ \mid \ C_i \leq 2\frac{p}{p'}T] \\ &\leq \Pr[Y > 2\frac{p}{p'}T] + \Pr[|\frac{1}{T}\sum_t X_t - \mu| > \frac{p\gamma}{\mu}\mu \ \mid \ C_i \leq 2\frac{p}{p'}T] \end{aligned} \tag{C.2}$$

Now by a Chernoff bound the first probability is much less than $\frac{\delta}{8(K+1)}$ for our choice of $T = O(\frac{1}{\gamma^2 p}\ln\frac{K}{\delta})$, since Y is the sum of independent $\frac{p}{p'}$-biased random variables.

The second probability can now be bounded by a Chernoff bound. Notice that while the *definition* of the distribution of the X_t depends on C_i, the actual randomness of the X_t is independent of C_i. This gives us that the second probability is bounded by

$$\Pr[|\frac{1}{T}\sum_t X_t - \mu| > \frac{p\gamma}{\mu}\mu \ \mid \ C_i \leq 2\frac{p}{p'}T] \leq 2^{-\Omega(\frac{p^2\gamma^2}{\mu}T)} \tag{C.3}$$

In the above, we are interested in the event $|\frac{1}{T}\sum_t X_t - \mu| > \frac{p\gamma}{\mu}\mu$ conditioned on $C_i \leq 2\frac{p}{p'}T$, so that we can bound

$$\mu = \frac{1}{T}(pD_i + p'C_i) \leq \frac{1}{T}(pD_i + 2pT) \leq 3p$$

Plugging this into Inequality (C.3) and recalling from the statement of Theorem 4.3.2 that we set $T = O(\frac{1}{p\gamma^2}\ln(K/\delta))$, we get

$$\Pr[|\frac{1}{T}\sum_t X_t - \mu| > \frac{p\gamma}{\mu}\mu \;\mid\; C_i \leq 2pT] \leq 2^{-\Omega(p\gamma^2 T)} \leq \frac{\delta}{8(K+1)}$$

Combining this with the previous bound on $\Pr[Y > 2\frac{p}{p'}T]$ gives us that Inequality (C.2) becomes

$$\Pr[|V_i' - \frac{1}{T}(D_i + \frac{1}{p}C_i)| > \gamma] \leq \frac{\delta}{4(K+1)}$$

To complete the proof of the lemma, it suffices to observe that if replacing a truly random function ϕ with a PRF alters the probability by more than $K\varepsilon_{\mathsf{prf}}$, then we can efficiently distinguish between ϕ and the PRF f with advantage $\varepsilon_{\mathsf{prf}}$ by using the distinguisher that simply simulates this entire game, using access to an oracle containing either ϕ or f to answer calls to the PRF from the scheme, and then outputting 1 iff the condition $|V_i - \frac{1}{T}(D_i + \frac{p'}{p}C_i)| \leq \frac{1}{2}\gamma$ is violated.[1] □ □

C.2.2 Proof of Lemma 4.3.4

Lemma C.2.2 (Restatement of Lemma 4.3.4)**.** *As long as $T = O(\frac{K^2}{p(\beta-\alpha)^2}\ln\frac{K}{\delta})$, for each $i, i+1 \notin E$ where E is the set of nodes corrupted by Eve it holds (up to negligible error) that*

$$\Pr\left[\frac{p'}{p}\frac{C_i - C_{i+1}}{T} > \frac{\gamma}{2}\right] < \frac{\delta}{2(K+1)}$$

where $\gamma = \frac{\beta-\alpha}{2(K+1)}$ and where C_i is the number of acks (destined for any node) dropped between R_i and R_{K+1}.

Proof of Lemma 4.3.4. Fix C_{i+1}. Let $M \leq T$ be the number of exchanges in the interval for which a data packet reaches Bob, and a corresponding ack packet returns to R_{i+1}. Since R_i is honest, $C_{i+1} - C_i$ will just be the number of acks that are dropped due to congestion on link $(i, i+1)$, which occurs with probability ρ. Let X_i be a ρ-biased $\{0,1\}$ variable.

Let $U = \frac{\gamma}{2}(1-(1-p)^K)T$. We can derive:

$$\begin{aligned}
\Pr[\tfrac{p'}{p}\tfrac{C_i - C_{i+1}}{T} > \tfrac{\gamma}{2}] &= \Pr[C_i - C_{i+1} > U] \\
&\leq \Pr\left[\sum_{j=1}^{M} X_i > U\right] = \Pr\left[\tfrac{1}{M}\sum_{j=1}^{M} X_i - \rho > (\tfrac{U}{\rho M} - 1)\rho\right] \\
&\leq 2^{-\Omega(\rho(\frac{U}{\rho M})^2 M)} = 2^{-\Omega(U^2/(\rho M))}
\end{aligned}$$

[1] This distinguisher in fact requires access to K oracles, either all computing either a truly random function or all computing a PRF. Then we can turn this into a distinguisher for a single oracle using the hybrid argument, which is why we lose a factor of K in the distinguishing advantage. See *e.g.,* [50] for details about this kind of argument.

Because we have $\gamma = \frac{\beta-\alpha}{2K} \gg \rho$, this implies that

$$\frac{U}{\rho M} \geq \frac{\frac{\gamma}{2}(1-(1-p)^K)}{\rho} = \Omega(1)$$

so $\Pr[C_i - C_{i+1} > U] \leq 2^{-\Omega(U)} = 2^{-\frac{\gamma}{2}(1-(1-p)^N)T}$. Substituting our value of T gives us that this is bounded by less than $\delta/(2(K+1))$. □ □

C.2.3 Proof of Lemma 4.3.6

Lemma C.2.3 (Restatement of Lemma 4.3.6)**.** *Let* $\Gamma = \frac{T}{K+1}\frac{\beta(2\alpha+\beta)}{\alpha+2\beta}$ *and* $\varepsilon_i = \frac{1}{2i}\frac{\beta-\alpha}{2\beta+\alpha}$*. For every* $i \in [K]$*, assume that* R_i *computes an estimate* V_i *that* $(\varepsilon_i, \delta_i)$*-estimates* $\|\mathbf{x}_i\|_p^p$*. Suppose also that* $\|\mathbf{x}_i\|_p^p \leq \frac{\beta i}{K+1}$*. Then with probability at least* $1 - 2\delta'$ *it follows that:*

1. *If "link* $(i, i+1)$ *is good" so that* $\|\mathbf{x}_{i+1}\|_p^p - \|\mathbf{x}_i\|_p^p \leq \frac{\alpha}{K+1}T$ *then* $V_{i+1} - V_i \leq \Gamma$*.*
2. *If "link* $(i, i+1)$ *is bad" so that* $\|\mathbf{x}_{i+1}\|_p^p - \|\mathbf{x}_i\|_p^p \geq \frac{\beta}{K+1}T$ *then* $V_{i+1} - V_i \geq \Gamma$*.*

Proof. We prove each case separately.

Link $(i, i+1)$ **is good.** Since V_i $(\varepsilon_i, \delta_i)$-approximates $\|\mathbf{x}_i\|_p^p$, we can apply (4.4) to find, that with probability $1 - 2\delta'$,

$$\begin{aligned}
V_{i+1} - V_i &\leq (1+\varepsilon_{i+1})\|\mathbf{x}_{i+1}\|_p^p + (1-\varepsilon_i)\|\mathbf{x}_i\|_p^p \\
&\leq (1+\varepsilon_{i+1})(\|\mathbf{x}_{i+1}\|_p^p - \|\mathbf{x}_i\|_p^p) + (\varepsilon_{i+1}+\varepsilon_i)\|\mathbf{x}_i\|_p^p \\
&\leq (1+\varepsilon_{i+1})\tfrac{\alpha}{K+1}T + (\varepsilon_{i+1}+\varepsilon_i)\tfrac{i\beta}{K+1}T \\
&= \tfrac{\alpha}{K+1}T\left(1 + \varepsilon_{i+1}(1+\tfrac{\beta}{\alpha}i) + \varepsilon_i i\tfrac{\beta}{\alpha}\right) \\
&\leq \tfrac{\alpha}{K+1}T\left(1 + (i+1)\varepsilon_{i+1}(1+\tfrac{\beta}{\alpha}) + i\varepsilon_i(1+\tfrac{\beta}{\alpha})\right) \\
&= \tfrac{T}{K+1}\tfrac{\beta(2\alpha+\beta)}{\alpha+2\beta} = \Gamma
\end{aligned} \tag{C.4}$$

where we get the required inequality by putting $\varepsilon_i = \frac{1}{2i}\frac{\beta-\alpha}{2\beta+\alpha}$.

Link $(i, i+1)$ **is bad.** Again, we apply (4.4) to find, that with probability $1 - 2\delta'$,

$$\begin{aligned}
V_{i+1} - V_i &\geq (1-\varepsilon_{i+1})(\|\mathbf{x}_{i+1}\|_p^p - \|\mathbf{x}_i\|_p^p) - (\varepsilon_{i+1}+\varepsilon_i)\|\mathbf{x}_i\|_p^p \\
&\geq (1-\varepsilon_{i+1})\tfrac{\beta}{K+1}T - (\varepsilon_{i+1}+\varepsilon_i)\tfrac{i\beta}{K+1}T \\
&= \tfrac{T}{K+1}\tfrac{\beta(2\alpha+\beta)}{\alpha+2\beta} = \Gamma
\end{aligned} \tag{C.5}$$

where we again get the required inequality by putting $\varepsilon_i = \frac{1}{2i}\frac{\beta-\alpha}{2\beta+\alpha}$. □

C.2.4 Proof of Lemma 4.3.8

Our proof of Lemma 4.3.8 relies on the assumption that Eve occupies less than $\sqrt{K}$ links on the path.

To better understand why we made this assumption, suppose Eve occupies a large number of links on the path, and let node R_e be node occupied by Eve. Suppose Eve adds a small number $\leq \frac{\beta}{K+1}$ of nonsense packets to each link she occupies that is upstream of node R_e. If the number of added packets at each link is small, then there is a probability greater than δ that Alice will not raise alarm for these links. Next, at node R_e, Eve drops all the packets she added

before, and additionally causes γ-fraction of failures. As such, we have that V_e will be large (proportional to the number of nonsense packets added downstream), and V_{e+1} will be large as well (proportional to the γT failures at R_e). It follows that there is a probability greater than δ that $V_{e+1} - V_e$ will be small enough for Alice not to alarm, and so security fails. To rule out this attack, we limit the number of nodes occupied by Eve; this forces Eve to add a larger number of nonsense packets to the links upstream of R_e and increases the probability that Alice will raise an alarm for one of these links.

However, our proof only uses a simple averaging argument to claim that if Eve occupies M links, there must be a *single* link where $\gamma T \geq \frac{\beta}{M} T$, and uses this to arrive at the fact that Eve can only occupy $M \leq \sqrt{K}$ links on the path. However, we have not used the fact that Eve must cause a total of βT failures at *all* the links she occupies; we conjecture that using this fact could allow us to arrive at a weaker bound on M. We leave this to future work.

Lemma C.2.4 (Restatement of Lemma 4.3.8)**.** *If Eve occupies $M \leq \sqrt{(K+1)(1-\frac{\rho}{\beta}K^2)}$ links and causes a β-fraction of failures in the interval, then there must be a link $(i, i+1)$ that is adjacent to Eve with*

$$\|\mathbf{x}_{i+1}\|_p^p - \|\mathbf{x}_i\|_p^p \geq \tfrac{\beta}{K+1} T$$

Proof of Lemma 4.3.8. Since Eve occupies M links causes at least a β-fraction failures, it immediately follows that there exists a link $(i, i+1)$ adjacent to Eve where at least $\frac{\beta}{M}$-fraction of failures, *i.e.,* $D_{i+1} - D_i \geq \frac{\beta}{M}$. Now if the following holds

$$\|\mathbf{x}_{i+1}\|_p^p - \|\mathbf{x}_i\|_p^p > \tfrac{\beta}{K+1} T \tag{C.6}$$

we are done, since link $(i, i+1)$ is adjacent to Eve. Thus, suppose (C.6) do *not* hold. Then, applying identity (4.6), we have that

$$\begin{aligned} \tfrac{\beta}{K+1} T &\geq \|\mathbf{x}_{i+1}\|_p^p - \|\mathbf{x}_i\|_p^p \\ &= D_{i+1} - D_i + \|\mathbf{a}_{i+1}\|_p^p - \|\mathbf{a}_i\|_p^p \end{aligned}$$

rearranging and then using that fact that $D_{i+1} - D_i \geq \frac{\beta}{M}$ we get

$$\|\mathbf{a}_i\|_p^p \geq = \beta T(\tfrac{1}{M} - \tfrac{1}{K+1}) \tag{C.7}$$

Next, consider the next link $(j, j+1)$ that is occupied by Eve and is upstream of link $(i, i+1)$. Now again, if the following holds

$$\|\mathbf{x}_{j+1}\|_p^p - \|\mathbf{x}_j\|_p^p > \tfrac{\beta}{K+1} T \tag{C.8}$$

then we are done, since link $(j, j+1)$ is adjacent to Eve. So, we again suppose (C.8) does *not* hold. Since Eve does not occupy any links between R_{j+1} and R_i, and only congestion-related loss could have occurred on the links between R_{j+1} and R_i. It follows that $\|\mathbf{x}_{j+1}\|_p^p \geq \|\mathbf{x}_i\|_p^p + \rho(i-j-1)$. Since (C.8) does *not* hold, we can apply identity (4.6) and the fact that $\|\mathbf{x}_{j+1}\|_p^p \geq \|\mathbf{x}_i\|_p^p + \rho(i-j-1) \geq \|\mathbf{a}_i\|_p^p + \rho(i-j-1)$ and the bound on $\|\mathbf{a}_i\|_p^p$ in (C.7) to get

$$\|\mathbf{x}_j\|_p^p > \beta T \left(\tfrac{1}{M} - \tfrac{2}{K+1} - \tfrac{\rho}{\beta}(i-j-1) \right) \tag{C.9}$$

We continue this argument for all $m \leq M-1$ links that are adjacent to Eve and upstream of link $(i, i+1)$. Finally, arriving at the last such link, which we call link $(e, e+1)$, we have

$$\begin{aligned} \|\mathbf{x}_{e+1}\|_p^p &> \beta T \left(\tfrac{1}{M} - \tfrac{m}{K+1} - \tfrac{\rho}{\beta}(i-e-1) \right) \\ &> \beta T \left(\tfrac{1}{M} - \tfrac{M-1}{K+1} - \tfrac{\rho}{\beta} K \right) \end{aligned} \tag{C.10}$$

where the last inequality follows by putting $m \leq M-1$ and $i-e \leq K$. Now since by definition Eve does not occupy any links downstream of link $(e, e+1)$, we immediately have that $\|\mathbf{x}_e\|_p^p = 0$. It follows that link $(e, e+1)$ has

$$\|\mathbf{x}_{e+1}\|_p^p - \|\mathbf{x}_e\|_p^p > \beta T(\tfrac{1}{M} - \tfrac{M-1}{K+1} - \tfrac{\rho}{\beta}K) > \tfrac{\beta}{K+1} \tag{C.11}$$

where the last inequality follows because we put $M \leq \sqrt{(K+1)(1-\frac{\rho}{\beta}K^2)}$. This concludes the proof of this lemma, since link $(e, e+1)$ is adjacent to Eve. □

C.3 Lower Bounds

After introducing some notation and technical lemmata, we prove Lemma 4.4.4 and Lemma 4.4.5.

C.3.1 Technical Lemmata

Notation. For two random variables, X, Y, we denote their concatenation with either (X, Y) or with XY.

Statistical distance. Recall that we define the statistical distance between two random variables X, Y as $\Delta(X, Y) = \frac{1}{2}\sum_{x \in U} |\Pr[X = x] - \Pr[Y = x]|$ where U is the union of the supports of X and Y (for more background on statistical distance, see *e.g.,* [50]).

Lemma C.3.1. *For any random variables X, Y, Z, X' satisfying $X = \eta Y + (1-\eta)Z$ and $\Delta(X, X') \leq \varepsilon$, there exists random variables Y', Z' and $\eta' \in [\eta \pm \varepsilon]$ such that $X' = \eta' Y' + (1-\eta')Z'$ and $\Delta(Y, Y') \leq \frac{3\varepsilon}{2\eta}$.*

Proof. Define a randomized process F acting on the support of X, where for each $x \in \mathsf{supp}(X)$, $F(x) = 1$ with probability $p(x) = \frac{\eta \Pr[Y=x]}{\Pr[X=x]}$ and $F(x) = 0$ with probability $1 - p(x)$, and say $F(x) = 0$ for all $x \notin \mathsf{supp}(X)$. We can check that

$$\Pr[F(X) = 1] = \mathbb{E}[F(X)] = \sum_{x \in \mathsf{supp}(X)} \Pr[X = x] \tfrac{\eta \Pr[Y=x]}{\Pr[X=x]} = \sum_{x \in \mathsf{supp}(X)} \eta \Pr[Y = x] = \eta$$

and similarly $F(X) = 0$ with probability $1-\eta$. Furthermore, we claim that $Y = (X \mid F(X) = 1)$ since for every x,

$$\Pr[X = x \mid F(X) = 1] = \tfrac{\Pr[F(X)=1 \wedge X=x]}{\Pr[F(X)=1]} = \tfrac{\Pr[F(x)=1]\Pr[X=x]}{\eta} = \tfrac{\eta \Pr[Y=x]}{\Pr[X=x]} \tfrac{\Pr[X=x]}{\eta} \Pr[Y = x]$$

and similarly $Z = (X \mid F(X) = 0)$.

Since $\Delta(X, X') \leq \varepsilon$, this means that $\Pr[F(X') = 1] = \eta'$ for $\eta' \in [\eta \pm \varepsilon]$, and also $\Delta((F(X), X), (F(X'), X')) \leq \varepsilon$. Define $Y' = (X' \mid F(X') = 1)$ and $Z' = (X' \mid F(X') = 0)$. We may derive:

$$\begin{aligned} \varepsilon &\geq \Delta((F(X), X), (F(X'), X')) \\ &= \Delta(\eta(1, Y) + (1-\eta)(0, Z), \eta'(1, Y') + (1-\eta')(0, Z')) \end{aligned}$$

Viewing the random variables as the characteristic vectors of their distributions, and using the ℓ_1 formulation of statistical distance, we have:

$$= \tfrac{1}{2}\|\eta(1, Y) + (1-\eta)(0, Z) - \eta'(1, Y') - (1-\eta')(0, Z')\|_1$$

Since coordinates of the form $(1, Y)$ are disjoint from coordinates of the form $(0, Z)$, we have the equality:

$$\begin{aligned}
&= \tfrac{1}{2}\|\eta(1,Y) - \eta'(1,Y')\|_1 + \tfrac{1}{2}\|(1-\eta)(0,Z) - (1-\eta')(0,Z')\|_1 \\
&\geq \tfrac{1}{2}\|\eta(1,Y) - \eta'(1,Y')\|_1 \\
&= \tfrac{1}{2}\|\eta Y - \eta' Y' - (\eta' - \eta)\|_1 \\
&\geq \tfrac{1}{2}\eta\|Y - Y'\|_1 - \tfrac{1}{2}|\eta' - \eta| \\
&\geq \eta\Delta(Y, Y') - \varepsilon/2
\end{aligned}$$

which, rearranged, gives us that $\Delta(Y, Y') \leq \frac{3\varepsilon}{2\eta}$. □

Lemma C.3.2. *Let X, Y, X', Y' be such that $\Delta(XY, X'Y') \leq \varepsilon$. Say that $x \in \mathsf{supp}(X) \cap \mathsf{supp}(X')$ is δ-*bad *if $\Delta(Y(x), Y'(x)) > \delta$, where $Y(x)$ denotes the conditional distribution $Y \mid X = x$ and $Y'(x)$ denotes $Y' \mid X' = x$. Then $\Pr[X$ is δ-bad$] \leq 2\varepsilon/\delta$.*

Proof. Our proof is by contradiction. Suppose $\Pr[X \text{ is } \delta\text{-bad}] > 2\varepsilon/\delta$. Then, use the triangle inequality to obtain:

$$\Delta(XY, X'Y') \geq \Delta(XY, XY'(X)) - \Delta(XY'(X), X'Y')$$

where the random variable $Y(X')$ denotes $Y(x) = y \mid x \leftarrow_{\mathrm{R}} X'$. By hypothesis, we know that $\Delta(X, X') \leq \varepsilon$ so we have

$$\begin{aligned}
&\geq \Delta(XY, XY'(X)) - \varepsilon \\
&= \Delta(Y, Y'(X)) - \varepsilon \\
&\geq \Pr[X \text{ is } \delta\text{-bad}]\Delta(Y \mid X\text{bad}, Y'(X) \mid X\text{bad}) - \varepsilon
\end{aligned}$$

and since the statistical distance between $Y(x)$ and $Y'(x)$ when x is δ-bad is at least δ, we have

$$> (2\varepsilon/\delta) \cdot \delta - \varepsilon \geq \varepsilon$$

which contradicts the hypothesis that $\Delta(X, X') \leq \varepsilon$. □

Lemma C.3.3. *Let X, Y, X', Y' be random variables where $\Delta(X, X') \leq \varepsilon_1$. We say that $x \in \mathsf{supp}(X) \cap \mathsf{supp}(X')$ is ε_2-*bad *if $\Delta(Y(x), Y'(x)) \geq \varepsilon_2$, and suppose $\Pr[X$ ε_2-bad$] \leq \varepsilon_3$. Then $\Delta(XY, X'Y')) \leq \varepsilon_1 + \varepsilon_2 + \varepsilon_3$.*

Proof. This follows from the triangle inequality:

$$\begin{aligned}
\Delta(XY, X'Y') &\leq \Delta(XY, XY'(X)) + \Delta(XY'(X), X'Y') \\
&\leq \Delta(Y, Y'(X)) + \Delta(X, X') \\
&\leq \Delta(Y, Y'(X)) + \varepsilon_1 \\
&\leq \Pr[X \ \varepsilon_2\text{-bad}] \cdot \Delta(Y \mid X \text{ bad},\ Y'(X) \mid X \text{ bad}) \\
&\quad + (1 - \Pr[X \ \varepsilon_2\text{-bad}]) \cdot \Delta(Y \mid X \text{ not bad},\ Y'(X) \mid X \text{ not bad}) + \varepsilon_1 \\
&\leq \varepsilon_3 \cdot 1 + (1 - 0) \cdot \varepsilon_2 + \varepsilon_1
\end{aligned}$$

□

C.3.2 Proof of Lemma 4.4.4

First a word about random oracles, which we treat as a function $\mathsf{RO} : \{0,1\}^* \to \{0,1\}$. We look at the RO using the "lazy evaluation" methodology: points in RO are not fixed until they have been queried. When an efficient algorithm executes with a random oracle, it can make only an efficient number of queries. This means that RO can be viewed as a polynomially long string representing the responses to the algorithm, rather than as an infinitely large function, and replacing RO by a different string RO' (of equal length) amounts to replacing the real random oracle with a "fake random oracle". Thus, in the following, when we say that an oracle outputs a fake random oracle consistent with the output h of an algorithm A, we mean it outputs a string $\mathsf{RO}' \in \{0,1\}^{\mathrm{poly}(n)}$ such that running A with the responses encoded in RO' generates h.

Learning Algorithm. We apply Naor and Rothblum's [83] learning algorithm for *adaptively changing distributions (ACD)*. The ACD we work with is defined as a pair (Init, D) of random processes, where Init is a key generation process that takes a uniform string s and generates secrets $\overrightarrow{\mathsf{aux}} = \mathsf{Init}(s)$, and D is a process that takes the initial state $\overrightarrow{\mathsf{aux}}$ and a history H_{i-1} and uses them to generate a sample τ_i of an exchange. The history H_{i-1} consists of tuples (r_j, τ_j) for all $j \leq i-1$, where r_j was the random string used to generate the transcript τ_j. Notice that the τ_j are the outputs of the ACD, while the initial state s and the r_j remain secret.

Theorem C.3.4 (Naor and Rothblum [83])**.** *There exists a* PSPACE *algorithm that, for any ACD* (Init, D)*, observes at most* $t = O(n/\varepsilon^4)$ *samples from* D *and generates with probability* $> 1-\varepsilon$ *a fake secret state* $\overrightarrow{\mathsf{aux}}'$ *and fake history* H'_t *such that simulating* D *with* $\overrightarrow{\mathsf{aux}}', H'_t$ *generates a sample* tau'_{t+1} *that is distributed* ε*-statistically close to an honest sample generated by* D *using* $\overrightarrow{\mathsf{aux}}, H_t$*.*

Lemma C.3.5 (Lemma 4.4.4 restated)**.** *Relative to a* $(\mathsf{PSPACE}, \mathsf{RO})$*-oracle, there exists an efficient algorithm that observes at most* $t = O(\frac{n}{\varepsilon^4})$ *honest* exchanges $\tau_j = \langle R_{i-1}, R_i \rangle_j$ *and then, with probability* $> 1-\varepsilon$*, outputs algorithms* $R'_0, \ldots, R'_{K+1}$ *such that a fake exchange* $\langle R'_{i-1}, R'_i \rangle_{t+1}$ *is distributed* ε*-close to an honest exchange* $\langle R_{i-1}, R_i \rangle_{t+1}$*.*

Proof of Lemma 4.4.4. Apply Theorem C.3.4 where the Init function is our key generation function, and D is the algorithm that simulates the interaction of all algorithms $R_0, \ldots, R_{K+1}$ given a uniformly random data packet to be sent, including simulating all the congestion along links between the nodes, and outputs the transcript along link $(i-1, i)$. To generate the transcript of the i'th exchange, D takes input $\overrightarrow{\mathsf{aux}}, H_{i-1}, \mathsf{RO}_i, r_i$ where RO_i are responses to new queries to the random oracle that D makes in generating the transcript, r_i is the fresh internal randomness used to generate the $i+1$'th transcript, and $H_{i-1} = (\tau_j, \mathsf{RO}_j, r_j)_{j \leq i-1}$ is a history of previous transcripts, responses of the random oracle, and internal randomness. Notice that because D simulates *all* the nodes, there is *no distinction* between how the learning algorithm treats RO_i and r_i.

After observing $t = O(n/\varepsilon^4)$ exchanges, using the learning algorithm of Theorem C.3.4, we get with probability $> 1-\varepsilon$ fake secrets $\overrightarrow{\mathsf{aux}}', H'_t$ consistent with the transcripts and such that generating the $t+1$'th transcript using the fake secrets is ε to generating the $t+1$'th transcript using the honest secrets. Set R'_i to be R_i but with the secrets in $\overrightarrow{\mathsf{aux}}', H'_t$ hardwired into the algorithm.

Efficiency is clear because we allow a PSPACE oracle and because the number of samples is $O(n/\varepsilon^4)$. □

C.3.3 Proof of Lemma 4.4.5

Lemma C.3.6 (Lemma 4.4.5 restated)**.** *Suppose that* $\Delta(\langle A, B\rangle, \langle A', B'\rangle) \leq \varepsilon$*, and there exist events* $E_1, \ldots, E_r$ *over the internal randomness of* A, B *such that (1)* $\forall j$*, conditioned on* E_j*, the first* j *messages from* B *to* A *are independent of* A*'s internal randomness, and (2)* $\Pr[E_j \mid E_{j-1}] \geq \rho$*. Set* $\varepsilon = (\rho/10)^{4r}$*, Then there exist* $\eta \geq (\rho/2)^r$*, and distributions over the transcripts* Y, Z *such that* $\langle A, B'\rangle$ *is a convex combination* $\eta Y + (1-\eta)Z$ *and*

$$\Delta((Y, \mathsf{view}_A(Y)), (\langle A, B\rangle, \mathsf{view}_A(\langle A, B\rangle) \mid E_r)) \leq 1/100$$

Proof. We define $\sigma_i = \langle A, B\rangle^i$, the first i messages in the partial transcript of $\langle A, B\rangle$. Notice that σ_i is a random variable so it makes sense to condition σ_i on the event E_i. Similarly define $\sigma'_i = \langle A', B'\rangle^i$ and $\sigma_i^{\mathsf{alt}} = \langle A, B'\rangle^i$. We will decompose A, B, A', B' into next-message functions A_i, B_i, A'_i, B'_i for $1 \leq i \leq r$, where we assume that each party takes turns communicating, and $2r$ is the maximum number of rounds of communication. Recall from the statement of the lemma that we think of applying the i'th next message function $B'_i(\tau)$ to a partial transcript τ of $2i-1$ messages as sampling from $(\langle A', B'\rangle^{2i} \mid \langle A', B'\rangle^{2i-1} = \tau)$.

Recall that $\mathsf{view}_A(\tau)$ is the distribution of the internal randomness of A conditioned on outputting the transcript τ.

Now, we define the conditional view $\mathsf{condview}_A(\tau)$ to be a *uniformly sampled view* of all the possible shared and independent internal randomness for A that causes A to output its messages in τ, such that the shared randomness is consistent with B output its messages in τ. Intuitively, we think of $\mathsf{condview}_A(\tau)$ and randomness for A sampled under the assumption that parties A, B that created τ correctly shared their randomness.

Define alternating view $\mathsf{altview}_A(\tau)$ to be a uniformly sampled view of the all possible shared and independent internal randomness for A that causes A to output its messages in τ, *even if the shared randomness would not result in* B *outputting its messages in* τ. We can think of $\mathsf{altview}_A(\tau)$ and randomness for A sampled under the assumption that B and A use independent randomness, *i.e.,* that party B incorrectly shares its randomness with A.

Notice that when the parties that created τ are independent, the $\mathsf{condview}$ and $\mathsf{altview}$ are the same. Otherwise, the support of the $\mathsf{condview}$ is a subset of the support of the $\mathsf{altview}$. As such, we can list some properties of $\mathsf{condview}, \mathsf{altview}, \mathsf{view}$. Before we begin, recall that A, B share randomness, but B' impersonates to A, the randomness that B' is supposed to share with A is independent of A's randomness.

1. $\mathsf{condview}_A(\langle A, B\rangle) = \mathsf{view}_A(\langle A, B\rangle)$ because in this case both parties are correctly sharing randomness.

2. $\mathsf{altview}_A(\langle A, B'\rangle) = \mathsf{view}_A(\langle A, B'\rangle)$ because when B' is impersonating to A, B' is not correctly sharing randomness with A. Thus the view of A is independent of the randomness of B'.

3. $\mathsf{condview}_A(\langle A, B'\rangle) \neq \mathsf{view}_A(\langle A, B'\rangle)$ because $\mathsf{condview}_A$ assumes that A, B' correctly share randomness, but when B' impersonates to A this is not the case.

4. $\mathsf{altview}_A(\langle A, B'\rangle^{2i}) = \mathsf{altview}_A(\langle A, B'\rangle^{2i-1})$ since B' computes the $2i$'th message, this does not affect A's $\mathsf{altview}$ since $\mathsf{altview}_A$ is independent of the randomness used by B'.

5. Conditioned on E_i, $\mathsf{condview}_A(\langle A, B\rangle^{2i}) = \mathsf{condview}_A(\langle A, B\rangle^{2i-1})$ because E_i tells us that the $2i$'th message (computed by B) is independent of A's randomness. It follows that the $2i$'th message will not affect A's $\mathsf{condview}$.

6. Conditioned on E_i, $\mathsf{condview}_A(\langle A,B\rangle^{2i}) = \mathsf{altview}_A(\langle A,B\rangle^{2i})$ and $\mathsf{condview}_A(\langle A,B\rangle^{2i+1}) = \mathsf{altview}_A(\langle A,B\rangle^{2i+1})$. This follows because messages $2,4,...,2i$ are computed by B and messages $1,3,...,2i+1$ are computed by A, and E_i tells us that all the messages from B to A are independent of A's randomness.

The proof of this Lemma 4.4.5 rests on the following claim, which will prove by induction:

Claim C.3.7. *Assuming $\varepsilon = (\rho/10)^{4r}$, for each i, $0 \le i \le r$, there exist $\eta_i \ge \rho/2$ and random variables Y_i, Z_i such that*

$$\sigma_{2i}^{\mathsf{alt}} = \prod_{j=1}^{i} \eta_j Y_i + (1 - \prod_{j=1}^{i} \eta_j) Z_i$$

and, for $\delta_i = \sqrt{\varepsilon}(10/\rho)^i$,

$$\Delta((\sigma_{2i}, \mathsf{condview}_A(\sigma_{2i}) \mid E_i), (Y_i, \mathsf{altview}_A(Y_i))) \le \delta_i$$

Apply this claim for the case of $\sigma_{2r} = \langle A,B\rangle$ and $\sigma_{2r}^{\mathsf{alt}} = \langle A,B'\rangle$ to obtain $Y = Y_r$ and $\eta = \prod_{j=}^{r} \eta \ge (\rho/2)^r$. This implies that we have the decomposition $\sigma_{2r}^{\mathsf{alt}} = \eta Y + (1-\eta)Z$ Next, we argued above (in the first item) that $\mathsf{condview}_A(\sigma_{2r}) = \mathsf{view}_A(\sigma_{2r})$. We argued above (in the second item) that $\mathsf{altview}_A(\sigma_{2r}^{\mathsf{alt}}) = \mathsf{view}_A(\sigma_{2r}^{\mathsf{alt}})$, and the decomposition of $\sigma_{2r}^{\mathsf{alt}} = \eta Y + (1-\eta)Z$ then gives that that $\mathsf{altview}_A(Y) = \mathsf{view}_A(Y)$. Finally, we can apply the claim to obtain that

$$\Delta((Y, \mathsf{view}_A(Y)), (\sigma_{2r}, \mathsf{view}_A(\sigma_{2r}) \mid E_r)) \le \sqrt{\varepsilon}(10/\rho)^r$$

which proves the lemma since $\sqrt{\varepsilon}(10/\rho)^r = (\rho/10)^r \le 100$. □

We now prove Claim C.3.7.

Proof of Claim C.3.7. Our proof is by induction. The base case $i = 0$ is trivial. The inductive hypothesis for $i-1$ is as follows: There exists $\eta_{i-1} \ge \rho/2$, $\delta_i = \sqrt{\varepsilon}(10/\rho)^i$ and random variables Y_{i-1}, Z_{i-1} such that

$$\Delta((\sigma_{2i-2}, \mathsf{condview}_A(\sigma_{2i-2}) \mid E_{i-1}), (Y_{i-1}, \mathsf{altview}_A(Y_{i-1})) \le \delta_{i-1} \tag{C.12}$$

and

$$\sigma_{2i-2}^{\mathsf{alt}} = \prod_{j=1}^{i-1} \eta_j Y_{i-1} + (1 - \prod_{j=1}^{i-1} \eta_j) Z_{i-1} \tag{C.13}$$

Our claim will follow if we show that this is also the case for i.

We are ready to start proving the inductive step. First, we apply A_i to both terms in Inequality C.12 and update the view to get (because this process is identical in both cases):

$$\Delta((\sigma_{2i-1}, \mathsf{condview}_A(\sigma_{2i-1}) \mid E_{i-1}), (\zeta_{2i-1}, \mathsf{altview}_A(\zeta_{2i-1}))) \le \delta_{i-1} \tag{C.14}$$

where for compactness we have set $\zeta_{2i-1} = Y_{i-1}A_i(Y_{i-1}, \mathsf{altview}_A(Y_{i-1}))$.

Applying Lemma C.3.2. Suppose that $\langle A,B\rangle$, $\langle A',B'\rangle$ that are statistically close, and consider a partial transcript σ_{2i-1} generated by A,B. Informally, we would like to show that it is extremely unlikely that the next message functions of B applied to σ_{2i-1} generates a transcript that is statistically far from the transcript generated by the next-message function of B' applied σ_{2i-1}. Formally, we do this by applying Lemma C.3.2, with

$$X = (\sigma_{2i-1}, \mathsf{condview}_A(\sigma_{2i-1})), \qquad Y = B_i(\sigma_{2i-1}, \mathsf{condview}_A(\sigma_{2i-1}))$$
$$X' = (\sigma'_{2i-1}, \mathsf{condview}_A(\sigma'_{2i-1})), \qquad Y' = B'_i(\sigma'_{2i-1})$$

Notice that that $(\sigma_{2i}, \mathsf{condview}_A(\sigma_{2i})) = XY$ and $(\sigma'_{2i}, \mathsf{condview}_A(\sigma'_{2i})) = X'Y'$, and we have $\forall i$,

$$\Delta(XY, X'Y') = \Delta((\sigma_{2i}, \mathsf{condview}_A(\sigma_{2i})), (\sigma'_{2i}, \mathsf{condview}_A(\sigma'_{2i}))) \leq \Delta(\sigma_{2i}, \sigma'_{2i}) \leq \Delta(\langle A, B\rangle, \langle A', B'\rangle) \leq \varepsilon$$

where the last inequality follows from the hypothesis in Lemma 4.4.5. Next, we say that a fixed transcript and view $x = (\tau_{2i-1},\ \mathsf{condview}_A(\tau_{2i-1}))$ is $2\sqrt{\varepsilon}$-bad if

$$\Delta(Y(x), Y'(x)) = \Delta(B_i(\tau_{2i-1}, \mathsf{condview}_A(\tau_{2i-1})),\quad B'_i(\tau_{2i-1})) > 2\sqrt{\varepsilon}$$

We can now apply Lemma C.3.2 to find that the probability that $(\sigma_{2i-1}, \mathsf{condview}_A(\sigma_{2i-1}))$ is $2\sqrt{\varepsilon}$-bad is at most $\frac{2\varepsilon}{2\sqrt{\varepsilon}} = \sqrt{\varepsilon}$. Before we move on, also observe that by the hypothesis in Lemma 4.4.5 we know that $\Pr[E_{i-1}] = \rho^{i-1}$ so that

$$\Pr[(\sigma_{2i-1}, \mathsf{condview}_A(\sigma_{2i-1})) \text{ is } 2\sqrt{\varepsilon} \text{ bad} \mid E_{i-1}] \leq \sqrt{\varepsilon}/\rho^{i-1} \tag{C.15}$$

Applying Lemma C.3.3. Informally, we want to argue that, if $(\sigma_{2i-1} \mid E_{i-1})$ and ζ_{2i-1} along with their views are statistically close (Inequality C.14), and if $(\sigma_{2i-1} \mid E_{i-1})$ is rarely bad (Inequality C.15) it follows that transcripts $(\sigma_{2i} \mid E_{i-1})$ and ζ_{2i} along with their views are also statistically close. We will do this using Lemma C.3.3, setting

$$\begin{aligned} X &= (\sigma_{2i-1}, \mathsf{condview}_A(\sigma_{2i-1}) \mid E_{i-1}), & Y &= B_i(\sigma_{2i-1}, \mathsf{condview}_A(\sigma_{2i-1})) \mid E_{i-1} \\ X' &= (\zeta_{2i-1}, \mathsf{altview}_A(\zeta_{2i-1})), & Y' &= B'_i(\zeta_{2i-1}) \end{aligned}$$

Notice that Inequality C.14 tells us that $\Delta(X, X') \leq \delta_{i-1}$. Furthermore, we have that

$$XY = (\sigma_{2i}, \mathsf{condview}_A(\sigma_{2i-1}) \mid E_{i-1})$$

and setting $\zeta_{2i} = \zeta_{2i-1}B'_i(\zeta_{2i-1})$ we have that

$$X'Y' = (\zeta_{2i}, \mathsf{altview}_A(\zeta_{2i-1}))$$

Furthermore, from Inequality C.15 it follows that $x = (\sigma_{2i-1}, \mathsf{condview}_A(\sigma_{2i-1}) \mid E_{i-1})$ is $2\sqrt{\varepsilon}$-bad (*i.e.,* $\Delta(Y(x), Y'(x)) \geq 2\sqrt{\varepsilon}$) with probability at most $\sqrt{\varepsilon}/\rho^{i-1}$. Thus, we can apply Lemma C.3.3 to obtain

$$\Delta(XY, X'Y') = \Delta((\sigma_{2i}, \mathsf{condview}_A(\sigma_{2i-1}) \mid E_{i-1}), (\zeta_{2i}, \mathsf{altview}_A(\zeta_{2i-1}))) \leq \delta_{i-1} + 2\sqrt{\varepsilon} + \sqrt{\varepsilon}/\rho^{i-1}$$

Before moving on, notice that this immediately implies that

$$\Delta((\sigma_{2i} \mid E_{i-1}), \zeta_{2i}) \leq \delta_{i-1} + 2\sqrt{\varepsilon} + \sqrt{\varepsilon}/\rho^{i-1} \doteq \gamma \tag{C.16}$$

Applying Lemma C.3.1: So far, we have been conditioning on E_{i-1}. We now condition on E_i. We want to say that because $(\sigma_{2i} \mid E_{i-1})$ and ζ_{2i} are close, and because $(E_i \mid E_{i-1})$ happens often, we can decompose ζ_{2i} so that part of it is close to $(\sigma_{2i} \mid E_i)$.

We shall do this using Lemma C.3.1. Set $X = (\sigma_{2i} \mid E_{i-1})$ and let $Y = X \mid E_i$ while $Z = X \mid \neg E_i$. By hypothesis in Lemma 4.4.5 the conditional event $(E_i \mid E_{i-1})$ occurs with probability ρ, so that $(\sigma_{2i} \mid E_{i-1}) = X = \rho Y + (1-\rho)Z$. From Inequality C.16 we know that $\Delta(X, \zeta_{2i}) = \gamma$. We can now apply Lemma C.3.1 to find that there exist $\eta_i \in [\rho \pm \gamma]$ and random variables Y_i, Z_i such that

$$\zeta_{2i} = \eta_i Y_i + (1-\eta_i)Z_i \tag{C.17}$$

and

$$\Delta((\sigma_{2i} \mid E_i), Y_i) \leq \tfrac{3\gamma}{2\eta_i} \tag{C.18}$$

Setting $\delta_{i-1} = \sqrt{\varepsilon}(10/\rho)^{i-1}$ and assuming $\varepsilon = (\rho/10)^{4r}$, since $n_i \geq \rho - \gamma$ implies that $\eta_i \geq \rho/2$, and Inequality C.18 is bounded by $\delta_i = \sqrt{\varepsilon}(10/\rho)^i$. Applying $\mathsf{altview}_A$ to both terms in Inequality C.18 we get

$$\Delta((\sigma_{2i}, \mathsf{altview}_A(\sigma_{2i}) \mid E_i), (Y_i, \mathsf{altview}_A(Y_i))) \leq \sqrt{\varepsilon}(10/\rho)^i$$

and then applying the fact that $\mathsf{condview}_A(\sigma_{2i}) = \mathsf{altview}_A(\sigma_{2i})$ conditioned on E_i, we get

$$\Delta((\sigma_{2i}, \mathsf{condview}_A(\sigma_{2i}) \mid E_i), (Y_i, \mathsf{altview}_A(Y_i))) \leq \sqrt{\varepsilon}(10/\rho)^i \tag{C.19}$$

which proves the first equation in our induction step (corresponding to Inequality C.12).

Finally, we finish by proving the part of our induction step corresponding to Equation C.13. We can derive

$$\sigma_{2i}^{\mathsf{alt}} = \sigma_{2i-2}^{\mathsf{alt}} A_{i-1}(\sigma_{2i-2}^{\mathsf{alt}}) B'_{i-1}(\sigma_{2i-2}^{\mathsf{alt}} A_{i-1}(\sigma_{2i-2}^{\mathsf{alt}}))$$

and using Equation C.13, we get

$$\begin{aligned}
&= \prod_{j=1}^{i-1} \eta_j Y_{i-1} A_{i-1}(Y_{i-1}) B'_{i-1}(Y_{i-1} A_{i-1}(Y_{i-1})) + (1 - \prod_{j=1}^{i-1} \eta_j) \dots \\
&= \prod_{j=1}^{i-1} \eta_j \zeta_{2i} + (1 - \prod_{j=1}^{i-1} \eta_j) \dots
\end{aligned}$$

now we apply Equation C.17 to get

$$\begin{aligned}
&= \prod_{j=1}^{i} \eta_j Y_i + \prod_{j=1}^{i-1} \eta_j (1 - \eta_i) Z_i + (1 - \prod_{j=1}^{i-1} \eta_j) \dots \\
&= \prod_{j=1}^{i} \eta_j Y_i + (1 - \prod_{j=1}^{i} \eta_j) W_i
\end{aligned} \tag{C.20}$$

where we used "..." to represent the rest of the convex combination of the Y_j, Z_j's which we finally collected in a new variable W_i[2]. Recalling that $\eta_i \geq \rho/2$ for all i, we have that $\prod_{j=1}^{i} \eta_j \geq (\rho/2)^i = 1/\mathrm{poly}(n)$.

Thus, combining Inequality C.19 with Equation C.20 completes the proof of our induction step. □

[2]We can do this because Z_i is from the decomposition of ζ_{2i} while W_i corresponds to the decomposition of $\sigma_{2i}^{\mathsf{alt}}$.