INTERNET LOCATION VERIFICATION: CHALLENGES AND SOLUTIONS

By

AbdelRahman Mohamed Abdou

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

Department of Systems and Computer Engineering

Carleton University

Ottawa, Ontario, Canada

Abstract

This thesis addresses the problem of verifying the geographic locations of Internet clients. First, we demonstrate how current state-of-the-art delay-based geolocation techniques are susceptible to evasion through delay manipulations, which involve both increasing and decreasing the Internet delays that are observed between a client and a remote measuring party. We find that delay-based techniques generally lack appropriate mechanisms to measure delays in an integrity-preserving manner. We then discuss different strategies enabling an adversary to benefit from being able to manipulate the delays. Upon analyzing the effect of these strategies on three representative delay-based techniques, we found that the strategies combined with the ability of full delay manipulation can allow an adversary to (fraudulently) control the location returned by those geolocation techniques accurately.

We then propose Client Presence Verification (CPV) as a delay-based technique to verify an assertion about a client's physical presence in a prescribed geographic region. Three verifiers geographically encapsulating a client's asserted location are used to corroborate that assertion by measuring the delays between themselves and the client. CPV infers geographic distances from these delays and thus, using the smaller of the forward and reverse one-way delay between each verifier and the client is expected to result in a more accurate distance inference than using the conventional round-trip times. Accordingly, we devise a novel protocol for accurate one-way delay measurements between the client and the three verifiers to be used by CPV, taking into account that the client could manipulate the measurements to defeat the verification process.

We evaluate CPV through extensive real-world experiments with legitimate clients (those truly present at where they asserted to be) modeled to use both wired and wireless access networks. Wired evaluation is done using the PlanetLab testbed, during which we examine various factors affecting CPV's efficacy, such as the client's geographical nearness to the verifiers. For wireless evaluation, we leverage the Internet delay information collected for wired clients from PlanetLab, and model additional delays representing the last-mile wireless link. The additional delays were generated following wireless delay distribution models studied in the literature. Again, we examine various factors that affect CPV's efficacy, including the number of devices actively competing for the wireless media in the vicinity of a wireless legitimate CPV client.

Finally, we reinforce CPV against a (hypothetical) middlebox that an adversary specifically customizes to defeat CPV (i.e., assuming an adversary that is aware of how CPV operates). We postulate that public middlebox service providers (e.g., in the form of Virtual Private Networks) would be motivated to defeat CPV if it is to be widely adopted in practice. To that end, we propose to use a Proof-of-Work mechanism that allows CPV to impose constraints, which effectively limit the number of clients (now adversaries) simultaneously colluding with that middlebox; beyond that number, CPV detects the middlebox.

Acknowledgements

Thanks to my supervisors, Dr. Ashraf Matrawy and Dr. Paul Van Oorschot, for their diligent efforts and close mentoring. Both were continuously keen to maintain high quality research outcomes throughout my PhD journey. Their vast knowledge and continuous guidance have helped shape the contributions in this thesis.

I would also like to thank my great wife Hala, who is not only my life partner but has also been a PhD colleague and a supporting friend. She has helped ameliorate many of the figures in this thesis, and was eager to proofread my papers even during her tight schedules. To my parents, Mohamed and Hayam, and my sisters, thanks for your tremendous moral and spiritual support throughout this tough journey. Your regular long distance calls were necessary to maintain my stamina, especially during phases of frustration.

I thank my PhD committee members, Urs Hengartner, Thomas Kunz, Michel Barbeau and Jiying Zhao, for their insightful comments. Thanks also to Pat Morin for his help with standard geometry, and David Lie for providing infrastructure for remote small-scale preliminary testing. I am grateful to the anonymous reviewers, and members of the networking/security community who generously gave their opinions, e.g., during casual discussions in conferences. Those include Michael Freedman, James Muir, Bill Aiello and Andrew Csinger.

Finally, I would like to express my gratitude to members of the Carleton Computer Security Lab (CCSL), especially David Barrera and Furkan Alaca, for hours of proofreading and discussions.

Glossary

ACK Acknowledgement

API Application Programming Interface

AS Autonomous System

BSD Berkeley Software Distribution

CBG Constraint-Based Geolocation

CDF Cumulative Distribution Function

CDN Content Distribution Network

CPV Client Presence Verification

CTS Clear To Send

CSMA/CA Carrier Sense Multiple Access with Collision Avoidance

DB Database

DCF Distributed Coordination Function

DIFS Distributed (coordination function) Interframe Space

DNS Domain Name Server

DSCP Differentiated Services Code Point

ECN Explicit Congestion Notification

FA False Accept

FR False Reject

GNU GNU's not Unix

GPS Global Positioning System

GREN Global Research and Educational Network

HTTP Hypertext Transfer Protocol

ICMP Internet Control Message Protocol

IHL Internet Header Length

IP Internet Protocol

ISP Internet Service Provider

LBS Location-Based Service

LSP Location-Sensitive Provider

MAC Medium Access Control

MB Middle Box

MitM Man in the Middle

NCS Network Coordination System

NFC Near Field Communication

NIC Network Interface Card

NRMSD Normalized Root-Mean-Square-Deviation

NTP Network Time Protocol

OS Operating System

OWAMP One-Way Active Measurement Protocol

OWD One-Way Delay

P2P Peer to Peer

PDF Probability Distribution Function

PGR Permitted Geographic Region

PID Process ID

PKI Public Key Infrastructure

PMF Probability Mass Function

PoW Proof-of-Work

QoS Quality of Services

RF Radio Frequency

RFC Request For Comment

RFID Radio-Frequency Identifier

RTS Request to Send

RTT Round Trip Time

SOI Speed of the Internet

TCP Transport Control Protocol

TIV Triangular Inequality Violation

TPM Trusted Platform Module

TTL Time to Live

UDP User Datagram Protocol

VoIP Voice over IP

VPN Virtual Private Network

W3C the World Wide Web Consortium, the standards body for web technologies

WiFi Wireless Fidelity

WPS WiFi Positioning System

WSN Wireless Sensor Networks

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Remotely proving that an Internet-connected device is geographically present at where it asserts to be remains one of the most challenging problems in today's Internet. This thesis addresses the problem of one party verifying an assertion about the geographic location of a typically remote second *target party* over the Internet. The verifying party is different from and not in physical possession of the target party's device. The target party is a physical device running a process that is able to send and receive Internet data. The goal of the thesis is to devise a mechanism that provides greater assurance about the correctness of an asserted location, compared to current state-of-the art techniques.

More formally, we state the research question as follows:

Assume two devices transmitting data between themselves over the Internet. If the geographic location of one of them is asserted, what mechanisms are available to provide assurance of the correctness of this assertion?

1.1 Terminology and Scope

Terminology. Throughout this thesis, the terms *client* and *Location-Sensitive Provider (LSP)* are such that the client is the target party whose location is important for the appropriate operation of the LSP. The term *geolocation* means (physical) location-determination; hence, *geolocating* means determining the location. Additionally, *Internet geolocation* means geolocating an Internet-connected device.

We define the *evasion of geolocation* as deliberately causing a geolocation technique

to fail or to return an incorrect location.

Scope. While geolocation is of interest, the main focus is *location verification*, meaning that some geolocation mechanism (beyond the scope of this thesis) first asserts a client's location, which is then verified by mechanisms within this thesis.

The thesis focuses on real-time location verification. Non real-time applications, such as in forensic analysis where geolocating IP addresses post-incident is of interest, are out of scope. Solutions devised in this thesis assume a two-way real-time communication between the client and the LSP.

1.2 Motivation

Numerous applications can benefit from reliable information about the locations of Internet clients. The following examples are those where evasion incentives may arise, motivating this thesis. Cases where little or no evasion incentive exists, e.g., location-directed ads, are outside the scope of this thesis.

Fraud prevention. The geographic location where credit card transactions are taking place could provide higher assurance to the authenticity of these transactions, compared to when the location is not known.

Impersonation prevention. Impersonation over the Internet, including through password-guessing attacks, can be reduced by restricting a user's login to locations previously associated with the user's account. An impersonating adversary may thus try to evade geolocation in order to place itself fraudulently in that location.

Policy compliance. Various legal agreements are location-dependent. For example, cloud providers often promise the sole storage of users' data within user-requested jurisdictions. However, motivations to violate such an agreement may arise due to cheaper overseas operations and maintenance costs. Additionally, video-on-demand providers, e.g., Hulu [78], are often licensed to stream only to restricted geographic regions. Gambling regulations differ across jurisdictions, placing a responsibility on gambling websites to enforce these regulations according to where the gambler is geographically located. Many online retailers are required to charge applicable taxes based on users' locations. In general, the motivation to evade geolocation in the cases under the *Compliance* category is usually for gaining location-dependent benefits.

Location-based access control. Sensitive data, such as military documents or patient records, are often allowed to be viewed only from within certain regions. Operations like online bidding, community-related voting, or even ordering home-delivery meals, can be regulated by users' locations, e.g., to reduce spammers.

1.3 Deficiency of Existing Geolocation Mechanisms

Commonly-used Internet geolocation techniques lack integrity or cross-checking of their results. Such techniques fail to adequately consider a knowledgeable adversary that is motivated to cheat about its location. Most of the geolocation literature focus on achieving higher geolocation accuracy, overlooking adversarial environments.

Tabulation-based techniques. These work by having the LSP look up the client's IP address in a pre-populated Database (DB) that maps IP-addresses to locations, e.g., MaxMind [103]. Studies have found that many of the major tabulation providers are evadable, e.g., by having the client simply hide its true IP address using a Virtual Private Network (VPN) [107].

Self-positioning systems. This is the class of techniques where an LSP requests the client's location information from the client itself. The client's device determines its own location using, e.g., Global Positioning System (GPS) [75], WiFi Positioning System (WPS) [158] (see Chapter 2), cell tower triangulation (in the case of mobile devices) [142], and communicates it to the LSP. No geolocation techniques under this class can be relied upon to geolocate adversaries motivated to forge their locations; the asserted location must be verified for prudent use in location-sensitive services.

Measurement-based geolocation techniques [91, 93]. These exploit the correlation between Internet delays and geographic distances in geolocating clients. Delays are measured between the client and a set of landmarks with known locations, and are mapped to distances according to some predefined (usually calibrated) mapping function. Multilateration is then used to determine the client's location relative to the landmarks. When the client's IP address is used in delay measurements (e.g., using the *ping* utility), employing a non-local IP address evades those techniques [107], i.e., similar to evading tabulation-based techniques discussed above. Additionally, even when the IP address is not used, delay manipulations can corrupt the geolocation process [59].

1.4 Thesis Contributions and Organization

1.4.1 Analyzing the security of delay-based geolocation

Previous literature analyzed the effect of an adversary increasing the measured delays on the accuracy and evadability of measurement-based geolocation techniques [59]. Our first contribution is research investigating adversarial evasion capabilities by (1) demonstrating that an adversary could also decrease the measured delays by means of exploiting the lack of integrity in common delay-measurement utilities; (2) demonstrating enhanced adversarial strategies to better utilize delay manipulation for a more accurate misrepresentation of location, thus showing additional vulnerabilities in existing mechanisms; and (3) evaluating the adversarial accuracy in forging its location, now considering the previous two contributions. This is presented in Chapter 3.

1.4.2 Accurate One-way Delay Estimation

Due to delay asymmetry [116], One-Way Delays (OWDs) have the potential to improve the performance of delay-dependent applications [70], such as delay-based geolocation. However, delay-based geolocation techniques usually map Round Trip Times (RTTs) to distances rather than mapping OWDs because the former is easier to estimate.

To that end, and as a tool useful in one of our other contributions (see CPV below), we devised the *minimum pairs* protocol (Chapter 4)—a OWD-estimation protocol that requires no more cooperation between the two parties than that required to estimate RTTs, yet is in many cases more accurate than simply taking half the RTT as a OWD-estimate. This later conclusion is reached by formally deriving the probability distribution of absolute error for these two alternative protocols, as a function of the delay distribution between network nodes.

The *minimum pairs* protocol is a generic contribution, which we believe to be of independent interest; e.g., it could be used by delay-dependent Internet applications for accurate OWD-estimation without the need for the overwhelming cooperation—between the two parties estimating delays—typically required by One-Way Active Measurement Protocol (OWAMP)-like protocols [134].

1.4.3 Client Presence Verification (CPV)

As the main contribution of this thesis, Chapter 5 introduces Client Presence Verification (CPV)—a delay-based location verification technique designed to verify in realtime the presence of an Internet-connected client in a prescribed triangular geographic region. The algorithm employs heuristics to reduce erroneous false rejects/accepts, while retaining reasonable granularity.

In CPV, three verifiers geographically encapsulating the client's asserted location are selected to verify this assertion. They use the *minimum pairs* protocol to estimate OWDs between themselves and the client, and leverage these delays for evidence supporting the client's presence within the triangle determined by their geographic positions.

CPV mitigates common geolocation-evasion tactics explored in the literature [59, 107], as well as the novel adversarial manipulations discussed above in Section 1.4.1. We discuss the integrity of CPV's decisions in the presence of a broad class of adversarial evasion tactics, and argue about the algorithm's defense capabilities against these tactics.

Viability of the CPV algorithm is extensively evaluated (from a networking perspective) through real world experiments on PlanetLab [33]. The effect of various factors on the correctness of CPV is examined, such as the clients' geographic proximity to the verifiers and the triangle they determine, and the number of delay measurements the verifiers perform. This evaluation is presented in Chapter 6.

The PlanetLab nodes employed are connected through a wired access network. To evaluate the use of CPV with wireless clients, we use wireless delay distribution models from the literature, and generate delays following these models. Those delays are then added to the delays measured using the wired PlanetLab nodes to model wireless clients. Several factors are considered, including the number of wireless devices in the vicinity of the wireless client. The wireless evaluation is presented in Chapter 7.

1.4.4 Hindering Unauthorized Traffic Relaying

CPV raises the bar for an adversary trying to forge its geographic location. It is designed to reject adversaries even when they are using a Middle Box (MB), geographically present at their intended location, to hide their IP addresses and relay their traffic from the server. A colluding MB, customized to specifically evade CPV,

may however succeed to mislead CPV to accept the MB's location as that of the client. For example, third party MB service providers, such as public VPNs [150], may well be motivated to customize their infrastructure to evade CPV upon its deployment.

We propose to use a Proof-of-Work (PoW) mechanism, such as client puzzles, to defeat colluding MBs, hindering their illicit traffic relaying. Our proposal is evaluated using a Markov queuing model, and additionally using simulations. Similar to the *minimum pairs* protocol, this proposal may be of independent interest as a standalone contribution since it can be used to resist unauthorized traffic relaying regardless of the application. In our case, we use it to strengthen CPV against colluding MBs. This contribution is presented in Chapter 8.

1.5 Related Publications

Detailed explanation to the CPV algorithm (Chapter 5), the *minimum pairs* protocol it uses (a proportion of Chapter 4), and the algorithm's evaluation in a wired network environment (a proportion of Chapter 6) were published as a full paper in the IEEE CNS conference. The paper was nominated for a Best Paper award.

- A. M. Abdou, A. Matrawy, and P.C. van Oorschot, "Location Verification on the Internet: Towards Enforcing Location-aware Access Policies Over Internet Clients". In *IEEE Communications and Network Security (CNS)*, Oct. 2014.

A followup version of the CNS paper, which includes additional analysis to the CPV algorithm is accepted for publication in IEEE TDSC.

- A. M. Abdou, A. Matrawy, and P.C. van Oorschot, "CPV: Delay-based Location Verification for the Internet". In *IEEE Transactions on Dependable and Secure Computing* (to appear; accepted June 14, 2015).

In addition to introducing the *minimum pairs* protocol, Chapter 4 also evaluates the protocol analytically by first deriving the probability distribution of its absolute error, then comparing its accuracy (using the derived distribution) to the RTT-halving protocol. The evaluation methodology and the derived model were accepted for publication.

- A. M. Abdou, A. Matrawy, and P.C. van Oorschot, "Accurate One-Way Delay Estimation with Reduced Client-Trustworthiness". In *IEEE Commu-*

nications Letters, vol. 19, no. 5, pp. 735–738, 2015.

Finally, Chapter 8 introduces the principle of using a PoW mechanism to thwart MBs (e.g., VPNs and proxies) from illicitly relaying traffic. The principle and its evaluation, both analytically and using simulations, were published.

- A. M. Abdou, A. Matrawy, and P.C. van Oorschot, "Taxing the Queue: Hindering Middleboxes from Unauthorized Large-Scale Traffic Relaying". In *IEEE Communications Letters*, vol. 19, no. 1, pp. 42–45, 2015.

Chapter 2

Background and Related Work

This chapter presents related work on the areas of geolocation and location verification, their limitations and vulnerabilities. Although we only focus on the verification aspect in this thesis, we review Internet geolocation techniques in general to help explore their susceptibility to evasion. Moreover, we analyze measurement-based geolocation techniques as they provide an insight about the nature of delays over the Internet, and the accuracy of mapping delays to distances. A brief survey on location verification in single-hop wireless networks is also presented, as well as state-of-the-art location-proof architectures and their applications.

2.1 Internet Geolocation

An Internet geolocation technique aims to bind a client's identifier (e.g., its IP address) to a geographic location. It either involves Internet delay measurements between the client and a set of reference objects with known locations, and the use of multi-lateration to determine the client's location relative to these objects; or it could be inference-based, where an estimate for the location is inferred from the client's attributes [107] and/or behavior [15]. Either way, a Location-Sensitive Provider (LSP) may ask the client to geolocate itself and inform the LSP, or ask a third party geolocation service provider to geolocate the client given an identifier.

Measurement-based techniques can be further categorized into either *delay-based* or *topology-aware* techniques. In what follows, we review proposals in the literature under each category, review inference-based approaches, then discuss other techniques by which a client geolocates itself and informs the LSP.

2.1.1 Delay-based techniques

Delay-based Internet Protocol (IP) geolocation is a class of techniques where the geographic location of the client machine is determined based on the observed network delays between the machine and a set of geographically scattered landmarks with known locations. These techniques assume the client is able to receive and respond to the delay-measurement probes, which in practice, commonly use Internet Control Message Protocol (ICMP)-based utilities like *ping* and *traceroute*.

Factors affecting Internet delays Four primary delay components exist between two Internet hosts: propagation, transmission,[1] queueing and processing delays at intermediate systems (e.g., routers) [89]. There are also delays imposed by end-systems protocols, such as the Transport Control Protocol (TCP)'s congestion and flow control mechanisms. The flow control mechanism receives its parameters from the destination. It would decrease the sender's transmission rate if the receiver signals "slow-down" cues, or increase the rate otherwise. Additionally, other delay components may arise from different aspects such as a low Quality of Services (QoS) provided by the Internet Service Provider (ISP), and excessively circuitous routes.[2] Route circuitousness could also be a result of user configuration, as in the case of using a proxy server or an anonymizing browser.

How delay-based techniques work Despite a plethora of factors that affect the delays between two nodes over the Internet [34, 153], numerous studies have established that there is a strong correlation between delays and geographic distances [67, 92, 113, 165, 167]. The main characteristic relied on is the propagation delay. Most, if not all, delay-based geolocation techniques mitigate the effect of other delay factors (e.g., queueing due to congestion) by using the minimum of multiple delay measurements (e.g., 10-20 RTTs) to the client from each landmark. Once delays are measured, the research question addressed by most techniques becomes finding the best function to map them to geographic distances.

[1] Note that the transmission delay of a packet is measured from the time the first bit of the packet is placed on the transmission media, until the last bit is similarly placed. It is a function of the packet length (in bits) and the media's transmission capacity (in bits per second). In contrast, the propagation delay of a (single) bit is the time required for the bit to propagate through the media from the sender to the receiver. It is a function of the distance spanned in the media (in meters) and the media's transmission speed (in meters per second).

[2] A network route is said to be *circuitous* when the geographic distance it spans is considerably larger than the (shortest) geographic distance between its source and destination.

An exception is one of the first delay-based techniques: GeoPing [113]. Instead of mapping delays to distances, GeoPing matches the location of the client to a location where the most similar delay behavior has previously been observed. Assuming n landmarks and m reference nodes with known locations,[3] the landmarks in GeoPing first create a delay vector to each of the m nodes. A delay vector of a node contains n values corresponding to the RTTs between the landmarks and the node. Each landmark then measures the RTT between itself and the client, enabling the landmarks to create a delay vector for that client. The location of the node with the *nearest* delay vector is then returned as the client's calculated location. The authors of GeoPing proposed to calculate nearness between two delay vectors as the n-dimensional Euclidean distance between the two vectors [113]. Ziviani *et al.* showed that Manhattan, Canberra, and Chebyshev distances can generate more accurate results when used as alternatives to the Euclidean distance [166]. This class of delay-based geolocation returns a location from a discrete space depending on the number of available reference nodes.

The authors of delay-based techniques often contribute a function that can map delays-to-distances accurately. In most proposals [44, 67, 156], the function's parameters are landmark-specific, and are calibrated prior to geolocating clients. Calibration occurs by having each landmark measure RTTs to all other landmarks; {RTT, distance} pairs are then used to calibrate the mapping function.[4] The fundamental difference between delay-based geolocation techniques in the literature lies in the proposed mapping function. After mapping delays to distances using this function, these distances are used to calculate the client's location using, e.g., multi-lateration.

Each landmark in Constraint-Based Geolocation (CBG) [67] calibrates a linear delay-to-distance function called the *best line*. On a graph where the x-axis is the distance (in km) and the y-axis is the RTT (in milliseconds), the authors of CBG define the best line to be the one "closest to, but below, all data points (x, y) and has a non-negative intercept"[5] [67]. After calibration, each landmark measures the RTT to the client, and maps it to distance using the best line function. The client's location is then estimated as the centroid of the intersection of circles whose centers are the landmarks and radii are the distances. The authors of CBG later proposed a mechanism to estimate and remove delays caused by buffering of the message along the route between landmarks and the client, resulting in a more accurate mapping

[3] The problem of the geographic placement of such infrastructure to enhance the geolocation accuracy was well studied in the literature [164, 167].

[4] The "*distance*" element is the geographic distance between a pair of landmarks.

[5] The *intercept* is the intersection with the y-axis.

to distances [66].

Dong *et al.* [44] proposed to cluster the {RTT, distance} coordinates of the landmarks into k clusters. The coordinates in each cluster are then fitted to a polynomial function, which is then used by the landmark to map delays to distances. Such a segmented polynomial approach makes use of the observation that delay-to-distance ratios vary according to the spanned geographic distance [44].

Youn *et al.* [156] proposed to apply kernel density estimation in the calibration phase to approximate the Probability Distribution Function (PDF) of delays with respect to distances, while Arif *et al.* [14] used maximum likelihood estimation [86]. A Naive Bayes technique for delay-distance calibration was also considered [48].

Laki *et al.* [90] proposed to calibrate one delay-to-distance mapping function across all landmarks, hypothesizing that the relationship between delays and distances is not landmark-dependent. By doing so, delay measurements from all landmarks were combined together to generate such a global mapping function.

GeoWeight [13] is another example geolocation technique that works by calculating a *weight* factor reflecting the client's presence inside a region, for some regions that are determined in realtime. Weights are calculated as the number of overlapping circles in that region. Recall, a *circle* at a landmark is one that has the landmark's location as its center and the estimated distance between the landmark and the client as its radius.

Eriksson *et al.* [47] devised a lightweight geolocation technique that can reduce the number of required probing messages, yet achieve comparable geolocation accuracy. Similar to GeoPing, Eriksson *et al.* [47] rely on delay vectors between a group of passive monitors and landmarks, while leveraging likelihood estimation.

2.1.2 Topology-aware techniques

Topology-aware geolocation techniques leverage the network topology to generate a richer set of constraints, compared to those of delay-based techniques, to more accurately geolocate clients. Intermediate systems between the client and the landmarks are iteratively geolocated using single-hop delay-based analysis. The increased accuracy comes at the cost of longer geolocation time and more required resources.

Katz-Bassett *et al.* [85] proposed to use *traceroute* measurements from the landmarks to the client in order to identify the network topology. They devised some techniques to refine their topology identification, such as detecting multiple device

interfaces and using Domain Name Server (DNS) LOC records [37]. The authors then use the network topology combined with the constraints of the speed of traffic propagation in fiber [117] to geolocate the client.

Similar proposals involved leveraging negative constraints to enhance the geolocation accuracy [153]. In contrast to the regular (positive) constraints, negative constraints exclude regions where the client cannot be present at, i.e., based on the delay measurements.

Others have proposed to leverage large numbers of *passive landmarks* with known locations to further enhance the accuracy [68]. A passive landmark is usually a public server that responds to ICMP queries, e.g., *ping* or *traceroute*, but is not under control of the LSP or the geolocation service provider—neither can conduct measurements originating from it. After populating a table of thousands of passive landmarks, Wang *et al.* [148] combined delay-based with topology-aware techniques to constrain the region where the client is. Their technique then returns the location of the nearest (delay-wise) passive landmark to the client as the client's location. This last step makes use of the *closest-shortest rule*, which states that shorter delays tend to result from smaller distances [93].

2.1.3 Client self-geolocation

In this geolocation category, the client determines its own location and informs the LSP. The client may have determined its location using, e.g., its GPS. Another abstract example under this category is having the LSP simply asking the (human) user to input its location, e.g., in an *address* field on the LSP's website [107]. Note that, regardless of how the client determines its location, we place all class of techniques by which the client sends location information to the LSP under the *self-geolocation* category even if the geolocation method involves, e.g., delay measurements (i.e., similar to those reviewed in Sections 2.1.1 and 2.1.2).

Commonly used over the Internet, the World Wide Web Consortium (W3C) geolocation Application Programming Interface (API) [123] defines an interface that allows the client's web browser to determine and return the client's location to the requesting LSP. Browser vendors usually rely on common location-determination technologies, such as GPS [75] or WPS[6] [158]. Because the client sends its location

[6]In WPS, a device's location is determined relative to the wireless access points. The device's Network Interface Card (NIC) reports a list of visible access points and their signal strengths

to the LSP, it can submit false information before submitting it [129].

2.1.4 Inference-based Approaches

In this class of approaches, the client's location is inferred from observations of its transmitted data [107]. For example, a time zone in a Hypertext Transfer Protocol (HTTP) packet header being UTC+12 likely indicates that the client is in New Zealand. If the Chinese language was set in the `Accept-Language` header, the party is likely to be from China. If the party's domain name ends in `fr`, it is likely in France. Even the preferred encoding of data gives an insight about the possible locations.

The client's location could also be inferred from its IP address [137]. The IP address is used to consult geolocation service providers that maintain lookup tables mapping IP addresses to locations [135], e.g., MaxMind [103] and HostIP [43]. However, such tabulation-based techniques were found unreliable [121].

2.2 Vulnerabilities of Internet Geolocation

As this section discusses attempts whereby a client is motivated to forge its own location, the *client* is referred to as the *adversary*.

Asking the adversary to calculate its location and inform the LSP of that location enables the adversary to provide misleading information about its location, provided that no additional verification mechanism is employed. In this case, the adversary may not only misrepresent its location, but also accurately control the location where it claims to be at. A verification mechanism could, for example, be to use Trusted Platform Module (TPM) chips [97] to trust GPS-calculated coordinates. However, location coordinates obtained from a TPM-supported GPS driver may still be vulnerable to the Cuckoo attack [115], where an adversary colludes with a remote party having a TPM-supported GPS to fake the adversary's location.

Against inference-based approaches, the adversary can alter information that indicates its true location [107], misleading the LSP into calculating the adversary's

(reflecting the distance between the NIC and the access point) to a "location provider". Location providers manage lookup tables that map access points to their corresponding geographic coordinates. The location provider calculates the location and informs the browser.

presence in the forged location. For example, changing the browser-requested language from *ja* to *it* may cause the LSP to believe the adversary is in Italy instead of Japan. The adversary's control over the forged location in this case depends on the information used to determine the location.

2.2.1 IP-hiding Attacks

IP geolocation techniques, whether they are measurement- or tabulation-based, are prone to being mislead using MBs such as proxy servers, VPNs, anonymizers [41] or similar IP-hiding technologies.

We believe that an adversary motivated to misrepresent its location would easily adopt any such technologies, especially given the wide availability of public VPN-service providers. A number of these public anonymizers are even available free of charge [150]. As such, a fundamental design goal in any Internet location verification mechanism is to address such a well-known evasion tactic, as we do by the mechanism introduced in Chapter 5.

MBs tend to alter transport-layer headers and/or react differently to ICMP messages, compared to (non-MB) end-systems [39]. To detect a MB, the provider and the client typically exchange especially-crafted packets and notice unexpected changes on the other end [72]. Due to such considerable client cooperation requirement between the two parties, these techniques cannot be implemented by an LSP to detect MBs before geolocating an adversary by its IP addresses.

Attempts to enumerate the IP addresses of MBs and block them do not ensure their detection due to the dynamic behavior of IP addresses assignment [154], and the risk of falsely blocking IP addresses that are not associated to MBs [30].

2.2.2 Delay-adding Attacks

Gill *et al.* [59] analyzed adversarial location-forging abilities when the adversary increases delays to evade a measurement-based geolocation. The authors [59] explored the case where the adversary injects delays by not responding to echo-request messages promptly. They found that, although the adversary was able to misrepresent its location, it had little control over the forged location. Additionally, the authors [59] found that such adversarial manipulations are better detected when the adversary attempts to fraudulently place itself farther away from its true geographic location.

Gill *et al.* [59] also tested a more sophisticated adversary that has control over a full network (such as an Autonomous System (AS)-owner or a cloud provider), not just the device it owns. They found that such an adversary can misrepresent its location more accurately against topology-aware techniques, than delay-based ones. This adversary was tested to model a cloud provider [59].

2.2.3 Delay-shortening Attacks

One of the findings of this thesis is that common ICMP-based delay-measurement tools allow an adversary to fully manipulate, i.e., increase and *decrease*, the delays observed by the measuring party, which is made possible due to the lack of integrity in these tools.

Because those ICMP-based tools are commonly used in measurement-based geolocation, full delay manipulation not only allows an adversary to misrepresent its location but also gives the adversary substantial control over the forged location, compared to the evasion tactic proposed by Gill *et al.* [59]. In such case, susceptibility to evasion stems from the reality that ICMP-based delay-measurement tools were not designed for adversarial environments.

We explain this attack in details in Chapter 3, where we also propose strategies by which an adversary can increase its control over the location which a geolocation technique perceives to be the adversary's actual location.

2.3 Location-verification

2.3.1 Single-hop wireless networks

Verifying the proximity of two devices to each other using delays has been well studied in contexts other than the Internet, such as single-hop wireless networks, e.g., Radio-Frequency Identifiers (RFIDs) and Wireless Sensor Networkss (WSNs) [57]. Brands and Chaum [23] have proposed a Radio Frequency (RF)-based distance bounding protocol that aims at proving an upper bound to the distance between a prover and a verifier. To address the high sensitivity to processing delays and the complexity of achieving highly accurate clock synchronization among the verifiers, Wagner *et al.* [133] proposed an ultrasound-based approach using a prover and a group of verifiers. Capkun *et al.* [28] emphasized the importance of having at least

three verifiers surrounding a prover to account for delay-adding attacks introduced by the prover or a third party attacker.

The nature of delays over the Internet differs from those in single-hop wireless networks. Internet delays alleviate some of the challenging problems in the single-hop wireless context (e.g., less sensitivity to processing delays), but introduce new challenges (e.g., stochastic queueing delays due to traffic/route uncertainty [44]). Thus, proximity verification in single-hop wireless networks is a distinct research problem from the one addressed in this thesis, since our focus is on Internet location verification.

2.3.2 Privacy-Preserving Location-Proof Architectures

Delay-based techniques for single-hop wireless networks (see Section 2.3.1) are often leveraged in the literature to design *location proof architectures*, also sometimes referred to as a *spatial-temporal attestation service* [63], which enable users to obtain proofs of their presence in a certain location [64]. The proof is often in the form of a certificate, where a *trusted* party in the client's vicinity is available to certify its presence [96, 149].

Saroiu *et al.* [132] proposed a location-proof architecture in which a user gets cell towers or Wireless Fidelity (WiFi) access points to certify that the user is present where it claims to be at. VeriPlace [100, 101] is another architecture that addresses *wormhole* attacks,[7] while focusing on the users' privacy by employing cryptographic techniques to spread user's identification credentials across different entities.

Other proposals involved decentralizing the trusted infrastructure [56], or replacing it with Bluetooth-based devices in the vicinity of the client [163], or with Near Field Communication (NFC) tags [122]. Zerosquare [120] is a privacy-preserving *location hub*, which allows location-based services to query the users' locations, while regulating access to personal information by separating user information from their location.

The threat of a compromised infrastructure has been addressed as well, where Khan *et al.* [88] proposed to use additional (trusted) witnesses in the user's vicinity to verify assertions in the presence of untrusted access points/location managers. The

[7] A wormhole is a relaying attack in wireless networks, where an adversary encapsulates bytes at one location, relays and decapsulates them at another location in the network [76].

principle of verified location tracking has been explored as well, where a chain of location proofs can represent the history of a user's locations [87, 146].

Relationship to the applications addressed by this thesis Compared to the Internet location-verification problem we address in this thesis, location-proof architectures target a problem with more constraints (e.g., preserving users' privacy and verifying locations with high granularity). Thus, they address a different class of applications than the ones addressed by this thesis.

The applications addressed by location-proof architectures provide the advantage of being privacy-centric; users are assumed to be unwilling to share (or publicly disclose) the credentials identifying them to an LSP. Such assumption must hold; otherwise, a user who wants to forge their location may send their credentials to a colluding party to get them bound to a remote location and endorsed by an access point at that location. As stated earlier in Chapter 1, we assume no client credentials playing that role in the applications addressed in this thesis.

There are many cases where users could be unwilling to disclose their credentials. For example, doing so may reveal private (location) information such as regular hospital visits. Additionally, sharing identification credentials may threaten losing benefits associated with these credentials [94]. Foursquare [53] for example enables coffee shops to reward users, identified by their personal (secret) credentials, when they visit regularly; a user sharing their credentials with a remote colluding party may risk having their rewards lost/stolen.

Another fundamental difference between the applications addressed by location-proof architectures and the ones addressed in this thesis lies in the verification granularity and trustworthiness of infrastructure. The granularity addressed by location-proof architectures is very high, e.g., verifying the presence in a hospital ward or inside a coffee shop. Such high granularity is made possible since wireless devices in the user's vicinity are typically relied upon for location verification. To achieve location verification at a global-level, we expect the high granularity of location-proof architectures would come at the cost of large-scale trustworthiness requirements, since a sufficient number of trusted wireless (endorsing) devices must be present to cover geographic regions at a high granularity. In contrast, the location-verification granularity addressed in this thesis is coarser (e.g., state- or country-level) as we explain in Chapter 5, and the required trusted infrastructure is therefore smaller (see Chapters 5 and 6 for details).

Chapter 3

Accurate Manipulation of Delay-based Internet Geolocation

Numerous delay-based Internet geolocation techniques have been proposed in recent years, and are repeatedly positioned as well suited for security-sensitive applications (e.g., location-based access control, credit card verification). Previous literature [59] showed that an adversary simply delaying response messages to increase measured RTTs gains only limited location control in forging its location, and decreasing RTTs was believed to be infeasible. In contrast, herein we show that indeed an adversary can decrease RTTs arbitrarily because commonly-used ICMP-based utilities are not intended to provide delay-measurement integrity, and explore how an adversary can leverage this to *accurately* manipulate geolocation results. Using several adversarial models, we evaluate (on three delay-based geolocation techniques) how selectively combining this with delay increases can achieve surprisingly high adversarial accuracy in forging location—e.g., modeled adversaries can fraudulently misrepresent their true location by over 15,000 km, some within 100 km of their intended (fraudulent) target location. Thus the new ability to decrease delays, combined with previous delay-increasing tactics, enables significantly greater adversarial location control over previous methods.

3.1 Introduction

The recent proliferation of Location-Based Services (LBSs) in the Internet has highlighted the requirement for reliable and accurate Internet geolocation tools. Some of these services employ location-based access policies [18], or restrict operations

by clients' geographic locations [143]. Examples include media streaming [26], online voting/gambling, location-based social networking [122], and fraud prevention [30]. Nanjee [109] is one example that provides commercial geolocation services based on active network measurements [148]. Tabulation-based IP geolocation service providers maintain lookup tables that map IP addresses to locations. Studies have found that many of the major tabulation providers (e.g., MaxMind [103] and HostIP [43]) are inaccurate/outdated [121, 135] and evadable [107].

IP geolocation techniques that rely on active network (delay) measurements have accuracy advantages over others, e.g., tabulation-based [137]. They are also resilient to security vulnerabilities that other techniques suffer, such as clients submitting false location information [107,148]. Although IP geolocation is often evadable using VPNs and similar IP-hiding technologies [107], such technologies can be detected [30] and thwarted [12]; some LBSs (e.g., Hulu [78]) have recently started employing these practices [52, 140]. For these reasons, delay-based techniques are gaining increasing community support [49,167], specifically advocated [121], and repeatedly positioned as well suited for security-aware contexts, e.g., ensuring legitimate storage of data in the cloud [62], or locating hidden servers [31].

Since 2001, more than 10 delay-based IP geolocation techniques have been proposed [13,14,44,47,48,67,68,91,93,113,156,166]. The RTT is measured between the client and a set of landmarks with known locations, and the client's location is estimated relative to the landmarks. Delay-based geolocation techniques require some way of measuring delays; because ICMP [125] utilities (e.g., *ping* and *traceroute*) are ubiquitous and facilitate delay measurements, they are commonly used for that purpose [44, 156].

In 2010, Gill *et al.* [59] studied the ability of an adversary to distort geolocation techniques that are based on active delay measurements, and the capability of the geolocating party to detect circumvention. For delay-based techniques, their analysis considered an adversary that can only increase the observed RTTs by selectively delaying response messages. Their modeled adversary succeeded to misrepresent its location, but was limited to a coarse control over the forged location.

In this chapter, we show that common delay-measuring utilities used by geolocation techniques are subject to (1) modifying and/or (2) predicting packet contents; this enables an adversary to fully manipulate, i.e., increase and decrease, the observed RTTs. For example, GNU's not Unix (GNU)'s implementation of *ping* relies on the ICMP echo request/reply protocol, and records the packet-creation time in the DATA field of the ICMP packet [54]. The RTT is then calculated by subtracting this

timestamp, as read from the echoed packet, from the time the echoed packet was received, expecting the client to return the data unchanged. However, an adversarial client seeking to manipulate the geolocation mechanism can alter the timestamp in the DATA field before echoing the packet. We show that using this, an intelligent adversary can selectively increase or decrease the observed RTT as necessary to beneficially control the calculated location.

Upon being able to manipulate delays, the adversary faces the question: What should the RTT between each landmark and the adversarial client be such that the geolocation process calculates the adversary's intended location? We propose different strategies whereby an adversary can utilize information known about the landmarks to answer this question. We model several adversaries adopting different strategies, and compare their accuracy in forging location.

To study the effectiveness of the modeled adversaries, we implemented three delay-based geolocation techniques, CBG [67], GeoPing [113] and segmented polynomial (*SegPoly* for short) [44], and evaluated adversarial location-forging accuracy, given the ability to fully manipulate delays. Some adversaries modeled obtained forged locations with *distance errors* (defined as the distance between the adversary's intended location and the one calculated by the geolocation technique) below 100 km; this relatively fine-grained location control was possible even for some who attempted fraudulent relocation more than 15,000 km from their true locations (see Fig. 3.1).

Our work highlights the need for integrity of timing information when relied upon by security-sensitive applications. Contributions:

1. We show how properties of common ICMP-based delay-measuring utilities allow an adversary to both increase and decrease measured delays.

2. We demonstrate several strategies that enable an adversary, exploiting these properties, to accurately forge the location calculated by delay-based geolocation techniques.

3. We evaluate the manipulation effectiveness to three techniques: CBG [67], GeoPing [113], and SegPoly [44]. This demonstrates how powerful an adversary can be upon being able to fully manipulate RTTs.

The rest of this chapter is organized as follows. Section 3.2 reviews common delay-measurement utilities. Section 3.3 explains how RTTs can be fully manipulated, i.e., increased and decreased. The adversarial models are defined in Section 3.4. Section 3.5 analyzes the effect of manipulating RTTs on delay-based geolocation.

(a) Attempted distance on CBG= 15,092 km. Dist error = 70.7 km.

(b) Attempted distance on GeoPing = 8,055 km. Dist error = 71.4 km.

(c) Attempted distance on SegPoly = 6,617 km. Dist error = 75.2 km.

Figure 3.1: Examples of adversarial capabilities after exploiting properties of common delay-measuring utilities. • = true location of adversary; × = intended location of adversary; ○ = locations calculated by (a) CBG [67], (b) GeoPing [113], and (c) SegPoly [44]; *attempted dist* is that between • and ×; *dist error* for the adversary is that between × and ○. Map data: Google, INEGI, Basarsoft.

Section 3.6 compares location-forging abilities of different adversarial models. Section 3.7 suggests countermeasures. Section 3.8 discusses related work, and Section 3.9 concludes.

3.2 Background: RTT Measurement Using Common ICMP-based Utilities

A *sender* can measure RTTs between itself and a *receiver* by having the receiver respond to (special) packets of the sender, and timing these responses. Assuming the sender issues these packets to the receiver every t ms, and the first one was created at time T, then the sender's system time when packet i was created, for all packets $i \geq 0$ is:

$$s_i = T + i \cdot t \tag{3.1}$$

If the packets take γ_1 ms one-way delay from the sender to the receiver, they reach the receiver at times:

$$m_i = s_i + \gamma_1 = T + i \cdot t + \gamma_1 \tag{3.2}$$

Assuming the receiver responds promptly, if packets take γ_2 ms one-way delay from the receiver back to the sender, the responses arrive at times:

$$r_i = m_i + \gamma_2 = T + i \cdot t + \gamma_1 + \gamma_2$$

The sender calculates the RTT for packet i as:

$$RTT_i = r_i - s_i = \gamma_1 + \gamma_2 \tag{3.3}$$

To measure RTTs, network utilities commonly use the ICMP protocol [125],[1] as it is implemented by default in most systems' protocol stack. An ICMP packet gets wrapped by an IP packet for delivery. Eleven ICMP types are specified by Request For Comment (RFC) 792. The type is indicated by the TYPE field of an ICMP header. *Echo-request/reply*, types 8 and 0 respectively, and *destination-unreachable*, type 3, are the ICMP options commonly used to measure RTTs. The RFC does not specify a mechanism to calculate RTTs for either types [125].

[1] Some utilities, such as *tcptraceroute* [3], rely on TCP messages.

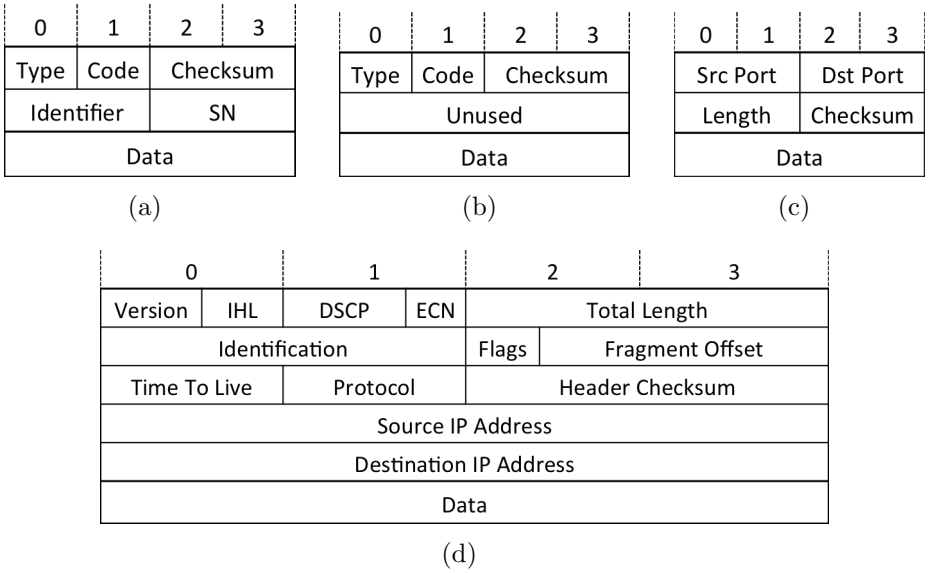


Figure 3.2: (a) ICMP echo-request/reply packet format; (b) ICMP destination-unreachable packet format; (c) UDP segment format; (d) IPv4 packet format.

Echo-request/reply

Echo-request and echo-reply share the same message format, which is shown in Fig. 3.2(a). To construct an echo-request, the sender sets the TYPE and CODE fields to 8 and 0 respectively, chooses two 16-bit values for the IDENTIFIER and SEQUENCE NUMBER fields, and finally (after filling the DATA) calculates the checksum and places it in its field. Values in the DATA, IDENTIFIER and SEQUENCE NUMBER fields are up to the implementer, however the latter two may aid in matching requests with replies as specified by the RFC [125]. One of the most commonly used network utility that implements the echo-request/reply type of the ICMP protocol is *ping*. We found that many, if not all, *ping* implementations on Linux, Berkeley Software Distribution (BSD) and Mac Operating System (OS) place the Process ID (PID) of the issuing process in the IDENTIFIER field. The SEQUENCE NUMBER field usually starts by either 0 or 1, and is incremented by 1 in each subsequent *ping* message [1, 7, 54]. When the receiver gets an echo-request message, it should change the TYPE field to 0, recalculate the checksum and echo the packet. According to RFC 792, "the data received in the echo message must be returned in the echo-reply message" [125]. However, the RFC does not specify a mechanism to ensure the receiver behaves as described. That is, providing integrity checking is not stated as a requirement. And, of course, attackers feel no particular obligation to follow RFCs.

To calculate the RTT using the echo-request/reply options, two common implementations exist: *stateless* and *stateful*. In the stateless implementation, the sender places the timestamp s_i (packet-creation time) in the DATA field of the ICMP packet.

When the echo-reply is received, the sender observes the receiving time r_i, reads s_i from the echoed packet, and uses them to calculate the RTT using (3.3). Other examples of such stateless implementation include, but not limited to, *ping* on FreeBSD [7] and Mac OS [1].

In the stateful echo-request/reply implementation, the sender records s_i in its local memory. The RTT is calculated also using (3.3), but reading s_i from the sender's local memory instead of the echo-reply packet. Examples of this stateful implementation that use the echo-request/reply options include GNU's *traceroute*—ICMP option (i.e., `traceroute -I <host>`) [8], and *hping3*—ICMP option (i.e., `hping3 -1 <host>`) [4]. These stateful utilities commonly fill the DATA field using a fixed predefined pattern, e.g., all zeros, a list of sequential ASCII characters, or hard-coded strings.

Destination Unreachable

To calculate the RTT using the destination-unreachable option, the sender creates a UDP segment and sends it to the receiver, with a destination port unlikely to be open on the receiver's machine. The sender records s_i in its local memory. As with the DATA field of the echo-request/reply type explained above, the data of this UDP segment is usually filled with fixed predefined patterns [2,6]. If the port was actually closed, the receiver is expected to respond with an ICMP destination-unreachable message [22]; when the sender receives it, the sender records r_i, and calculates the RTT using (3.3). Utilities implementing this behavior are commonly stateful, s_i is recorded locally, because the receiver is not echoing an exact copy of the sender's packets. GNU's *traceroute* is an example employing this implementation through its (default) UDP probes [8].

The destination-unreachable message format is shown in Fig. 3.2(b). To construct its header, the receiver sets the TYPE and CODE fields to 3 and 3 respectively, fills the 32-bit UNUSED field with zeros, and finally (after setting the DATA field) calculates the checksum and places it in its field [125]. To enable the sender match responses with their corresponding processes, the receiver places the IP header and the first 8 payload bytes of the IP packet it received from the sender in the DATA field of the destination-unreachable message [125].

Table 3.1: Properties of ICMP-based utilities, and the effects of exploiting them on the observed RTTs. A bullet (•) in column 3 means Case i has property j.

	Property	Effect		Case			Discovered
		↑ RTT	↓ RTT	1	2	3	
1	Suspendable responses	✓		•	•	•	[59]
2	Modifiable pkt contents	✓	✓	•			herein
3	Predictable pkt contents		✓		•	•	herein

3.3 Manipulating Latencies

From Section 3.2, the cases whereby a sender can measure RTTs using ICMP are:

- Case 1: stateless using echo-request/reply.
- Case 2: stateful using echo-request/reply.
- Case 3: stateful using destination-unreachable.

Table 3.1 lists potentially-exploitable properties of common ICMP-based network utilities. These properties become vulnerabilities when the measured RTTs are relied upon by security-sensitive applications; because we investigate the effect of using ICMP-based utilities in security-sensitive geolocation purposes, we refer to the properties in Table 3.1 as *vulnerabilities*. The table also shows which of the RTT-measurement cases listed above has which vulnerability. Note that despite having the same effect on RTTs, the first and second vulnerabilities in Table 3.1 increase RTTs in a different way; likewise, the second and third decrease differently. We now discuss how an adversary can increase/decrease the RTTs when it exploits the corresponding vulnerabilities in each case.

Exploiting the first vulnerability in Table 3.1 enables the adversary increase RTTs in all three cases, because the adversary needs only hold on to the response messages to increase the RTTs [59]. Decreasing RTTs, in each case, is achieved as follows.

Case 1. The packet-creation time, s_i, is recorded in the ICMP echo-request in this case. To decrease RTTs, the adversary exploits the second vulnerability in Table 3.1; it increases the value of s_i before including it in the echo-reply. Changing s_i to $s_i + \delta$ decreases the RTTs the sender observes by δ. Using (3.3), the sender calculates the manipulated RTT of packet i as:

$$\mathrm{RTT}'_i = r_i - (s_i + \delta) = RTT_i - \delta \tag{3.4}$$

RTTs can also be fraudulently increased by δ ms by changing s_i to $s_i - \delta$. If the adversary knows the actual RTT between itself and the sender, it can mislead the sender into calculating the RTT as a specific value of the adversary's choosing, τ, by setting:

$$\delta = RTT_i - \tau \tag{3.5}$$

causing the sender to calculate the manipulated RTT as:

$$\text{RTT}'_i = r_i - (s_i + \delta) = RTT_i - \delta = \tau \tag{3.6}$$

Case 2. To decrease RTTs, the adversary first estimates the *waiting* time, t in (3.1), that the sender waits between sending echo-requests. Recall that delay-based geolocation techniques take multiple RTT measurements to a client, and use the smallest in geolocation. To estimate t, the adversary refrains from responding to the first $n > 1$ echo-requests, or drastically delays their responses to ensure none of them will be chosen as the smallest. It then subtracts the receiving time of the echo-request, m_i in (3.2), from m_{i+1} for all $0 \leq i < n - 1$ (Section 3.2). Because the accuracy of this method depends on the stability of the one-way delay from the sender to the adversary, the adversary averages the waiting time over multiple packets:

$$t = \frac{1}{n-1} \sum_{i=0}^{n-2} (m_{i+1} - m_i) \tag{3.7}$$

The adversary can then estimate the receiving time of the next echo-request packet as:

$$m_i = m_{i-1} + t \tag{3.8}$$

To decrease the RTT that the sender observes from packet i by δ ms, the adversary issues an early (fake) echo-reply at times m'_i, instead of m_i, such that:

$$m'_i = m_i - \delta = s_i + \gamma_1 - \delta \tag{3.9}$$

The sender will then receive replies at times r'_i, such that:

$$r'_i = m'_i + \gamma_2 = s_i + \gamma_1 - \delta + \gamma_2 \tag{3.10}$$

and hence, calculate the RTT of packet i as:

$$\text{RTT}'_i = r'_i - s_i = \gamma_1 + \gamma_2 - \delta \tag{3.11}$$

If the adversary knows the actual RTT, it can use (3.5) to mislead the sender into calculating the RTT as τ.

Issuing early ICMP echo replies requires the adversary to craft them before receiving their corresponding requests. Exploiting the third vulnerability in Table 3.1 enables the adversary achieve this because the values in the header of an echo-reply message, Fig. 3.2(a), are highly predictable. When the sender receives an echo-reply (type 0, code 0), it only uses the IDENTIFIER and SEQUENCE NUMBER fields to match them with corresponding requests; they are the only two fields an adversary needs to predict, before receiving them in echo-requests. The IDENTIFIER (commonly being the PID of the issuing process) is usually constant across echo-requests issued within the same session, the SEQUENCE NUMBER is usually 1 plus the previous echo-request [4, 5, 8]. After receiving the first echo-request, which the adversary ignores, it predicts the values of those two fields for subsequent requests.

Case 3. Similar to the previous Case, the adversary decreases RTTs by sending early (fake) destination-unreachable messages. Timing analysis is, thus, similar to that of Case 2. From the destination-unreachable header, Fig. 3.2(b), we see that the ICMP header constitutes no difficulties for the adversary to predict; the TYPE and CODE fields are set to 3 [125], the UNUSED bytes must be set to 0 [125], and the CHECKSUM is calculated after placing the data. Predicting the DATA field requires the adversary to predict the sender's IP header and the first 8 bytes of the IP payload. We found that given common implementations of ICMP-based utilities, both headers are highly predictable after receiving the first UDP segment from the sender.

For the IP header (Fig. 3.2(d)), the following fields are not expected to change across multiple packets issued within the same session: version, Internet Header Length (IHL), total length, fragmentation bytes (flags+offset), protocol, and source and destination IP addresses. Fragmentation is likely to remain zero because UDP segments are typically small in size; otherwise, they may distort the measurement RTTs due to additional processing and transmission delays of large packets. The protocol number will be set to 17, for UDP [124]. The following fields are already prone to changes by intermediate systems (e.g., routers) [126]: Differentiated Services Code Point (DSCP), Explicit Congestion Notification (ECN), Time to Live (TTL), and header checksum. Thus, the sender cannot rely on those fields to match the returned ICMP messages to an issuing process (we noticed no utilities relying on them). For the remaining field, IP identification, most systems increment it by 1 in

each subsequent IP packet. This summarizes the adversary's ability to predict the contents of the next IP header after receiving at least one.

The first 8 bytes of the IP payload constitute the UDP header (Fig. 3.2(c)). On many implementations, including the *traceroute* utility of GNU [8], FreeBSD [6], and Mac OS X [2], the source and destination port numbers are fixed over a single session, or incremented by one with each subsequent UDP segment. Similar to the stateful echo-request utilities (Section 3.2), the DATA field of UDP segments is commonly a fixed predefined pattern. However, we found that many utilities overlook the returned values in this field, as well as the returned UDP header length and checksum; that is, only the UDP source and destination port numbers are used to match destination-unreachable messages with corresponding UDP segments.

3.4 Adversarial Models

3.4.1 Common Capabilities

The adversary is a client that tries to misrepresent its own location by manipulating geolocation. The adversary's objective is to have the technique return a location as close as possible to its intended location, rather than its true location. We consider a LSP that uses a delay-based geolocation technique that relies on ICMP messages to measure delays.[2]

The adversary has full control over its own machine, but no other machines. It cannot influence the delay-distance calibration process of the landmarks (see Section 2.1.1, page 9), nor infer their calibration functions. Note that the adversary is nonetheless a powerful one since, as shown below, the adversary can achieve high accuracies while manipulating geolocation, even lacking knowledge of those parameters. The adversary is able to selectively manipulate the delays between itself and any landmark, as explained in Section 3.3, and accurately increase or decrease the RTTs observed by a measuring party. The adversary only knows the geographic locations of the landmarks, but does not know the RTT between each landmark and its *intended* location (where the adversary wants to appear to be, in terms of the result computed by the geolocation technique), nor between each landmark and its

[2]Note that the adversary can lead the LSP to rely on ICMP-response messages simply by filtering all TCP ports, i.e., no TCP response messages are sent on attempted connections to any port.

Table 3.2: Capabilities and assumptions of 5 modeled classes of adversaries, their assumed traffic propagation speed, and where they are discussed.

Adv. class	Able to ↑ RTT	Able to ↓ RTT	Knows G	Knows T	Knows F	Traffic Speed	Proposed	Section
A	✓	✓	✓			(1/3)**c**	herein	3.5
B	✓		✓			(2/3)**c**	[59]	3.6
C	✓		✓			(1/3)**c**	herein	3.6
D	✓	✓	✓	✓		Variable	herein	3.6
E	✓		✓		✓	Variable	[59]	—

G = landmarks' locations; T = adversary-to-landmark RTT; F = landmarks' calibrated delay-distance function; **c** = speed of light.

true location. We make the latter assumption because the landmarks may prevent anyone from *ping*ing their addresses except, perhaps, themselves.

3.4.2 Strategies for Modeling Traffic Speed

Let the adversary's true location be a, the set of landmarks be L, and the RTT (at a given time) between the adversary's true location and each landmark $l \in L$ be $\alpha(a, l)$. To deceive a geolocation process, the adversary manipulates the RTTs, observed by each landmark $l \in L$, between itself and l. To forge its location to a', the adversary ideally deceives each $l \in L$ to measure the RTT as one that would be consistent with $\alpha(a', l)$ instead of $\alpha(a, l)$. The challenge for the adversary is that (by our assumption) it does not know both $\alpha(a, l)$ and $\alpha(a', l)$.

If the adversary guesses the speed of traffic propagation, it can estimate the RTT (at current time) because it knows the distances between itself and the landmarks. The adversary may use the constant (2/3)**c** (i.e., speed of light in fiber [117], where **c** is the speed of light in vacuum) as an estimate to the traffic propagation speed [59]. However, Katz-Bassett *et al.* [85] found that a speed between (2/9)**c** and (4/9)**c** better reflects the one-way delay nature of the typically multi-hop Internet routes. We study the adversary's manipulation capabilities when it uses (3/9)**c** = (1/3)**c** as an approximation to the traffic propagation speed. The adversary's estimated RTT between its true/intended location and landmark l is:

$$\beta(a, l) = \frac{2 \times dis(a, l)}{(1/3)\mathbf{c}} \tag{3.12}$$

and

$$\beta(a', l) = \frac{2 \times dis(a', l)}{(1/3)\mathbf{c}} \tag{3.13}$$

where $dis(a, l)$ and $dis(a', l)$ are the great circle[3] geographic distances [58] between the adversary's true/intended location and landmark l. The distance (in the numerator) is doubled because β is a round-trip, rather than one-way, delay. To forge its location from a to a', the adversary sets δ in (3.5) as:

$$\delta = \beta(a, l) - \beta(a', l) \tag{3.14}$$

The difference between the adversary's estimated RTT and the actual contributes to the adversary's errors in forging location.

Adversary *A*. After evaluating this adversary, we compare its efficacy to three other classes of adversaries. We refer to the adversary described above as adversary A. Adversaries B, C and D have similar assumptions, except for the factors in Table 3.2. Note that in contrast to the *achieved ability* in the second column of the table, the third column presents an *assumed* knowledge.

Adversary *B*. Gill *et al.* [59] studied the effect of an adversary increasing the RTT observed by a measuring party by delaying response messages. To realize how much an adversary gains by also being able to decrease RTTs (as explained in Section 3.3), we implemented the manipulation tactic of Gill *et al.* [59],[4] which is equivalent to adversary B, to compare it with adversary A.

Adversary *C*. Similar to the modeled adversary of Gill *et al.* adversary B uses $(2/3)\mathbf{c}$ to model traffic speed, whereas A uses $(1/3)\mathbf{c}$. To understand whether B's retrogressions/improvements over A were due to its limited delay-manipulation abilities or its parameterization, we involve in the discussion adversary C which only differs from B in that it uses $(1/3)\mathbf{c}$ as the traffic speed.

Adversary *D*. We assume this adversary has the advantage of knowing the RTT between itself and each $l \in L$, $\alpha(a, l)$ (e.g., by *ping*ing each $l \in L$). However, it does not know the RTT between the landmarks and its intended location. To estimate it, D benefits from its knowledge of $\alpha(a, l)$, and calculates the traffic speed between itself and each $l \in L$ as follows:

$$\lambda_l = max\left(\frac{2 \times dis(a, l)}{\alpha(a, l)}, (2/9)\mathbf{c}\right) \tag{3.15}$$

[3] A great circle is one whose center and radius are those of the Earth.

[4] The results obtained from our implementation closely match those reported by Gill *et al.* [59]; we believe that any dissimilarities arise from differences in the data sets and the experimental environment.

The calculated speed λ_l reflects l's access network speed; it increases with fast access network, and decreases otherwise. Since l's calibrated delay-to-distance function (see Section 2.1.1, page 9) would have already been affected by the speed of its access network, using λ_l increases l's accuracy in calculating the distance between itself and D's intended location, i.e., in favor of the adversary. The lower bound $(2/9)\mathbf{c}$ in (3.15) is applied to avoid the effect of increased circuitousness (indirectness) and highly varying delay-to-distance ratios occurring with short distances over the Internet [139]. D then estimates the RTT that l should observe at the intended location, a' using the speed λ_l:

$$\beta(a', l) = \frac{2 \times dis(a', l)}{\lambda_l} \tag{3.16}$$

The adversary finally sets δ in (3.5) as:

$$\delta = \alpha(a, l) - \beta(a', l) \tag{3.17}$$

Adversary *E*. Proposed by Gill *et al.* [59], adversary E was assumed to have access to each landmark's calibration function, and was using this function to model the traffic propagation speed. It may not be trivial for an adversary to have access to such information in practice. Thus, we only list Adversary E in the table for completeness, but do not explore it further. Note however that having access to each landmark's calibration function makes the adversary stronger. We show later in this chapter that even when lacking knowledge of this information, some of the modeled adversaries are already powerful enough to control the forged location at a country-level granularity.

3.5 Evaluation Results

The primary evaluation metrics we use are the adversary's distance error and direction error (Fig. 3.3). The first is the distance between the adversary's intended location and the location calculated by the geolocation technique. We used, again, the great circle distance to calculate this metric. The second metric is the absolute spherical angle (i.e., $\leq$180) between the lines passing through both locations and the adversary' true location. We used spherical trigonometry to calculate this metric, where we adopted 6,371 km as an approximation to the Earth's radius [106].

We implemented three representative delay-based techniques for evaluating manip-

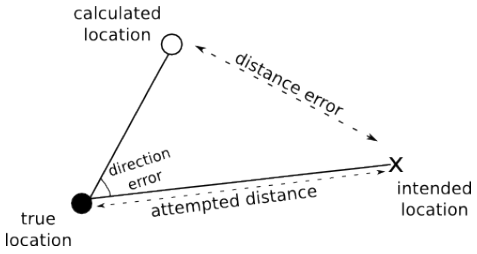


Figure 3.3: Distance and direction errors. The calculated location is the one returned by the geolocation technique, whereas the intended location is the one the adversary intends to appear at (fraudulently).

Figure 3.4: Locations of 122 landmarks and 51 modeled adversaries used in our experiments. Each adversary attempted to forge its location to 50 other (intended) locations, for a total of 2,550 modeled attempts to manipulate geolocation. ▲ = landmarks; • = true locations of adversaries; × = intended locations of adversaries. Note: this graph shows experimental design, not results. Map data: Google, INEGI.

ulations: GeoPing [113], CBG [67], and SegPoly [44], and believe analogous manipulation effect extends to other techniques. To evaluate the manipulations, we used PlanetLab [33], where we selected 144 nodes (Fig. 3.4) to represent 122 landmarks and 51 adversaries (some nodes acted as both). We obtained the delays between these nodes from the iPlane project [102], which were collected on March 27, 2014. Each client made 50 location-forging attempts, marked by × in Fig. 3.4, giving a total of 2,550 attempts.

Figure 3.5 shows a Cumulative Distribution Function (CDF) of the *attempted distances* (see Fig. 3.3 for definition); 50% of all attempts intended to move at least ∼7,600 km away from the true locations. Such large distances are not typically state or city level relocation, but rather country or continent level. In fact, we chose the centroids of 20 countries to represent 20 of the 50 intended locations, and the

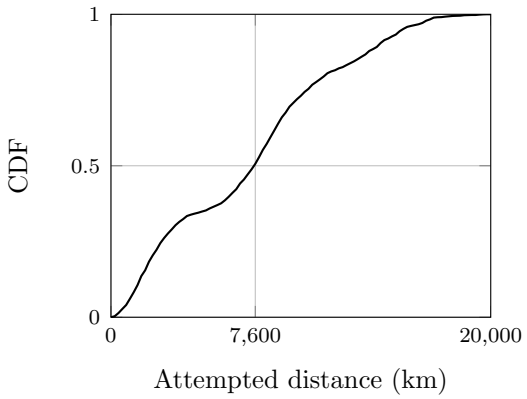


Figure 3.5: CDF of the attempted distances. A point (x,y) means a fraction y of all 2,550 manipulations attempted to move x km or less away from the true location. Note: this graph shows experimental design, not results.

centroids of 30 US states to represent the remaining ones.

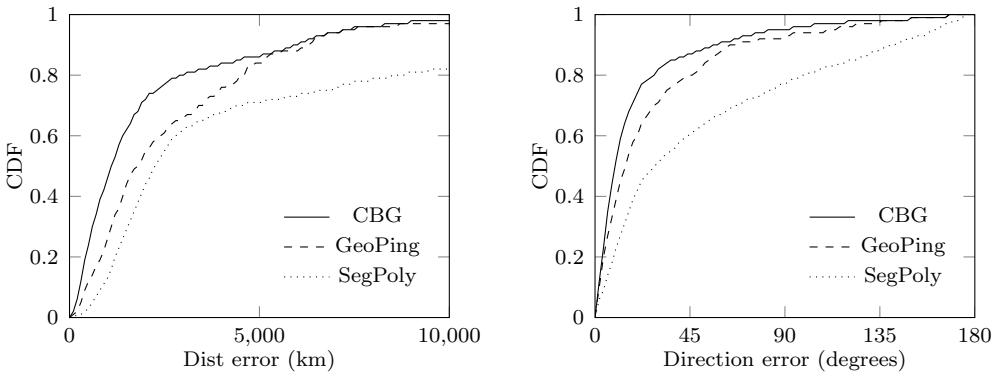
3.5.1 Manipulation Accuracy

Figure 3.6(a) shows a CDF of A's distance errors; one-third of manipulations to CBG resulted in errors below 700 km (close to the width of France), and two-thirds below 1,700 km.[5] Both values are less than half the US width; e.g., if Pandora [114] used CBG to enforce US geographic restriction policies, at least two-thirds of non US-based clients are expected to bypass these restrictions.

The adversary's distance errors were larger while manipulating GeoPing; one-fifth of all manipulations resulted in errors below 850 km, and half had errors below 1,800 km. The difference between CBG and GeoPing, however, partly stems from CBG being generally more accurate than GeoPing [67]. When the adversary can fully manipulate the delays, such higher accuracy may unfortunately help adversaries more accurately control the calculated location.

For SegPoly, 80% of manipulation attempts resulted in more than 1,200 km error, which is due to the linear function adversary A uses to map distances to delays (distance = delay $\times$ $(1/3)\mathbf{c}$). The function leads to a large deflection between the distance it wants a landmark to calculate, and the one the landmark actually calculates. Despite using linear mapping against a technique that uses polynomial

[5]Note that these adversarial errors arise in part due to inherent inaccuracies of the geolocation methods themselves, making the relatively smaller errors more noteworthy.



(a) (b)

Figure 3.6: CDF of (a) distance errors and (b) direction errors for adversary A upon manipulating geolocation. A point (x,y) means a fraction y of all manipulation attempts resulted in error of x (km or degrees) or less.

mapping, 44% of A's manipulations resulted in less than 2,000 km distance error.

The CDF of the direction error for the three techniques is shown in Fig. 3.6(b); 88%, 82%, and 63% of adversary A's manipulations to CBG, GeoPing and SegPoly respectively resulted in direction errors less than 50°. To interpret this result, one can think of a US bank restricting credit card transactions to the US, e.g., for fraud prevention [30,91]. Using CBG, and assuming relatively small distance errors, about 88% of European-based adversaries are expected to succeed to pretend to be in the (contiguous) US. That is because for most adversaries whose true locations are Europe (excluding Iceland) and who intend to be in the US, a direction error below ∼50° (and distance error below ∼5,000 km—the country's width) enables them achieve their objective. Figure 3.7 shows the spherical angle at the intersection point, close to the extreme west of Europe, of two lines enclosing the contiguous US. If a European-based adversary, for example, expects to incur a 50° direction error clockwise, it can plan to pretend to be in Florida so that its location ends up being calculated as Washington, and vice versa.[6] Note that the angle in Fig. 3.7 decreases when the two lines intersect further to the east of Europe.

Next we explore the relationship between A's attempted distance and its distance error. The correlation[7] between the two variables are 0.55 for CBG and GeoPing, and 0.68 for SegPoly. A powerful adversary should exhibit lower correlation be-

[6]The adversary is not assumed to control whether the direction error is clockwise or counter-clockwise; if one fails, it tries the other.

[7]The Pearson Correlation Coefficient ranges from -1 to +1, with 0 indicating no correlation, and ±1 indicating extreme +/-ve correlation.

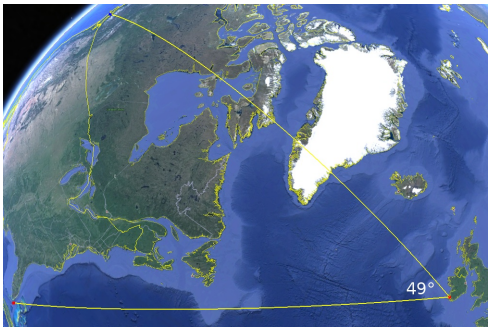


Figure 3.7: The spherical angle at the intersection point, close to the extreme west of Europe, of two lines enclosing the contiguous US is ∼49°. Map data: Google, SIO, NOAA, U.S. Navy, NGA, GEBCO.

tween both variables, meaning that its accuracy does not degrade when its intended location is far away from its true one. The relatively moderate correlation while manipulating CBG and GeoPing (0.55) indicates that an adversary able to increase and decrease RTTs can accurately control extremely remote fraudulent locations. Note that the correlation is positive because manipulations of small attempted distances result in small distance errors.

Manipulations to SegPoly experienced higher correlation, compared to CBG and GeoPing, because of the discrepancy between SegPoly's segmented polynomial mapping function and the linear function adversary A uses, which manifests quite clearly as larger delays get mapped to distances.

3.5.2 Manipulation Detection

CBG calculates a client's geographic location as the centroid of a convex region enclosed by the intersection of multiple circles. Gill *et al.* [59] suggested the area of this region could be used to detect manipulations, which involves an adversary increasing the RTT, because larger adversary-landmark RTTs increase the area. We analyze detection abilities of this against an adversary that can also decrease delays. GeoPing generates no intersection regions; we are not immediately aware of a method to precisely detect manipulations against GeoPing.

Figure 3.8 shows a CDF of the intersection-region areas while operating CBG and SegPoly to calculate the forged and true locations of adversaries.[8] The speed that adversary A uses, $(1/3)\mathbf{c}$, is slow relative to the average traffic propagation speed [85].

[8]These true locations are calculated from the original delays between the landmarks and the 51 PlanetLab nodes, before changing these delays to model adversaries.

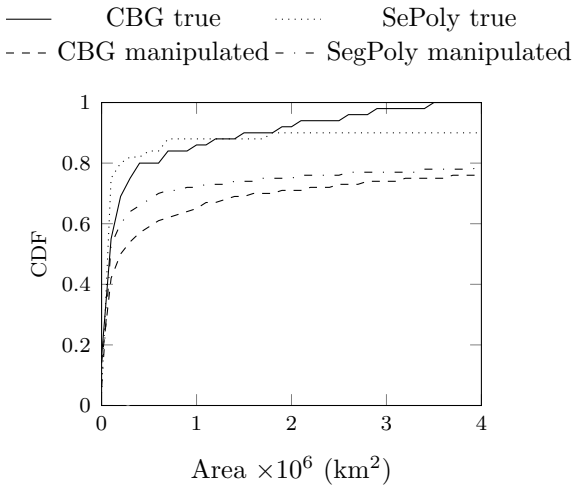


Figure 3.8: CDF of the intersection-region areas while determining the (51) *true* and the (2,550) *manipulated* locations. A point (x,y) means a fraction y of all attempts had areas of x km^2 or less. Higher (*manipulated*) curves indicate less detectable manipulations.

This results in relatively large RTT estimates to the intended location, $\beta(a', l)$ in (3.13), increasing the distances the landmarks calculate from mapping those RTTs. This explains the large areas (low curves in Fig. 3.8) while manipulating geolocation.

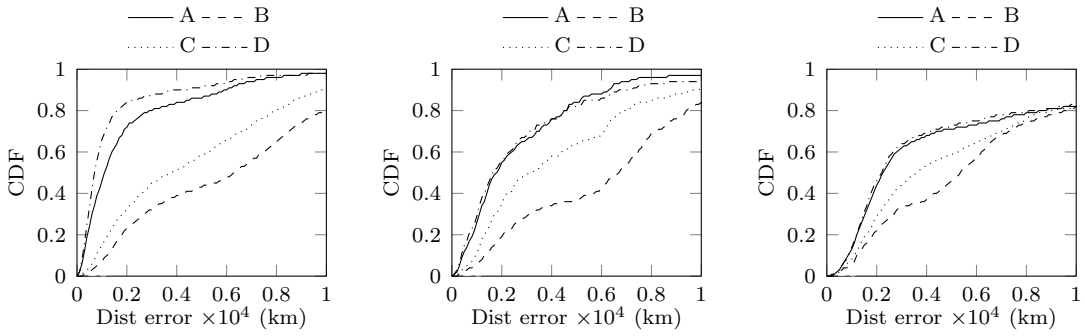
However, 92% of the areas that CBG calculated while locating the true nodes were equivalent to 71% of those while calculating the forged locations, at $x = 2{\times}10^6$ km^2. This implies that if the geolocating party decides to reject clients whose intersection-region areas are greater than this value, it falsely rejects 8% of legitimate clients and falsely accepts 71% of adversaries. According to these results, detecting manipulations based on the intersection-region areas is not trivial.

3.6 Comparing the Adversarial Models

We compare the adversaries modeled in Table 3.2 (Section 3.4). The same delay dataset was used across them to establish a comparable experimental set up.

3.6.1 Manipulation Accuracy

Figure 3.9 compares the distance errors for the four adversaries. About 80% of B's manipulations to the three techniques resulted in distance errors above ∼1,900 km; only 29%, 48% and 59% of A's manipulations to CBG, GeoPing, and SegPoly



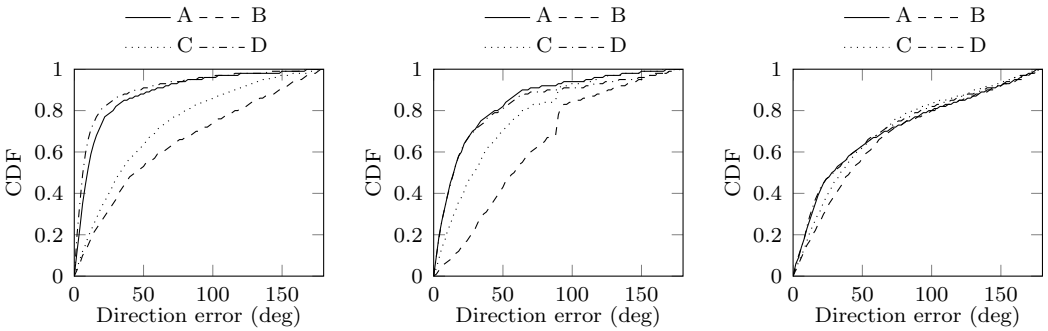
(a) CBG (b) GeoPing (c) SegPoly

Figure 3.9: Distance errors for the adversaries in Table 3.2.

respectively resulted in errors above 1,900 km. The median distance errors for A and C while manipulating CBG were 1,100 km and 3,800 km respectively. The corresponding values for GeoPing were 1,800 km and 3,100 km, and for SegPoly were 2,250 km and 3,600 km. The improvements of adversary A over C underscore the effectiveness of full delay manipulation on an adversary's location-forging abilities.

Adversary D shows a distance error improvement over A while manipulating CBG, with 66% of D's manipulations resulting in errors below 1,000 km, versus 46% of A's manipulations. Surprisingly, A showed slight improvement over D while manipulating GeoPing. One possible explanation for this could be D's access network; if it is relatively slow, the varying delay-distance mapping decreases the mapped delays between D and *all landmarks*. A constant traffic speed protects A from the effect of a slow access network. Finally, for SegPoly, almost unnoticeable distance error improvements were made by adversary D's manipulations over A.

Figure 3.10 compares the results for the direction errors; 50%, 38% and 53% of adversary B's manipulations to CBG, GeoPing and SegPoly respectively resulted in direction errors below 45°, versus 87%, 80% and 60% of adversary A's manipulations. A lower direction error for the adversary indicates a more accurate (hence, more worrisome) adversary. Similar to the distance errors, adversary C's overall direction error was better than that of B but worse than A, again highlighting A's devastating abilities. Adversary D showed direction error improvements over A only while manipulating CBG, but no considerable improvements were observed upon manipulating the other two geolocation techniques.



(a) CBG (b) GeoPing (c) SegPoly

Figure 3.10: Direction error for the adversaries in Table 3.2.

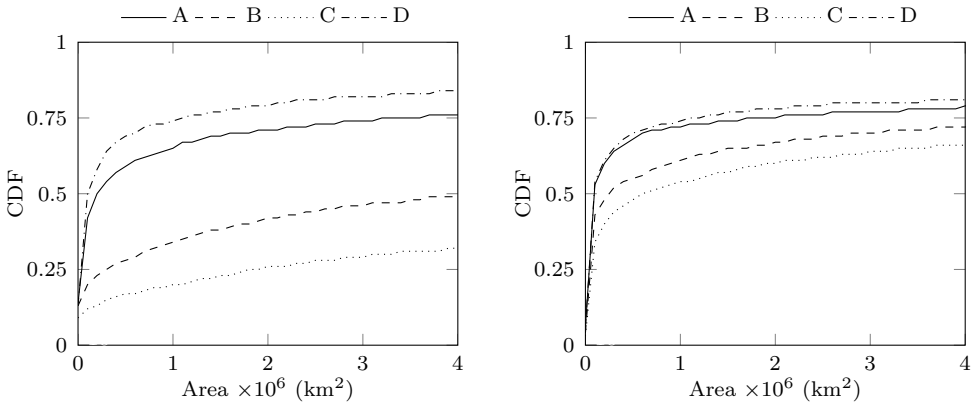
3.6.2 Manipulation Detection

Figure 3.11 shows CDFs of the intersection regions areas; 58% of B's manipulations to CBG resulted in areas above $2{\times}10^6$ km^2, versus only 29% of A's. Clearly, adversary A's manipulations to CBG are harder to detect using the area as the detection factor.

Areas resulting from B's manipulations to SegPoly were significantly smaller compared to its manipulations to CBG, and more interestingly, were close to those resulting from A's manipulations to SegPoly (the curves A and B are close in Fig. 3.11(b) than in Fig. 3.11(a)). This is because B uses double the speed that A uses to model the traffic speed. If both adversaries pretend to be farther from a landmark by a certain distance, B increases the RTT by half the amount that A increases. When the landmark maps those RTTs, smaller values get mapped to smaller distances, decreasing the area of intersection. Nonetheless, the average distances resulting from adversary B's mapping are not expected to be relatively small since B can only increase RTTs. B's faster traffic speed combined with its ability to only increase delays explains its area similarity with A. This argument does not apply to CBG because the linear calibration the landmarks use blindly maps larger delays to larger distances.

Adversary C had the largest intersection-region areas compared to A and B because it combines two factors that tend to increase areas: only increasing RTTs and using a small constant to model traffic speed. Therefore, C is the most exposed to being detected based on the area of the intersection region.

Adversary D was less detectable than A. Half of D's manipulations to CBG resulted in intersection-region areas below $0.1{\times}10^6$ km^2, compared to double this number for



(a) CBG (b) SegPoly

Figure 3.11: CDF of the intersection-region areas for the adversaries in Table 3.2. A point (x,y) means a fraction y of all attempts resulted in areas of x km^2 or less. Higher curves indicate less detectable manipulations.

Table 3.3: Median distance (km) & direction errors (degrees), median areas of intersection regions (km^2), and correlation coefficients between the distance errors and the attempted distances for the adversaries in Table 3.2; Adversary B is similar to that of Gill *et al.* [59]. Smaller values for all fields indicate a more powerful adversary.

Geolocation	Dist error (km)				Direction error (deg)				Area ×10^6 (km^2)				Correlation			
method	*A*	*B*	*C*	*D*	*A*	*B*	*C*	*D*	*A*	*B*	*C*	*D*	*A*	*B*	*C*	*D*
CBG [67]	1,100	6,300	3,800	700	9.5	44	33	6	2	4.1	17.3	<1	0.54	0.89	0.73	0.33
GeoPing [113]	1,800	6,700	3,100	1,630	14	58	29	14	–	–	–	–	0.55	0.86	0.64	0.57
SegPoly [44]	2,250	5,350	3,600	2,200	28	41	34	29	<1	<1	<1	<1	0.68	0.85	0.8	0.64

half of A's manipulations.

3.6.3 Summary

Table 6.2 summarizes the differences between the four modeled adversaries. Adversary A achieves 83%, 73% and 58% reductions[9] to the median distance errors over B while manipulating CBG, GeoPing, and SegPoly respectively; and achieves 71%, 41.94% and 37.5% over C while manipulating the three techniques. A's improvement over C is solely due to its ability to fully manipulate delays (since it is the only difference between them), highlighting the powerful nature of manipulation when an adversary is able to decrease and increase the RTTs.

[9]Percentage reduction = $\frac{\text{Median error of } B \text{ - Median error of } A}{\text{Median error of } B} \times 100$.

Compared to B, adversary C achieves 40%, 54%, and 33% improvement to the median distance error while manipulating CBG, GeoPing, and SegPoly respectively (see Table 6.2). The only difference between both adversaries is the constant they used to model traffic speed, suggesting that a clever modeling can by itself increase the adversary's location-forging accuracy drastically.

Adversary D achieves 36%, 9.4%, and 2.2% improvement to the median distance error over A while manipulating the three techniques respectively. Thus, against GeoPing and SegPoly, an adversary's knowledge of the RTTs between itself and the landmarks did not significantly improve the results.

Adversary D's manipulations to CBG resulted in the smallest intersection-region area, thus hiding manipulation attempts on CBG is easier when the adversary knows the RTTs between its true location and the landmarks. However, knowledge of this information did not reveal significant advantages in hiding manipulation attempts on SegPoly since all adversaries resulted in very small areas. These results highlight that the accuracy SegPoly gains using polynomial regression comes at the cost of lower ability to detect manipulations by the constrained region area.

Finally, from Table 6.2, it is evident that B and C exhibit the highest correlation between the attempted distance and distance error (i.e., poor performance). Adversaries A and D relax the correlation, enabling them to fraudulently relocate themselves at extremely remote locations with high accuracy. Thus, the combined ability of increasing and decreasing delays reduces the impact on the distance error when large distances are attempted.

3.7 Countermeasures

We discuss possible countermeasures that specifically aim at preserving the integrity of delay measurements used by a geolocation technique. We stress that the root cause of the vulnerabilities lies not in the ICMP utilities themselves but rather in (improperly) leveraging them to carry out a task (geolocation) for which they were not designed. A high level countermeasure would therefore be to avoid using ICMP-based utilities for geolocation. If ICMP-based utilities are to be used nonetheless, the following countermeasures could be considered. These measures require only the landmarks conducting geolocation to modify their network stacks.

As discussed in Section 3.3, the vulnerabilities lie in the adversary's ability to tamper with both the sending (s) and receiving (r) times of ICMP-based network utilities.

Every landmark must ensure the integrity of both parameters. Locally recording s enables a landmark to retrieve s from its memory instead of the DATA field of an echo-reply packet. Obviously, the landmark's own local memory is more reliable than an unprotected packet returned from the receiver/adversary. This precludes the adversary from undetectably tampering with the value of s. If a stateless implementation is desired, landmarks may use a Message Authentication Code (MAC) protecting s, the ICMP identifier, and the ICMP sequence number of the echo request; the landmarks can then place the MAC in the DATA field of the packet along with s (see Fig. 3.2(a)). Note that a landmark cannot include the TYPE and CHECKSUM fields in the MAC because the receiver must change them in the echo reply, as specified by RFC 792 [125]. The landmark can store the non-shared key of the MAC locally. A single key suffices for multiple sessions. Landmarks can then verify the integrity of their own timestamp s retrieved from a received echo reply.

As for the receiving time r, recall that a key factor for an adversary to beneficially manipulate it is the adversary's ability to measure the waiting time between a successive pair of echo requests. Consequently, randomizing the waiting times raises the bar for the adversary to accurately predict this time. Such a precaution is simple to implement as it may not necessarily require modifications to the local utilities (e.g., *ping* and *traceroute*). However, because the adversary calculates the average waiting time, this precaution does not stop the adversary from undetectably manipulating r; it only increases the adversary's error range. Another countermeasure to provide timestamp integrity is to include an element of randomness in the DATA field of echo requests, i.e., similar to DNS cache-poisoning countermeasures [138]. For all practical purposes, ample unpredictability should prevent the adversary from successfully issuing fraudulent echo replies, forcing it to wait for echo requests first.

Although the countermeasures presented in this section could be technically simple, we expect little community support to deploy modifications to the widely used generic network utilities for the sole purpose of hardening geolocation.

3.8 Related work

In 2010, Gill *et al.* [59] studied the effect of delay increases on topology-aware (see Section 2.1.2, page 11) and delay-based geolocation techniques, choosing one representative technique for each. They modeled two classes of adversaries: simple (controls only its own machine) and sophisticated (controls a full wide area network). The former was able to increase delays (adversaries B and E herein—Section 3.4)

and the latter was able to increase the number of hops to the landmarks. Against delay-based techniques, both adversaries had limited control over the forged location [59].

Muir *et al.* [107] investigated geolocation over the Internet from a security perspective, and enumerated a broad spectrum of tactics for an adversary to manipulate geolocation techniques, including using proxies to hide the IP address, and falsifying location records of public registries (*whois* databases, DNS LOC records [37], etc). They argued that despite a plethora of proposals to geolocate Internet hosts, none appears to be robust against all classes of adversaries. Our work is complementary as it provides concrete evidence, based on practical evaluations, supporting their assertion with respect to popular implementations of delay-based geolocation techniques.

Goldberg *et al.* [61] addressed the problem of path quality monitoring, devising protocols to detect if an adversary sitting in the path between two end systems is manipulating their traffic. Although their research is motivated, in part, by the lack of integrity checking in network-monitoring utilities, their proposed solutions assume the collaboration of the two end systems. In our case, one of the end systems is the adversary itself and therefore collaboration cannot be assumed (hence: none of their solutions fit the problem studied herein).

Delay-based location verification techniques have been proposed [10,127,160]. However, proposals for single-hop wireless networks [127,160] cannot be directly applied to the Internet because of the difference in delay nature between both domains [59].

In Network Coordination Systems (NCSs), such as Vivaldi [36] and Meridian [152], network nodes are assigned coordinates according to the delays between them. NCSs are generally seen as different from geolocation because the coordinates of a node reflect its *network location* rather than geographic longitude and latitude; thus, no delay-to-distance mapping is required. Adversarial environments (e.g., to disrupt an NCS) were explored [60,157], and proposals for securing NCSs addressed adversarial delay-increase [84].

3.9 Conclusion

Virtually all current implementations of conventional network utilities (e.g., *ping* and *traceroute*) fail to check the integrity of the measured RTTs. Thus, misusing them for delay-based geolocation allows an adversary of moderate abilities to in-

crease *or decrease* the RTTs observed by the measuring party. Without controlling any devices or network traffic other than its own, the adversary can then manipulate geolocation techniques that are based on active delay measurements to the extent of accurately controlling the calculated (forged) location, e.g., for a resulting error as small as 100 km—providing country-level granularity control.[10] This may defeat location-aware security systems, e.g., a cloud provider violating service level agreements [62], especially given that such geolocation techniques are increasingly being advocated for use in security-aware contexts [31].

By evaluating several adversarial situations, we have demonstrated that better estimates to the traffic propagation speed can enhance the adversary's accuracy in controlling the forged location. This finding even extends to adversaries only capable of increasing RTTs [59]; e.g., an adversary that uses the constant $(1/3)\mathbf{c}$ as an estimate to the traffic speed, as shown herein, is 40% more accurate in forging a CBG-calculated location [67] than the one using $(2/3)\mathbf{c}$ studied in previous literature [59].

We note there are countermeasures, based on well-known and technically simple techniques, which provide integrity to the timing information exploited by the manipulations we discussed in Section 3.3, and thereby would (if implemented and deployed) preclude the evasion of geolocation that we analyzed in Sections 3.5 and 3.6. However, these add overhead to core ICMP utilities, and thus may well face deployment resistance since they are unnecessary for core services. Designers of delay-based geolocation usually focus on achieving high location accuracy, but to date have failed to propose integrity-preserving yet deployable delay-measurement algorithms—despite being motivated by security-sensitive applications [14,44,67,91].

The analysis in this chapter provides some useful insights. For example, landmarks in CBG [67] would ideally allow only themselves to measure RTTs between each other; in our experiments, an adversary knowing the RTTs between itself and the landmarks was 36% more accurate. Additionally, if SegPoly [44] is used, the areas of the constrained region cannot be relied upon for detecting manipulations since they become considerably smaller. In fact, security-sensitive applications [62] should not rely on the constrained region areas for detecting manipulations because, while geolocating adversaries who can fully manipulate delays, the constrained regions become almost indistinguishable from those of legitimate clients.

[10] Related to geolocating cloud data, Peterson *et al.* [118] emphasize: "Of particular interest is establishing data location at a granularity sufficient for placing it within the borders of a particular nation-state."

Our work highlights the importance of ensuring timing integrity in delay-based geolocation. We believe more investigations into these techniques are required, and expect that the search for a geolocation mechanism which is not easily defeated remains challenging. We hope our work raises awareness of the importance of devising such evasion-resilient geolocation mechanisms, and encourages further research in this area.

Chapter 4

Accurate One-Way Delay Estimation with Reduced Client-Trustworthiness

This chapter proposes a novel protocol that enables a server to estimate OWDs between itself and a client by cooperating with two other servers, requiring neither client-clock synchronization nor client trustworthiness in reporting one-way delays. Due to these benefits, the proposed protocol is of value to, and is used in, the location verification mechanism introduced later in Chapter 5.

We evaluate the protocol by deriving the probability distribution of its absolute error, and compare its accuracy with the well-known round-trip halving protocol. While neither protocol requires client-trustworthiness nor client clock synchronization, the analysis shows that the new protocol is more accurate in many situations.

4.1 Introduction

Delay-dependent applications can benefit from accurate OWD-estimation mechanisms [32]. For example, measurement-based geolocation (see Chapter 2) may be

The first part of this chapter, the *minimum pairs* protocol, was published at the 2014 IEEE CNS conference [10] (with a full length version accepted for publication in IEEE TDSC [9]). The second part, the evaluation using probability models, was published at the IEEE Communications Letters [11].

performed with a greater precision if accurate OWD-estimates were relied upon, rather than RTTs. The common methods that enable a server to accurately measure one-way delays to/from a client [134] rely on the client's honest cooperation—the client is assumed to synchronize its clock accurately with the server, calculate and honestly report its view of the delays.[1]

Because RTTs are easier to estimate than OWDs, half the RTTs—or the *average* (*av*) of the actual forward and reverse OWDs—are often use as OWD-estimates [159]. However, the asymmetric nature of Internet routes [116] highlights the potential for OWDs to improve the efficacy of such delay-dependent applications [70]. Compared to half the RTTs, OWDs are more likely to exclude noisy delay components caused by, e.g., congested or circuitous routes because the delay is measured in one direction.

A delay-based location verification mechanism requires a combination of both: accurate delay-estimation and minimal client cooperation. The accuracy is required to reduce false client rejects and false adversarial accepts as much as possible, while the minimal cooperation is required to reduce the adversary's attack surface. Using common OWD-estimation methods (e.g., OWAMP [134]) allows dishonest clients to forge delay-estimates, and using half the *av* protocol is expected to result in incorrect decisions.

This chapter introduces a new protocol, *minimum pairs* (*mp*), which allows a server to estimate OWDs between itself and a client by mainly cooperating with two other servers , while requiring less client cooperation than classical OWD-estimation protocols; e.g., neither client-clock synchronization nor client trustworthiness in reporting OWDs is required by the *mp* protocol. The required client cooperation is similar to that required by the *av* protocol (i.e., responding to echo-request messages for measuring RTTs). These features make *mp* more suitable for location verification, as we show in Chapter 5.

The *mp* protocol is evaluated by deriving the probability distribution of its absolute error. Because the protocol's client-cooperation requirements are similar to that of the *av*, we similarly derive the probability distribution of error for the *av* protocol, and compare both protocols assuming a Poisson delay-distribution. While neither protocol requires client-trustworthiness nor client clock synchronization, the analysis shows that the *mp* protocol provides more accurate OWD-estimates than *av* in many

[1] Although OWDs are generally measured between peers, we use the server/client terminology to discriminate between the party measuring the delays (server) and the one the delays are measured to/from (client).

situations.

This chapter makes the following contributions:

- Proposing the *minimum pairs* (*mp*) protocol for accurate OWD-estimation, which is to be used later in Chapter 5 for location verification.
- Deriving the Probability Mass Function (PMF) of the absolute error for the *mp* and the *av* protocols as a function of the delay distribution between the client and the servers.
- Using the derived probability models to compare the accuracy of both protocols assuming Poisson delay distribution with various representative means. This example comparison can now be drawn since the derived models allow general determination of the more accurate protocol given the probability distribution of delays; Poisson is used as an example.

The rest of this chapter is organized as follows. Section 4.2 explains the threat model. Section 4.3 presents the *mp* protocol, while Sections 4.4 and 4.5 derive the PMF of absolute errors for the *av* and *mp* protocols respectively. Section 4.6 provides an example of comparing the accuracy of both protocols assuming Poisson delay distribution with various means. Section 8.5 concludes.

4.2 Threat model

Recall that in OWAMP-like protocols, OWDs between a server and a client are estimated by having them synchronize their clocks together, and exchange timestamps. The server can calculate the OWD (at some moment) only in the direction *client-to-server* by subtracting the timestamp that the client sends from the time the stamp was received; the client does the same procedure for calculating OWDs in the reverse direction, and informs the server with the calculated OWDs.

Because the *mp* protocol is designed to address possibly dishonest clients, it must assume the client is able to:

1. Refrain from appropriately synchronizing its clock with the server;
2. Falsify OWDs before informing the server about them, during the estimation of *server-to-client* OWDs;
3. Falsify the timestamps before sending them, during the estimation of *client-to-server* OWDs; or

4. Delay or reject timestamp messages.

As we explain in the next section, the *mp* protocol neither relies on the client's clock, nor on any information reported by the client. Thus, the first three threats do not affect *mp*. In the next chapter, we show how the location verification mechanism itself handles the fourth threat—delaying or rejecting timestamp messages.

4.3 The Minimum Pairs Protocol

The *mp* protocol is designed to estimate the smaller of the forward and reverse OWDs at current network conditions. The larger OWD can then be estimated as the difference between the smaller and the RTT. However, we discard the larger delay between the two parties since the smaller provides a more accurate estimate to the distance between them; the larger delay must have been influenced by route congestion, circuitousness [153] (see Section 2.1.1, page 9), or other noisy circumstances that increase delays.

To use *mp*, three servers must cooperate together. These servers will be the ones implementing the location verification algorithm later in Chapter 5, and will be referred to as *verifiers*. To simplify the discussion, we refer to them as *verifiers* from this point on. We assume each of the three verifier possesses a public-private key pair, and is aware of the public keys of the other two verifiers, possibly through a closed Public Key Infrastructure (PKI).

Notation There are three bidirectional edges joining a client with three verifiers, and three bidirectional edges joining the three verifiers, as shown in Fig. 4.1. Each of the six edges has two OWDs in opposite directions. Denote $\mathbf{D}^{\bullet}$ as an ordered list holding six OWD estimates at a given time. The estimates correspond to the smaller of the forward and reverse OWDs (i.e., at current network conditions) at each of the six bidirectional edges in Fig. 4.1. The superscript $\bullet$ is the protocol used to estimate the delays in $\mathbf{D}^{\bullet}$.

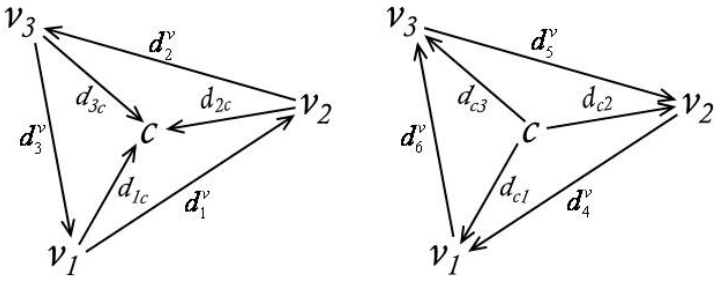


Figure 4.1: Notation of OWDs between client c and verifiers v_1, v_2 and v_3.

Table 4.1: Notation

Notation	Description
$S_a(m)$	denotes message m digitally signed by entity a.
$A \xrightarrow{m} B$	A sends message m to B.
t_a	the most recent timestamp according to verifier a's clock.
e_{ij} (line 10)	corresponds to $d_{ic} + d_{cj}$ (see Fig. 4.1).

4.3.1 Protocol description

When requesting a location-sensitive service from an LSP, the LSP notifies the client of the IP addresses of a set, V, of three verifiers, which the client must connect to[2] in order to have their location verified. Details about how the verifiers are chosen are discussed in Chapter 5.

Algorithm 1 explains the *mp* protocol; see Table 4.1 for notation used in the algorithm. Note that the location verification protocol presented in Chapter 5 also relies on *av* as an alternative OWD-estimation protocol. For convenience, Algorithm 1 also calculates OWD-estimates following the *av* protocol.

Algorithm Explanation The three verifiers take turns to send the client digitally signed timestamps of their most recent system time (line 3). Once received, the client is required to forward this message to the three verifiers.[3]

When all three verifiers are done their turns, they will have nine values of delays corresponding to $d_{ic}+d_{cj}$ for all $1 \leq i, j \leq 3$. The *mp* protocol estimates the smaller of d_{ic} and d_{ci} independently, for all $1 \leq i \leq 3$, as follows. First, for all $1 \leq i, j \leq 3$ and $i \neq j$, the larger of $d_{ic}+d_{cj}$ and $d_{jc}+d_{ci}$ is discarded (line 13) because the smaller

[2]The client and the verifiers may use websockets [50] to connect to the verifiers, as they are a stable means of delay measurement through the browser [95].

[3]This behavior can be implemented in the browser through javascript.

Algorithm 1: The *mp* protocol. See notation inline.

Input: The set of the three verifiers, V (see Fig. 4.1).
Output: $\mathbf{D}^{mp}$ and $\mathbf{D}^{av}$
begin

foreach v_i **in** V **do**
  v_i retrieves its current system time $b := t_i$
  $v_i \xrightarrow{b, S_i(b)}$ client
  foreach v_j **in** V **do**
    client $\xrightarrow{b, S_i(b)} v_j$
    v_j records the message-receiving time $r := t_j$
    v_j validates $S_i(b)$
    if *invalid signature* **then**
      Abort "possible client cheating attempt"
    $e_{ij} := r - b$
for $i := 1$ **to** 6 **do**
  The verifiers in V measure d_i^v (see Fig. 4.1)

`/* Calculating` $\mathbf{D}^{mp}$ `*/`
$m := \{min(e_{12},\ e_{21}),\ min(e_{23},\ e_{32}),\ min(e_{31},\ e_{13})\}$
for $i := 1$ **to** 3 **do**
  $j = ((i + 1) \bmod 3) + 1$
  $k = (i \bmod 3) + 1$
  $x_i := (m_i + m_j - m_k)/2$
  $y_i := min(d_i^v, d_{i+3}^v)$
  Append x_i and y_i to $\mathbf{D}^{mp}$

`/* Calculating` $\mathbf{D}^{av}$ `*/`
for $i := 1$ **to** 3 **do**
  $x_i := e_{ii}/2$
  $y_i := (d_i^v + d_{i+3}^v)/2$
  Append x_i and y_i to $\mathbf{D}^{av}$

return $\mathbf{D}^{mp}$ and $\mathbf{D}^{av}$

sums are likely to correspond to the smaller OWDs. Second, the three remaining sums are equated to the corresponding smaller OWDs, and estimates to the smaller delays are obtained by solving simultaneously for x_1, x_2, x_3:

$$x_i + x_j = min(d_{ic} + d_{cj},\ d_{jc} + d_{ci}) \quad \forall\ 1 \leq i < j \leq 3$$

where x_i is the estimate to the smaller of d_{ic} and d_{ci}. To work out these equations, let m_1 be the minimum between $d_{1c}+d_{c2}$ and $d_{2c}+d_{c1}$; similarly, m_2 is the minimum between $d_{2c}+d_{c3}$ and $d_{3c}+d_{c2}$; and m_3 is the minimum between $d_{3c}+d_{c1}$ and $d_{1c}+d_{c3}$. The simultaneous equations are:

$x_1 + x_2 = m_1$
$x_2 + x_3 = m_2$
$x_3 + x_1 = m_3$

Solving these equations yields:

$x_1 = (m_1 + m_3 - m_2)/2$
$x_2 = (m_2 + m_1 - m_3)/2$
$x_3 = (m_3 + m_2 - m_1)/2$

This is demonstrated in lines 13 to 17 of Algorithm 1.

Discarding the larger delays (line 13) provides a fundamental advantage to *mp* over *av*, as it helps reduce the unfavourable effect of delay spikes occurring in one direction but not the other. Compared to *av*, the probability of *mp* to exclude delay spikes is higher.

In line 12, estimating the smaller OWDs of the edges between the verifiers (i.e., d_i^v in Fig. 4.1) is simpler, since the verifiers trust each other; for example, the OWAMP [134] tool can be used. Again, the verifiers discard the larger of the forward and reverse OWDs for each of the three edges between them (line 18). Finally, the set $\mathbf{D}^{mp}$ holds the six smaller OWD estimates (line 19).

4.3.2 Clock synchronization among the verifiers

In *mp*, the verifiers may choose to synchronize their clocks to the nearest millisecond to increase the accuracy of OWD estimates [34, 38], or use techniques that do not require accurate synchronization [98, 145]. For example, Gurewitz *et al.* [69] proposed a technique that estimates OWDs in the absence of accurate clock synchronization between network nodes. Strong cooperation between these nodes is, however, required. The nodes conduct many OWD measurements among themselves using the poorly synchronized clock, and use those preliminary estimates to derive constraints of an objective function. The function uses optimization techniques,

and reaches a per-link OWD estimate that minimizes the error with respect to the provided constraints.

While this class of techniques addresses imperfect clock synchronization, the *mp* protocol addresses client untrustworthiness. Therefore, such a class of techniques can be used among the verifiers if accurate clock synchronization cannot be achieved. However, due to its strong cooperation and trustworthiness requirement, it cannot be used with potentially dishonest clients.

4.4 Analyzing the Average Protocol (av)

In this section, the PMF of absolute error is derived for the *av* protocol. The *absolute error* is the absolute difference between the smaller of the forward and reverse OWDs and the OWD estimated by the protocol. Let $f_x(d)$ be the PMF of the delay of edge d, for each of the six bidirectional edges in Fig. 4.1.

Throughout this section (and Section 4.5), we focus on the OWDs between the client and verifier v_1 in Fig. 4.1. Similar analysis applies to the other two bidirectional edges.

4.4.1 Absolute error of av

The *av* protocol estimates the smaller OWD between v_1 and c as:

$$t^{av} = \frac{\text{RTT}}{2} = \frac{d_{1c} + d_{c1}}{2} \tag{4.1}$$

The absolute error of the *av* protocol is:

$$\varepsilon^{av} = |t^{av} - min(d_{1c}, d_{c1})|$$

The magnitude of the error thus depends on the difference between d_{1c} and d_{c1}. Table 4.2 lists the three cases. Denoting by ε_i^{av} the error in Case i, then:

$$\varepsilon_1^{av} = \left| \frac{d_{1c} + d_{c1}}{2} - d_{1c} \right| = \frac{d_{c1} - d_{1c}}{2}$$

We can drop the *"absolute"* sign ($||$) because in Case 1, $d_{1c} < d_{c1}$. The error for the remaining two cases is given in Table 4.2.

Table 4.2: Cases relating d_{1c} with d_{c1}, the calculated delay (t^{av}) in each case, and the error (ε^{av}) of the av protocol.

Case (i)	Condition d_{1c} [relation] d_{c1}	t_i^{av}	ε_i^{av}
1	$<$	$(d_{1c}+d_{c1})/2$	$(d_{c1}-d_{1c})/2$
2	$=$	$(d_{1c}+d_{c1})/2$	0
3	$>$	$(d_{1c}+d_{c1})/2$	$(d_{1c}-d_{c1})/2$

4.4.2 PMF of error for av

The PMF of ε_i^{av} depends on the probability of occurrence of Case i. Thus, for all $x \geq 0$:

$$\begin{aligned} P\{\varepsilon^{av}=x\} &= \sum_{i=1}^{3} P\{\text{Case } i\} \cdot P\{\varepsilon_i^{av}=x \mid \text{Case } i\} \\ &= \sum_{i=1}^{3} P\{\text{Case } i\} \cdot \frac{P\{\varepsilon_i^{av}=x\ ,\ \text{Case } i\}}{P\{\text{Case } i\}} \\ &= \sum_{i=1}^{3} P\{\varepsilon_i^{av}=x\ ,\ \text{Case } i\} \end{aligned} \tag{4.2}$$

where the *"comma"* indicates the intersection of the two events. Expanding the term at $i=1$ yields:

$$\begin{aligned} P\{\varepsilon_1^{av}=x\ ,\ \text{Case 1}\} &= P\left\{\frac{d_{c1}-d_{1c}}{2}=x\ ,\ d_{1c}<d_{c1}\right\} \\ &= P\{d_{c1}=2x+d_{1c}\ ,\ d_{1c}<d_{c1}\} \\ &= P\{d_{c1}=2x+d_{1c}\ ,\ d_{1c}<2x+d_{1c}\} \\ &= P\{d_{c1}=2x+d_{1c}\ ,\ x>0\} \\ &= \left(\sum_{i=0}^{\infty} P\{d_{1c}=i\}\cdot P\{d_{c1}=2x+i\}\right)\cdot P\{x>0\} \\ &= \begin{cases} \sum_{i=0}^{\infty} f_i(d_{1c})\cdot f_{2x+i}(d_{c1}), & x>0 \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{4.3}$$

Since $\varepsilon_2^{av} = 0$ (see Table 4.2), therefore,

$$P\{\varepsilon_2^{av} = x\ ,\ \text{Case 2}\} = P\{x = 0\ ,\ d_{1c} = d_{c1}\}$$
$$= \begin{cases} P\{d_{1c} = d_{c1}\}, & x = 0 \\ 0, & \text{otherwise} \end{cases}$$

where:

$$P\{d_{1c} = d_{c1}\} = \sum_{i=0}^{\infty} f_i(d_{1c}) \cdot f_i(d_{c1})$$

The term for $i = 3$ in (4.2), $P\{\varepsilon_3^{av} = x\ ,\ \text{Case 3}\}$, can be expanded analogous to Case 1. We thus rewrite (4.2) as:

$$P\{\varepsilon^{av} = x\} = \begin{cases} P\{d_{1c} = d_{c1}\}, & x = 0 \\ P\{\varepsilon_1^{av} = x\ ,\ \text{Case 1}\} + P\{\varepsilon_3^{av} = x\ ,\ \text{Case 3}\}, & x > 0 \end{cases}$$
$$= \begin{cases} \sum_{i=0}^{\infty} f_i(d_{1c}) \cdot f_i(d_{c1}), & x = 0 \\ \sum_{i=0}^{\infty} f_i(d_{1c}) \cdot f_{2x+i}(d_{c1}) + \sum_{i=0}^{\infty} f_i(d_{c1}) \cdot f_{2x+i}(d_{1c}), & x > 0 \end{cases} \quad (4.4)$$

4.5 Analyzing the Minimum Pairs Protocol (mp)

In this section, the PMF of absolute error is derived for the mp protocol. Again, we focus our analysis on the OWDs between the client and v_1. Throughout the section, the notation d_{ij}^{+} is used to denote $d_{ic} + d_{cj}$; likewise, d_{ij}^{-} denotes $d_{ic} - d_{cj}$.

4.5.1 Absolute error of mp

In Algorithm 1, lines 17 to 19 define three simultaneous equations that estimate the smaller OWD (t^{mp}). Although the mp protocol does not enable the verifiers to calculate d_{ii}^{-} for all $i \in \{1, 2, 3\}$, it enables them to sort these differences. For example, assume in line 17 that $d_{2c} + d_{c1} \leq d_{1c} + d_{c2}$. Rearranging yields $d_{22}^{-} \leq d_{11}^{-}$. Also assuming in line 18 that $d_{3c} + d_{c2} < d_{2c} + d_{c3}$ (equivalent to $d_{33}^{-} < d_{22}^{-}$), the verifiers can deduce that $d_{33}^{-} < d_{22}^{-} \leq d_{11}^{-}$.

The order of d_{11}^{-}, d_{22}^{-} and d_{33}^{-} identifies the cases in Table 4.3; possible outcomes of the $min()$ function in lines 17 to 19 are indicated at the header of the *"Conditions"*

Table 4.3: Cases relating d_{ij}^{+} with d_{ji}^{+}, the calculated delay in each case (t_i^{mp}), and the absolute error (ε^{mp}) of the mp protocol. In each Case, a circled condition is implied by the other two.

Case (i)	Conditions	Order	t_i^{mp}	$\varepsilon_{i,j}^{mp}$	
	d_{31}^{+} [relation] d_{13}^{+} d_{21}^{+} [relation] d_{12}^{+} d_{32}^{+} [relation] d_{23}^{+}			$d_{1c} \leq d_{c1}$	$d_{1c} > d_{c1}$
1	(<) $\leq$ $<$	$d_{33}^{-} < d_{22}^{-} \leq d_{11}^{-}$	$d_{c1} + d_{22}^{-}/2$	$\lvert d_{22}^{-}/2 - d_{11}^{-}\rvert$	$\lvert d_{22}^{-}/2\rvert$
2	$<$ (<) $\geq$	$d_{22}^{-} \leq d_{33}^{-} < d_{11}^{-}$	$d_{c1} + d_{33}^{-}/2$	$\lvert d_{33}^{-}/2 - d_{11}^{-}\rvert$	$\lvert d_{33}^{-}/2\rvert$
3	$\leq$ $>$ (<)	$d_{33}^{-} \leq d_{11}^{-} < d_{22}^{-}$	$d_{11}^{+}/2$	$-d_{11}^{-}/2$	$d_{11}^{-}/2$
4	$=$ $=$ (=)	All three are equal	$d_{11}^{+}/2$	$-d_{11}^{-}/2$	$d_{11}^{-}/2$
5	$\geq$ $<$ (>)	$d_{22}^{-} < d_{11}^{-} \leq d_{33}^{-}$	$d_{11}^{+}/2$	$-d_{11}^{-}/2$	$d_{11}^{-}/2$
6	$>$ (>) $\leq$	$d_{11}^{-} < d_{33}^{-} \leq d_{22}^{-}$	$d_{1c} - d_{33}^{-}/2$	$\lvert -d_{33}^{-}/2\rvert$	$\lvert d_{11}^{-} - d_{33}^{-}/2\rvert$
7	(>) $\geq$ $>$	$d_{11}^{-} \leq d_{22}^{-} < d_{33}^{-}$	$d_{1c} - d_{22}^{-}/2$	$\lvert -d_{22}^{-}/2\rvert$	$\lvert d_{11}^{-} - d_{22}^{-}/2\rvert$
	d_{33}^{-} [relation] d_{11}^{-} d_{22}^{-} [relation] d_{11}^{-} d_{33}^{-} [relation] d_{22}^{-} Rearranged Conditions				

column, with their rearrangements indicated at the bottom. Two conditions imply the third; the implied condition is circled in Table 4.3.

The smaller between d_{1c} and d_{c1} is indicated by the t_i^{mp} column in Table 4.3. In Case 1 for example, where $d_{31}^{+} < d_{13}^{+}$, $d_{21}^{+} \leq d_{12}^{+}$, and $d_{32}^{+} < d_{23}^{+}$, the simultaneous equations of lines 17 to 19 will be $\beta_1 + \beta_2 = d_{21}^{+}$, $\beta_2 + \beta_3 = d_{32}^{+}$, and $\beta_3 + \beta_1 = d_{31}^{+}$. In Algorithm 1, β_1 is returned as the estimate to the smaller between d_{1c} and d_{c1}, which evaluates to:

$$\begin{aligned} t_1^{mp} = \beta_1 &= \frac{d_{21}^{+} + d_{31}^{+} - d_{32}^{+}}{2} \\ &= \frac{d_{2c} + d_{c1} + d_{3c} + d_{c1} - (d_{3c} + d_{c2})}{2} \\ &= \frac{d_{2c} - d_{c2} + 2d_{c1}}{2} = d_{c1} + \frac{d_{22}^{-}}{2} \end{aligned}$$

Similarly, t_i^{mp} can be calculated for the remaining cases.

The returned OWD estimate (t^{mp}) can indicate whether there were large delay asymmetries between each verifier and the client. For example, if $t^{mp} < 0$, then the difference between the forward and reverse delays of some links between the client and the verifiers is relatively large.

4.5.2 Comparison between t^{mp} and t^{av}

As is now shown, in none of the seven cases will the mp protocol return a larger estimate to the smaller OWD than that of the av protocol; that is, the inequality

$t_i^{mp} \leq t^{av}$ holds for all $i \in \{1..7\}$. In Case 1, we have (Table 4.3):

$$t_1^{mp} = d_{c1} + \frac{\bar{d_{22}}}{2} \tag{4.5}$$

Since $\bar{d_{22}} \leq \bar{d_{11}}$ in this case (second rearranged condition, bottom of the *"Conditions"* column in Table 4.3), therefore:

$$t_1^{mp} \leq d_{c1} + \frac{\bar{d_{11}}}{2}$$

Simplifying yields

$$t_1^{mp} \leq \frac{d_{1c} + d_{c1}}{2} = t^{av} \qquad \text{from (4.1)}$$

Analogous analysis applies to Cases 2, 6 and 7, which we omit for conciseness. The equation $t_i^{mp} = t^{av}$ already holds for $i \in \{3, 4, 5\}$ (see Table 4.3). Thus, the mp protocol never returns an estimate, to the smaller between the forward and reverse OWDs, that is larger than that of the av protocol.

4.5.3 PMF of error for mp

The PMF of error depends on the probability of occurrence of each case in Table 4.3, and the probabilities of $d_{1c} \leq d_{c1}$ and $d_{1c} > d_{c1}$ in each case. We index those two additional conditions using the variable $j \in \{1, 2\}$ respectively. For example, to calculate the error in Case 1 given additional condition 2 (which is $d_{1c} > d_{c1}$):

$$\varepsilon_{1,2}^{mp} = |t_1^{mp} - min(d_{1c}, d_{c1})| = \left| d_{c1} + \frac{\bar{d_{22}}}{2} - d_{c1} \right| = \left| \frac{\bar{d_{22}}}{2} \right|$$

The probability that the error is equal to x is the probability that any of the expressions listed under the $\varepsilon_{i,j}^{mp}$ column in Table 4.3 evaluates to x, for all $x \geq 0$. The PMF of the absolute error can, thus, be expressed as:

$$\begin{aligned} P\{\varepsilon^{mp} = |x|\} &= \sum_{i=1}^{7} \sum_{j=1}^{2} P\{X_{i,j}\} \cdot P\{\varepsilon_{i,j}^{mp} = |x| \ \mid \ X_{i,j}\} \\ &= \sum_{i=1}^{7} \sum_{j=1}^{2} P\{X_{i,j}\} \cdot \frac{P\{\varepsilon_{i,j}^{mp} = |x| \ , \ X_{i,j}\}}{P\{X_{i,j}\}} \\ &= \sum_{i=1}^{7} \sum_{j=1}^{2} P\{\varepsilon_{i,j}^{mp} = |x| \ , \ X_{i,j}\} \end{aligned} \tag{4.6}$$

where $X_{i,j}$ is the intersection of all three conditions under the *"Conditions"* column of Case i with additional condition j. Because the error, $\varepsilon_{i,j}^{mp}$, in each of those 14 cases is the absolute difference, then:

$$P\{\varepsilon_{i,j}^{mp} = |x| \ , \ X_{i,j}\} = \begin{cases} P\{\varepsilon_{i,j}^{mp} = 0 \ , \ X_{i,j}\}, & x = 0 \\ P\{\varepsilon_{i,j}^{mp} = x \ , \ X_{i,j}\} + P\{\varepsilon_{i,j}^{mp} = -x \ , \ X_{i,j}\}, & \text{otherwise} \end{cases} \tag{4.7}$$

At $i = 1$ and $j = 1$, the event $X_{1,1}$ is (from Table 4.3):

$$X_{1,1} = (d_{31}^{+} < d_{13}^{+}) \ \cap \ (d_{21}^{+} \leq d_{12}^{+}) \ \cap \ (d_{32}^{+} < d_{23}^{+}) \ \cap \ (d_{1c} \leq d_{c1})$$

The condition $d_{31}^{+} < d_{13}^{+}$ can be removed because it is implied by the other two conditions in Case 1, Table 4.3. Therefore:

$$\begin{aligned} X_{1,1} &= (d_{21}^{+} \leq d_{12}^{+}) \ \cap \ (d_{32}^{+} < d_{23}^{+}) \ \cap \ (d_{1c} \leq d_{c1}) \\ &= (d_{22}^{-} \leq d_{11}^{-}) \ \cap \ (d_{33}^{-} < d_{22}^{-}) \ \cap \ (d_{11}^{-} \leq 0) \end{aligned}$$

By substitution, we have

$P\{\varepsilon_{1,1}^{mp} = x \ , \ X_{1,1}\}$

$$\begin{aligned} &= P\{\frac{d_{22}^{-}}{2} - d_{11}^{-} = x \ , \ d_{22}^{-} \leq d_{11}^{-} \ , \ d_{33}^{-} < d_{22}^{-} \ , \ d_{11}^{-} \leq 0\} \\ &= \sum_{i=-\infty}^{0} P\{d_{22}^{-} = 2(i+x) \ , \ d_{22}^{-} \leq i \ , \ d_{33}^{-} < d_{22}^{-} \ , \ d_{11}^{-} = i\} \\ &= \sum_{i=-\infty}^{0} \left(P\{d_{11}^{-} = i\} \cdot P\{d_{22}^{-} = 2(i+x) \ , \ d_{22}^{-} \leq i \ , \ d_{33}^{-} < d_{22}^{-}\}\right) \\ &= \sum_{i=-\infty}^{0} \left(g_i(d_{1c}, d_{c1}) \cdot \sum_{j=-\infty}^{i} P\{j = 2i + 2x \ , \ d_{22}^{-} = j \ , \ d_{33}^{-} < j\}\right) \\ &= \sum_{i=-\infty}^{0} \left(g_i(d_{1c}, d_{c1}) \cdot \sum_{j=-\infty}^{i} \left(P\{j = 2i + 2x\} \cdot P\{d_{22}^{-} = j\} \cdot P\{d_{33}^{-} < j\}\right)\right) \\ &= \sum_{i=-\infty}^{0} \left(g_i(d_{1c}, d_{c1}) \cdot \sum_{j=-\infty}^{i} \left(P\{j = 2i + 2x\} \cdot g_j(d_{2c}, d_{c2}) \cdot \sum_{k=-\infty}^{j-1} P\{d_{33}^{-} = k\}\right)\right) \\ &= \sum_{i=-\infty}^{0} \left(g_i(d_{1c}, d_{c1}) \cdot \sum_{j=-\infty}^{i} \left(P\{j = 2i + 2x\} \cdot g_j(d_{2c}, d_{c2}) \cdot \sum_{k=-\infty}^{j-1} g_k(d_{3c}, d_{c3})\right)\right) \end{aligned}$$

where the function $g_x(Y, Z)$ is the probability $P\{Y - Z = x\}$ for two independent

Table 4.4: Means of the Poisson distributions of the delays for each edge in Fig. 4.1, and their corresponding chart in Fig. 4.2.

Scenario		Mean (ms)					
		d_{1c}	d_{c1}	d_{2c}	d_{c2}	d_{3c}	d_{c3}
Fig. 2	(a)	30	30	30	30	30	30
	(b)	30	7	8	25	5	5
	(c)	2	20	5	50	7	80
	(d)	35	5	45	70	2	15
	(e)	10	10	30	12	30	60
	(f)	10	10	30	3	20	5

discrete random variables Y and Z. It is calculated as follows:

$$g_x(Y,Z) = P\{Y - Z = x\} = P\{Y = x + Z\}$$
$$= \sum_{i=-\infty}^{\infty} P\{Z = i\} \cdot P\{Y = x + i\} = \sum_{i=\infty}^{\infty} f_i(Z) \cdot f_{x+i}(Y)$$

This concludes an example expansion to one of the terms in (4.7). Analogous expansion could be made for the remaining terms, which we omit for conciseness.

4.6 Examples of Accuracy Comparison

It has been established that Internet delays follow a Gamma distribution with varying parametrization [21, 108]. We model the OWDs of the six edges of Fig. 4.1 as independent and discrete random variables that follow Poisson distributions,[4] and take on integer values (e.g., delays in milliseconds). Poisson is used because it is a discrete distribution that is a special case of Gamma. Table 4.4 lists the distribution means in six example scenarios. The scenarios were chosen to analyze the effect of delay asymmetry between the client and the verifiers. Figure 4.2 plots the Cumulative Distribution Functions (CDFs) of the absolute errors for each scenario in Table 4.4, using (4.2) and (4.6) for the *av* and the *mp* protocols respectively.

Scenario (a) (Table 4.4) addresses delay symmetry in all six edges.[5] Figure 4.2(a) shows that *mp* is more accurate than *av* in this scenario, with a 54% chance of producing an absolute error <1.5 ms, versus 35% for *av*.

[4]Note that this is not the packet arrival times.

[5]Note that the numbers in Table 4.4 do not represent the delays on each edge. The delays are rather modeled as a random variable following Poisson distributions with the means listed in the table.

Scenario (b) addresses the effect of delay symmetry between the client and one verifier. In this scenario, we deduce that *mp* will operate in Case 2 most of the time (from the *"Order"* column in Table 4.3), and thus $\varepsilon^{mp} = \varepsilon^{mp}_{2,2}$ as it is highly probable that $d_{1c} > d_{c1}$. Because d_{3c} and d_{c3} have equal means (5 ms), the error $\varepsilon^{mp}_{2,2} = |d^{-}_{33}/2|$ becomes relatively small, as shown in Fig. 4.2(b). The *mp* protocol has a 90% chance of resulting in <2.5 ms absolute error, versus 0.1% for the *av*, making it significantly more accurate in this scenario.

Scenarios (c) and (d) explore delay asymmetry in all six edges. Despite the huge asymmetries in (c), *mp* has a ∼25% chance to result in <2.5 ms absolute error, versus ∼0.2% for *av*. The smaller delay variations of scenario (d), compared to (c), caused *mp* to be substantially more accurate (Fig. 4.2(d)).

Scenarios (e) and (f) analyze the effect of delay symmetry between d_{1c} and d_{c1}, and asymmetry in the other two links. In Fig. 4.2(e), where the two graph lines coincide, the accuracy of *mp* is similar to that of *av* because, with higher probability, *mp* operates in Case 3 of Table 4.3 (the resulting OWD-estimates are similar to *av*). In (f), delay asymmetry between the client and $\{v_2, v_3\}$ mislead *mp*, but do not affect the average of d_{1c} and d_{c1}. Because d_{1c} and d_{c1} are highly symmetric (see Table 4.4), *av* is more accurate.

4.7 Related Work

Most research in the area of accurate OWD estimation is primarily to achieve accurate clock synchronization [136], e.g., by predicting delay jitters [77]. Estimation errors and the accuracy of clock synchronization are two metrics generally used to evaluate an OWD-estimation technique. Commonly, there is a tradeoff between the two metrics. The OWAMP tool [134] is a popular example that relies on clock synchronization to accurately estimate OWDs. In the lack of synchronized clocks, the typical method is to measure the RTT, and use its half as an estimate to the OWD [159].

Other methods leveraged the accuracy of GPS clocks to enhance OWD estimation [110]. Additionally, since network queuing delays constitute the most unpredictable delay component, researchers have worked towards devising techniques that enable a sender and a receiver of a Voice over IP (VoIP) application estimate one-way queuing delays without requiring perfect clock synchronization [111]. Despite addressing imperfect clock synchronization, all these proposals assume honest coop-

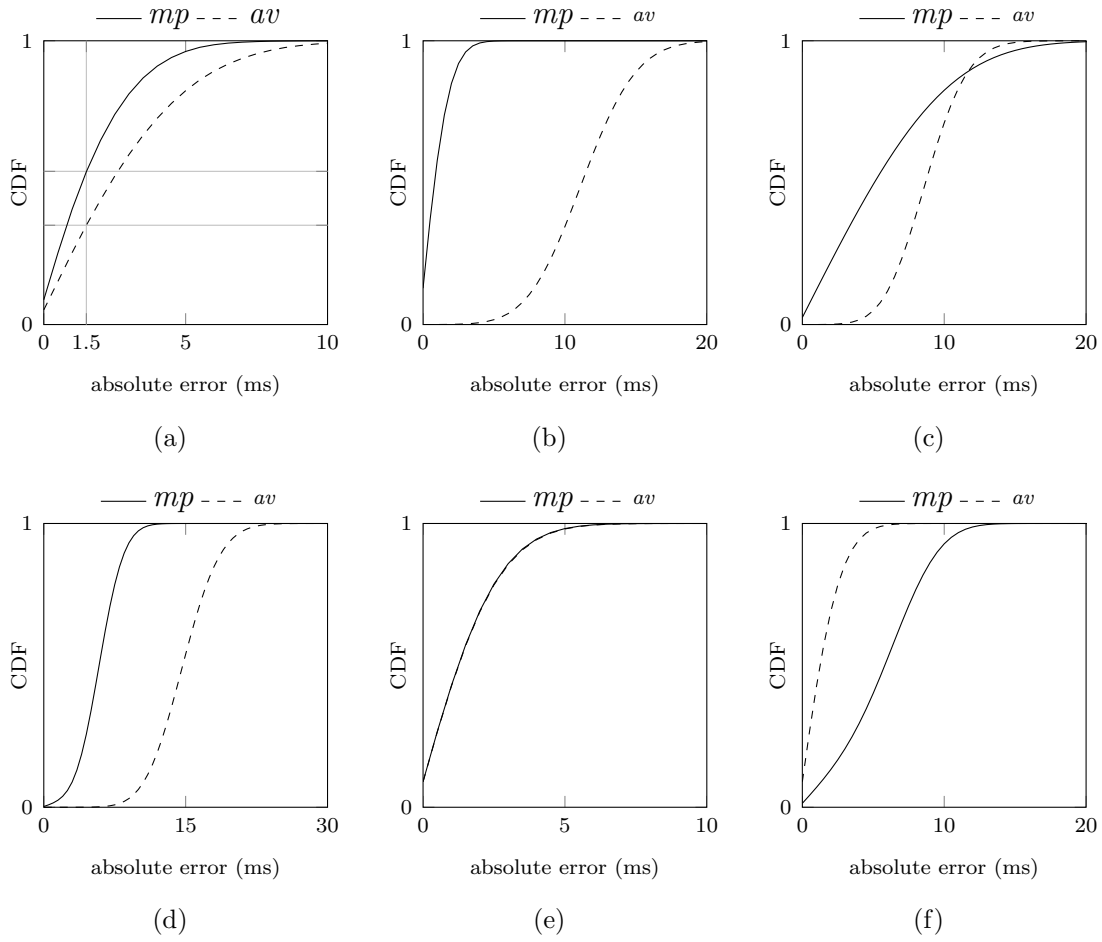


Figure 4.2: Absolute errors between the estimated and the actual OWD, assuming Poisson delay distributions (see Table 4.4 for means) for the edges in Fig. 4.1.

eration between both parties, and thus cannot be used in hostile environments.

4.8 Conclusion

This chapter proposed a novel OWD-estimation protocol that combines accuracy and reduced-cooperation advantages over current state-of-the art techniques that provide one advantage but not the other. The protocol was formally analyzed by deriving the probability distribution of its absolute error, and comparing it with that of *av*. The comparison establishes that the *mp* protocol is in many cases more accurate in estimating OWDs than the commonly-used *av* protocol. This is achieved with the added bonus of the *mp*'s reduced client-cooperation requirements, making it suitable for adversarial environments, but comes at the cost of requiring extra infrastructure (the verifiers).

The probability distribution models derived herein for the *mp* protocol did not consider errors due to imperfect clock synchronization among the verifiers because such errors can be mitigated as shown in the literature [145].

We highlight that the degree of delay asymmetry between the verifiers and the client is a key element affecting the accuracy of both protocols. The PMFs derived herein are thus useful to an application deciding between both protocols. This follows from the properties of the PMFs derived herein: (1) they allow determination of which protocol is more accurate in estimating OWDs given the delay environment, and (2) they are generic—they evaluate the probability mass of error given any discrete delay distribution (Poisson was used herein). Note however that, despite being generic, the PMFs derived herein must be used with discrete delay distributions. We did not pursue the avenue of PDFs that can be used with continuous delay distributions models, and therefore cannot advise on whether any technical difficulties would be encountered. However, it would appear that analogous steps would provide a corresponding analysis for the case of continuous distributions. This is left as future work.

Chapter 5

CPV: Delay-based Location Verification for the Internet

The number of location-aware services over the Internet continues growing. Some of these require the client's geographic location for security-sensitive applications. Examples include location-aware authentication [17, 73], location-aware access policies, fraud prevention, complying with media licensing [51], and regulating online gambling/voting. An adversary can evade existing geolocation techniques, e.g., by faking GPS coordinates or employing a non-local IP address through proxy and virtual private networks. This chapter presents CPV, a delay-based technique designed to verify an assertion about a device's presence inside a prescribed geographic region. CPV does not identify devices by their IP addresses. Rather, the device's location is corroborated in a novel way by leveraging geometric properties of triangles, which prevents an adversary from manipulating measured delays. To achieve high accuracy, CPV mitigates Internet path asymmetry using the OWD-estimation protocol introduced in Chapter 4, and leverages delay-related information for evidence supporting/refuting the asserted location. We explain the threat model, detail the CPV algorithm, and discuss its security benefits.

5.1 Introduction

Over the Internet, LSPs are those that customize their content/services based on

The content of this chapter is accepted for publication at IEEE TDSC [9].

the geographic locations of their *clients* (the software that communicates with the LSP, typically a web-browser). Some LSPs restrict their services to certain geographic regions, such as media streaming [26] (e.g., hulu.com); others limit certain operations to a specific location, such as online voting (e.g., placespeak.com), online gambling (e.g., ballytech.com), location-based social networking [122] (e.g., foursquare.com), or fraud prevention (e.g., optimalpayments.com). LSPs may also use location information as an additional authentication factor to thwart impersonation and password-guessing attacks (e.g., facebook.com). Privacy laws differ by jurisdiction, which allows/bans content based on region [143]. The nature of the provided services may motivate clients to forge their location to gain unauthorized access.

Existing geolocation technologies, commonly used in practice, are susceptible to evasion [107], as discussed in Section 2.2 (page 2.2). Tabulation-based techniques, where a geolocation service provider maintains tables that map IP addresses to locations—e.g., MaxMind [103], can be evaded through IP address-masking technologies [30] such as proxy servers and anonymizers [41]. Geolocation that is based on active delay measurements [13,91] is prone to an adversary corrupting the delay-measuring process [59]. A location verification technique is therefore required to provide greater assurance of the veracity of the specified location.

Various solutions have been proposed to verify location claims in wireless networks [28,133]. However, solutions in this domain cannot be directly adopted by multi-hop networks, e.g., the Internet, due to delay characteristics of different domains. For example, Internet delays are stochastic [44], whereas in single-hop wireless networks, delays can be estimated from the distance the signal spans and the speed of its propagation.

Verifying the location of Internet clients is a challenging problem [107]. A practical approach must address critical challenges such as handling of IP address-masking, and ensuring the correctness of location information submitted by the client. We present and evaluate CPV, a delay-based technique designed to verify a client's geographic location. Experimental results show that CPV provides a high level of assurance that a correct (i.e., honest) location assertion is verified to a granularity equivalent to a circle of radius $\sim 400km$. CPV is designed to resist known geolocation-circumvention tactics as it (1) does not rely on the client's IP address, (2) does not rely on client-submitted information, and (3) is designed such that manipulating the delays is not in the dishonest client's favor, e.g., CPV precludes the attacks of Chapter 3, as well as those of Gill *et al.* [59].

A common challenge faced by delay-based geolocation techniques is to find an accurate delay-to-distance mapping function, and thus factors affecting the correctness of this mapping have been well studied in the literature [92, 167]. CPV undertakes a set of measures to mitigate the effect of these factors. For example, it mitigates path asymmetry [116] by relying on OWD-estimates, instead of RTTs, to/from a potentially dishonest client, using the *minimum pairs* protocol introduced in Chapter 4. Additionally, CPV mitigates network instability [35] by iterating the OWD-estimation process.

In Chapter 6, the effect of several factors on the correctness of CPV is analyzed by evaluating its False Reject (FR) and False Accept (FA) rates using PlanetLab [33], where all modeled clients are assumed to use wired access networks. Further in Chapter 7, the correctness of CPV is analyzed when only legitimate clients are using wireless access networks.

The rest of this chapter is organized as follows. Section 5.2 provides a summary of the literature on delay behavior over the Internet, and its relationship to geographic distances. The threat model is discussed in Section 5.3, and CPV is explained in Section 5.4. A security discussion is presented in Section 5.5. Section 5.6 concludes.

5.2 Background

Delay characterization between Internet hosts plays a prominent role in numerous applications such as distributed web-caching, server placement in Content Distribution Networks (CDNs), clock synchronization, overlay Peer to Peer (P2P) networks, Internet geolocation, application-layer mutlicast, and timeout estimations in TCP. Due to the importance of understanding the impacts of delays between Internet hosts on delay-dependent applications, factors affecting these delays have been well studied [92, 147, 153, 167] including the spanned geographic distances, routing policies, etc.

Delay-based IP geolocation includes a broad class of techniques aiming to calculate the geographic location of a client based on the delays observed between the client and a set of landmarks with known locations [67]. Most techniques apply regression analysis to find a function that best models the relationship between the measured delays and geographic distances [44,91]. Multilateration is then used on the distances mapped between the landmarks and the client to constrain the region where the client is located. Recent techniques incur a median error of as low as a few kilometres

[91]. To infer distances from delays, the speed at which packets are transmitted over the Internet has been approximated by Katz-Bassett *et al.* [85] to 4/9 the speed of light in vacuum, a ratio called the Speed of the Internet (SOI) [85]. However, the actual speed is affected by several factors such as time of the day, region and characteristics of the underlying network. Based on 19 million RTT measurements in the Internet, Landa *et al.* [92] found that the knowledge of the geographic distance between two nodes, their /8 IP prefixes, and their countries can help scope down delay-estimation errors to within $\sim 22ms$.

Network Coordination Systems (NCSs) [36] model a network as a geometric space by assigning coordinates to each node in the network. The coordinates denote a node's position relative to other nodes in the *network delay space*, i.e., according to its delay to/from them. One essential advantage of NCSs is the ability to locate a node's network position relative to *almost all* other nodes without overwhelming the network with storms of delay sampling [45]. NCSs are vulnerable to an adversary falsifying its coordinates [60].

The aforementioned delay studies provide solid evidence of a strong correlation between Internet delays and geographic distances [155], which is commonly speculated to stem from improved global network connectivity [67]. CPV leverages these results to address location verification.

5.3 Threat Model

We now explain the threat model addressed by CPV. Note that this threat model is different from the adversarial models explained in Section 3.4 (page 28); those in Chapter 3 explain how various adversarial capabilities can manipulate delay-based geolocation.

The adversary is a human user that programs its client software to evade a geolocation process, to intentionally misrepresent its location. The adversary is in physical possession of the client device (e.g., laptop or smartphone), which is connected to the Internet and thereby to the LSP. The adversary has full control over its client device; it can install/uninstall any software.

We consider within scope an adversary that uses public proxies, VPNs and/or anonymizers to hide its IP address or to hide any other identifying information that may reveal its true location. The adversary is also capable of manipulating delays, as explained in Chapter 3.

CPV is designed to verify the output of a geolocation technique. The adversary must thus be able to mislead that technique first to forge its location. We assume, for simplicity, that the geolocation step prior to the operation of CPV is an unverified location assertion; CPV is then to verify this assertion. By considering this case, whereby the adversary can simply assert a location (e.g., the LSP asks its users to simply input their location), the adversary is powerful enough to evade any basic geolocation technique.[1]

We define the *target location* as the location the adversary attempts to appear at. The following two use cases explain adversarial motivation to forge location, both of which are within the threat model.

Impersonation. To mitigate online impersonation of users' accounts, typically done through password-guessing attacks, logins can be restricted to location(s) (e.g., country) associated with the legitimate user's account. To impersonate a user, the adversary needs to not only guess the user's password, but also the user's associated location, and place itself fraudulently in that location. In this case, the adversary's target location changes widely according to the account being attacked.

Violation of geographic-restriction policies. When an LSP customizes its services/content based on the location of its users, such as location-sensitive multimedia providers (e.g., Pandora [114] and Hulu [78]), adversaries may be motivated to evade geolocation to gain location-dependent benefits. This threat is harder to defend against than the previous one, since the adversary's target location is fixed (i.e., the adversary does not have to keep modifying its geolocation evasion mechanism to appear at different parts of the world), and immediately known to the adversary.

5.4 CPV: Client Presence Verification

CPV builds on the established result that Internet delays and geographic distances have strong positive correlation [139] (see Section 5.2). In CPV, when a client asserts its presence in a geographic location, delays are measured between the client and three *verifiers*[2] encompassing the asserted location. These delays are then processed to provide assurance that the client is truly present (geographically) inside the tri-

[1] Some geolocation techniques are harder to evade than others. See Chapter 3.

[2] In practice, verifiers could be dedicated servers maintained by an independent party providing location verification as a service.

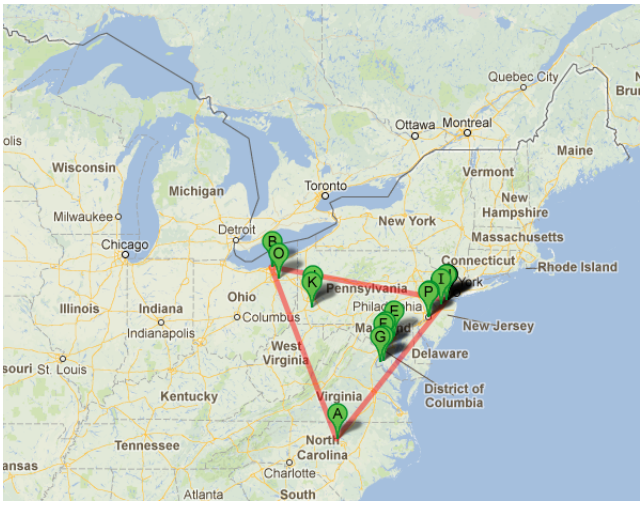


Figure 5.1: An example of 13 clients inside a triangle projected by verifiers in Duke University, Case Western Reserve University and Rutgers.

angle determined by the three verifiers. The size of that triangle is the verification granularity. Figure 5.1 shows an example triangle and several inside clients.

To reduce falsely rejecting legitimate (honest) clients and falsely accepting adversaries, factors affecting the delay-distance correlation (e.g., route circuitousness, queuing delays and congestion) must be addressed. The forward and reverse paths between any two hosts over the Internet are often affected by those factors differently, resulting in delay asymmetry [116]. The less affected path is likely to be the faster one (i.e., with a smaller OWD), and thus better represents the distance between the two hosts. Relying on the smaller OWD between the client and the verifiers rather than the RTTs is, thus, expected to improve CPV's accuracy in judging location assertions.

CPV uses the *minimum pairs* protocol for OWD-estimation (see Chapter 4). Accurate OWD-estimation is one measure utilized by CPV for accurate delay-to-distance mapping. By the end of this section, a summary is provided on how CPV manages the delay-measurement process to reduce the factors affecting this mapping, without jeopardizing the integrity of the location verification process.

After mitigating these factors, CPV uses a simple function to map delays to distances, and verifies assertions based on these distances (see Section 5.4.3 below).

5.4.1 Operational Requirements

CPV requires geographically-distributed verifiers whose locations are consistent with the LSP's Permitted Geographic Regions (PGRs). PGRs are the regions in which clients are permitted to receive services/content or carry out location-specific operation (e.g., login or vote). The client must not control any of the verifiers involved in corroborating its assertion. To successfully enforce the LSP's location-aware policies, the verifiers must:

1. be publicly reachable over the Internet; and
2. the convex hull of the verifiers must encapsulate the LSP's PGRs.

5.4.2 Notation and definitions

The set of verifiers available to the LSP is denoted $\mathbb{V}$. For any triangle, $\triangle$, the set of the three verifiers determining $\triangle$ is denoted $V_{\triangle} \subset \mathbb{V}$. For any geographic location $l = \{\text{latitude, longitude}\}$, E_l is the set of triangles enclosing l, such that all $\triangle_l \in E_l$ are near equilateral in the network delay-space (see Section 5.2), and do not cross the PGR border.

A client and three verifiers make four triangles. The function $valid(\mathbf{D})$ checks for Triangular Inequality Violations (TIVs) in the four triangles whose side lengths are mapped from the six OWDs in $\mathbf{D}$. It returns true only if, for each of the four triangles, the sum of each two sides is greater than the third. The function $area\_v(\mathbf{D})$ calculates the area of the triangle determined by the three verifiers; the side lengths of that triangle are mapped from the three OWDs in $\mathbf{D}$ that belong to the edges between the verifiers. The function $area\_c(\mathbf{D})$ similarly calculates the areas of the three triangles determined by each pair of verifiers and the client, and returns the summation of those areas.

5.4.3 CPV description

CPV's verification process begins with an asserted client location as input, $l = \{lat, lon\}$. The LSP chooses a triangle $\triangle_l \in E_l$, and informs the client of the IP addresses of the verifiers in $V_{\triangle_l}$. The client connects to the verifiers and the verification process, Algorithm 2, begins.

First (in line 4), the verifiers estimate the *smaller* of the forward and reverse OWDs

Algorithm 2: Executed by the verifiers in $V_{\triangle_l}$ when a client asserting to be at location l connects to them. See inline for the definition of the function *acceptable*(); similarly, see Section 5.4.2 for the definitions of the functions *valid*(), *area_c*() and *area_v*().

Input: Number of iterations, $n_{\triangle_l}$; tolerance of area inequality, $\epsilon_{\triangle_l}$; and acceptance threshold $\tau_{\triangle_l}$.
Output: Accept/Reject client's location assertion
begin

```
1  pass := 0
2  for i := 1 to n_{△_l} do
3      D_i := φ
4      Estimate, in real time, the one-way delays for D^mp and D^av using
           Algorithm 1 (see Chapter 4).
5      if valid(D^mp) then D_i := D^mp
6      else if valid(D^av) then D_i := D^av
7      if D_i ≠ φ then
8          δ_i := area_c(D_i) − area_v(D_i)
9          if δ_i ≤ ε_{△_l} and acceptable(D_i) then
10             pass := pass +1
11 Γ := pass/n_{△_l}
12 if Γ < τ_{△_l} then
13     Reject client's location assertion
14 else
15     Accept client's location assertion
```

at the six edges between the verifiers and the client using two protocols: *minimum pairs* (*mp*) and *average* (*av*), as explained in Chapter 4. The six OWDs are then mapped to distances according to the simple mapping function $f(x) = x$, i.e., x *ms* is equal to x *km*. The resulting distances are never used in an absolute form; they are only processed relative to each other. This design provides the advantage of resilience to factors that affect the network comprising the client and the three verifiers, e.g., a network congestion that affects the delays of the six edges altogether.

OWD estimation is done iteratively (line 2), where the input parameter $n_{\triangle_l}$ specifies the number of iterations to be performed, to account for possible delay instability [162]. The *confidence ratio*, Γ (line 11), represents the verifiers' confidence of the truthfulness of the asserted location. It is calculated as the proportion of iterations where the values of $area\_c(\mathbf{D}^{mp})$ and $area\_v(\mathbf{D}^{mp})$ (see Section 5.4.2 for notation) match within a suitable error tolerance, ϵ. From a geometric perspective, we have the following claim (see Appendix B for proofs):

Claim 1 *Let P be a point in the Cartesian plane, and let $\triangle XYZ$ be the triangle determined by the points X, Y and Z. If P is strictly outside $\triangle XYZ$, then the sum of the areas of $\triangle XYP$, $\triangle XPZ$ and $\triangle PYZ$ is greater than the area of $\triangle XYZ$.*

TIVs are evident in the Internet [99]. Because CPV relies on triangular areas in verifying location assertions, TIVs can thwart CPV's successful operation. Additionally, an adversary can increase the estimated OWDs of the *mp* protocol, flattening some triangles and resulting in TIVs. Thus, the verifiers become less confident about the truthfulness of the asserted location as more TIVs occur, which is a security precaution to reduce potential false accepts. This can be seen in line 9, where $\mathbf{D}_i$ must hold a valid set of delays (from lines 5 or 6) for Γ (line 11) to increase.

Iterating the delay-estimation process helps reduce the number of benign TIVs [147], hence reducing the number of FRs. Additionally, more than one delay-estimation protocol (namely, both *mp* and *av*) further lessens the effect of TIVs; *av* is used as a fallback if the estimates in $\mathbf{D}^{mp}$ result in TIVs [147]. In lines 5 and 6, $\mathbf{D}^{mp}$ is checked first because it is more resilient to delay spikes, as discussed in Chapter 4.

On the other hand, such iterative delay-estimation approach may affect the usability of CPV, as it increases the time required by CPV to reach a decision. Some applications may require a decision before providing the location-sensitive service to users, such as online credit card transactions. However, in other applications, the verification algorithm may run in the background (i.e., continuously and concurrent to the location-sensitive application), such as media streaming. As such,

despite its potential usability drawbacks, the impact of the number of iterations on the usability of CPV depends essentially on the application.

The error tolerance, ϵ (line 9), accounts for route circuitousness [153], congested routes, or other factors that contribute to inaccuracies in the delay-distance mapping over the Internet. If an adversary's true location is so far from the asserted location that one of the *inner* triangles (those having the client as one of their vertices) becomes obtuse, the triangle becomes flattened and its area decreases. An unnecessarily large error tolerance may thus falsely accept this adversary.

To mitigate this effect, we include the *acceptable*($\mathbf{D}$) function (line 9), which checks that the OWD between verifier v and the client is not larger than the OWDs between v and the other two verifiers. The function returns true only if the previous statement is true for the three delay-mapped distances in $\mathbf{D}$ that are between the client and the verifiers. From a geometric perspective, using the notation $\overline{AB}$ for the length of line segment AB, we have the following claim (see Appendix B for proofs):

Claim 2 *Let W be a point in the Cartesian plane, and let $\triangle XYZ$ be the triangle determined by the points X, Y and Z such that $\overline{XZ} \leq \overline{XY}$. If $\overline{XW} > \overline{XY}$, then W is strictly outside of $\triangle XYZ$.*

Calibration of input parameters. Calibration of input parameters. To set the three input parameters of Algorithm 2 for each $\triangle$, the three verifiers in $V_{\triangle}$ can operate CPV to verify the geographic presence/absence of network nodes that are known (as a ground-truth) to be inside/outside $\triangle$ (e.g, using other verifiers in V). Based on the delays between the verifiers and these nodes, the input parameters should be set such that CPV accepts inside nodes, and rejects outside ones. For example, in line 11 (Algorithm 2), if $\Gamma \geq 0.6$ for all such nodes, then $\tau_{\triangle_l}$ should be set to 0.6.

Summary. CPV's measures to reduce factors negatively affecting delay-to-distance mapping can be summarized as follows:

1. Two protocols are used to estimate OWDs instead of one to reduce the effect of TIVs.
2. Active delay measurement is used with each client, which reflects the most recent delay status in the region [162].
3. No universal delay-to-distance mapping is used. Rather, mapping is done relative to other delays in the region.

4. Delay-estimation is conducted iteratively to more accurately converge to the actual delays at current network conditions [74].
5. The three verifiers are chosen within a geographical proximity of the asserted location to
   (a) reflect regional delays [44, 167];
   (b) span fewer Autonomous Systems, which reduces route circuitousness [139];
   (c) reduce the number of TIVs [147]; and
   (d) exhibit stronger positive correlation between delays and distances [92].

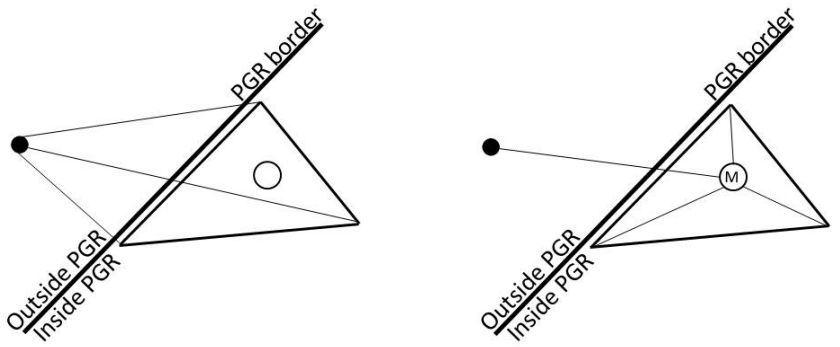
5.5 Security Discussion

5.5.1 Classical Geolocation Attacks

Submitting false information. Although this may mislead simple geolocation techniques [107], it does not defeat CPV because the verification process (Algorithm 2) is independent of any information submitted by the client. Chapters 6 and 7 analyze CPV's efficacy in detecting false location assertions (Fig. 5.2(a)) due to area mismatch or large client-verifier delays.

Using middleboxes. Some IP geolocation techniques can be circumvented if a client's IP address is concealed using generic MBs such as proxies, anonymizers, or VPNs [107]. These do not threaten the integrity of the verification process of CPV because delay measurements are conducted over the client's application layer. MBs that blindly relay application-layer traffic (Fig. 5.2(b)) will also relay the timestamps (see Section 5.4) to the client [30]. Chapter 8 shows how a MB specifically designed to defeat CPV by searching application-layer traffic for timestamps could be mitigated using a PoW mechanism.

Manipulating delays to increase calculated distances. Delay-adding attacks [59] can be attempted on CPV when the adversary inserts a delay before forwarding timestamps. Assuming verifier i sent a timestamp, the adversary failing to forward it promptly to verifier j enlarges d_{ic} and d_{cj} fraudulently, increasing the value of $d_{ic}+d_{cj}$ (see Fig. 4.1 in Chapter 4 for notation). Because the *mp* protocol estimates the smaller OWD at each edge by solving simultaneous equations, selectively delaying timestamps can result in delay estimates that are smaller than the actual delay. For example, solving simultaneously the equations $a + b = 7$, $a + c = 8$ and $b + c = 9$



(a) (b)

Figure 5.2: An adversary asserting a false location (a) without using a middlebox, and (b) using a middlebox at the asserted location. •=true location; ○=asserted location; M=middlebox; PGR=Permitted Geographic Region.

gives $a = 3$, $b = 4$, and $c = 5$. Whereas $a + b = 7$, $a + c = 8$, and $b + c = 13$ results in $a = 1$, $b = 6$ and $c = 7$. Thus, *increasing* $b + c$ resulted in a *smaller* value for a.

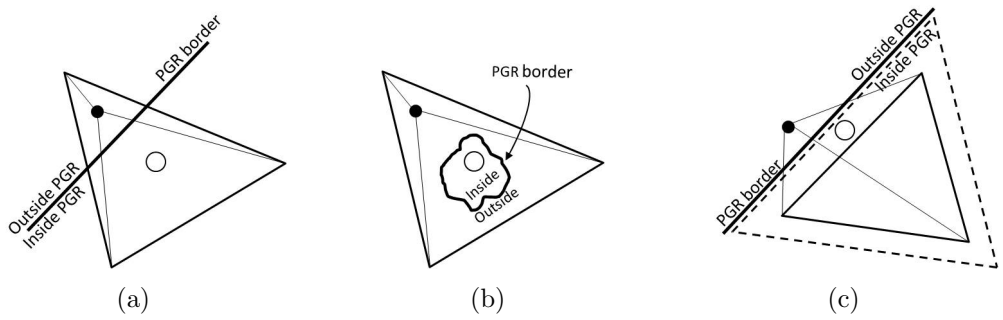
However, the adversary cannot reduce the *summation* of d_{ic} and d_{cj} as this requires speeding up the traffic propagation between the adversary and the verifiers [59]. From a geometric perspective, increasing the summation of any pair of edges does not help an adversary outside a triangle to forge its location making it inside. Formally, using the notation $\overline{AB}$ for the length of line segment AB, we have the following claim (see Appendix B for proofs):

Claim 3 *Let P be a point in the Cartesian plane, and let $\triangle XYZ$ be the triangle determined by the points X, Y and Z. If P is strictly outside $\triangle XYZ$, then increasing the sums $\overline{XP} + \overline{PZ}$, $\overline{XP} + \overline{PY}$ or $\overline{YP} + \overline{PZ}$ without reducing at least one of the other sums cannot place P inside $\triangle XYZ$.*

Manipulating delays to cause TIVs. As shown in Algorithm 2, CPV holds the number of TIVs against the client (the condition $\mathbf{D}_i \neq \phi$ in line 9 means $\mathbf{D}_i$ must not violate the triangle inequality to increment *pass*). In conclusion, manipulating delays does not help the adversary, but rather signals the adversary's evasion attempts.

5.5.2 Attempts to Evade CPV

To study potential vulnerabilities in CPV, we review steps where the verifiers interact with the client.



(a) (b) (c)

Figure 5.3: (a) and (b) inappropriately deployed verifiers; (c) insufficiently deployed verifiers. •=true location; ○=asserted location; PGR=Permitted Geographic Region.

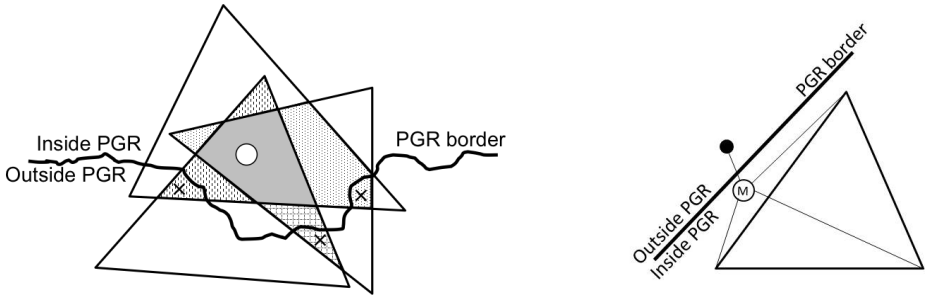
Connecting to the verifiers. Assuming the adversary's target location (location it is trying to appear at) is l, connecting to a set of verifiers $V_{\triangle_{l'}} \neq V_{\triangle_l}$ does not help the adversary in pretending to be at l as those verifiers cannot verify the adversary's presence inside $\triangle_l$.

Forwarding the timestamp. Because the verifiers sign the timestamps, the adversary can neither forge nor inject fake ones. Delaying a timestamp is discussed in Section 5.5.1.

5.5.3 Poor Verifier Deployment and PGR Proximity

Adversaries bordering the PGR may be able to exploit inappropriate or insufficient verifier deployment. Figures 5.3(a) and 5.3(b) show examples of inappropriately deployed verifiers with respect to the PGR, where a triangle crosses the PGR border or encloses the PGR inside itself. As shown, a close adversary could be outside the PGR but inside those triangles. Verifying the presence inside the triangle does not ensure presence inside the PGR in those cases. Figure 5.3(c) shows potential vulnerability due to insufficient verifiers/triangles: not all regions inside the PGR are covered with triangles. The verifiers determining the shown (solid) triangle should not overly relax $\epsilon_\triangle$ to account for the uncovered region (relaxing $\epsilon_\triangle$ is depicted by the *dashed* triangle in Fig. 5.3(c)). Otherwise, the verifiers falsely accept an adversary close to the PGR asserting to be at the uncovered region of the PGR, as shown in Fig. 5.3(c).

Possible countermeasures. To address PGR border crossing, additional overlapping triangles could be used to enclose the asserted location as long as a single triangle, or the intersection of multiple triangles, crosses the PGR border. The in-



(a) ×=possible true locations. (b) M=middlebox;

Figure 5.4: Defenses against a bordering adversary that exploits inappropriate or insufficient verifier deployment. •=true location; ○=asserted location; PGR=Permitted Geographic Region.

tersection region of the triangles must (1) not cross the PGR border and (2) enclose the asserted location, as shown in Fig. 5.4(a). Client presence inside the PGR is then verified only if the verifiers of each triangle accept the assertion. For example, in Fig. 5.4(a), if the client's (adversary's) true location was at any of the areas marked with ×, two triangles may falsely accept the assertion. Two triangles are insufficient in that case because the PGR border crosses the overlapping areas of each two of the three triangles. Verifying the presence inside all three suffices to verify the correctness of the assertion.

As for insufficient deployment of verifiers, whenever an assertion is made in a region not covered by any triangle, the LSP (location-sensitive provider) could use a measurement-based IP geolocation technique instead of relying on client-dependent geolocation (such as GPS). A bordering adversary must then evade this technique prior to bypassing CPV. It would then be challenging for the adversary to precisely target a location not covered by any triangle only through delay manipulation [59]. In such a case, using a measurement-based IP geolocation technique motivates the adversary to use a MB inside the uncovered region of the PGR (Fig. 5.4(b)). However, MBs tend to increase delays [30], which helps the verifiers detect the adversary's false assertion.

5.6 Conclusion

CPV is a delay-based technique which, to the best of our knowledge, is the first to verify a client's location over the Internet without assuming the client's possession of a secret personal identifier (see Section 2.3). CPV mitigates delay spikes injected

by the Internet as it iterates the delay-measuring process, and corroborates the client's location based on the *smaller* OWD, as estimated using the *minimum pairs* protocol (Chapter 4). In CPV, delays are estimated between a client and three verifiers, which enclose the client's unverified location within their convex hull. The verifiers estimate the delays over the client's application layer to overcome IP hiding tactics, typically carried out using Middle Boxes (MBs). For clients using web-browsers, CPV requires no extra client-side software; the client's browsing experience is retained as the verification process could run in the browser. These advantages highlight CPV's potential for practical adoption.

In the following chapter, CPV is evaluated using detailed experiments in a real-world environment when legitimate clients are using wired access networks. Further, in Chapter 7, experimental logs collected from the wired testing are modified to represent last mile delays of a client using a wireless access network, and CPV is reevaluated under these conditions.

Chapter 6

Evaluating CPV in Wired Networks

In this chapter, CPV is evaluated in wired networks through detailed experiments on PlanetLab [33], exploring various factors that affect its efficacy, including the granularity of the verified location, and the verification time. The evaluation of CPV in wireless netowrks is presented in Chapter 7.

We use the rates of False Rejects (FRs) and False Accepts (FAs) as the assessment metrics. If a client asserts to be at location l, an FR occurs when this client is actually present somewhere inside $\triangle_l$, and is judged by the verifiers in $V_{\triangle_l}$ as absent from $\triangle_l$. By contrast, an FA occurs when that client is actually absent from $\triangle_l$, and is judged by the verifiers in $V_{\triangle_l}$ as present in $\triangle_l$.

We use 80 PlanetLab [33] nodes in USA and Canada (Fig. 6.1), and identified 34 different sized triangles satisfying the requirements stated in Section 5.4. The triangles were chosen with internal angles ranging 50-70 degrees so as to be near-equilateral in the network delay-space, as specified in Section 5.4.2, page 68. Triangular areas ranged from $\sim$32,000 km^2, almost the size of Maryland state, to $\sim$500,000 km^2, almost the size of Spain.

We assume that the Permitted Geographic Region (PGR) (see Chapter 5) is a triangular-shaped region that perfectly coincides with the dimensions of the triangle. One triangle was considered at a time. For each triangle, all nodes—except the

The content of this chapter was published at the 2014 IEEE CNS conference [10] (with a full length version accepted for publication in IEEE TDSC [9]).

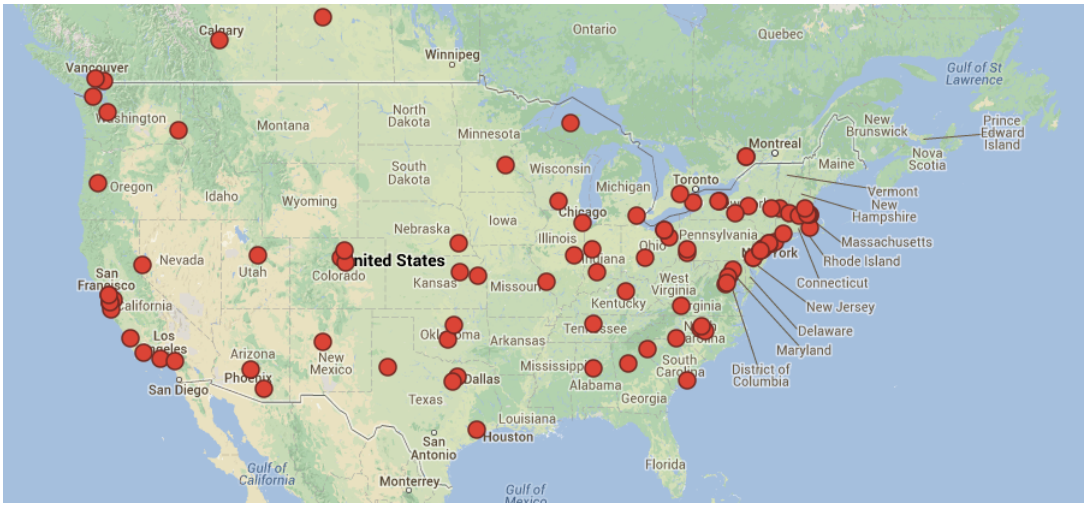
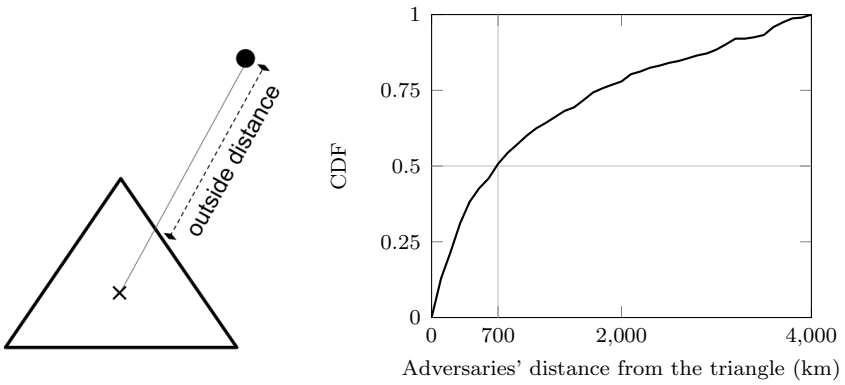


Figure 6.1: Locations of the 80 PlanetLab nodes used in our experiments. Map data: Google, INEGI.

three determining the triangle—acted as clients; all clients had provided assertion to be at the centroid of that triangle. Combining clients of all triangles, *legitimates*[1] (clients actually inside) totalled 146 and *adversaries* (clients actually outside) totalled 2,301 for a total of 2,447 experiments. The verifiers determining each triangle were verifying assertions of all clients concurrently. The verifiers used Network Time Protocol (NTP) [105] to synchronize their clocks. Knowing the ground truth of legitimates and adversaries with respect to each triangle, our objective is to identify the optimal values for the tolerance of the area inequality ($\epsilon_{\triangle}$) and the acceptance threshold ($\tau_{\triangle}$) for each of the 34 triangles, and quantify the FRs and FAs at these values.

To see how far adversaries were from the triangles in the experiments, we define the adversaries' *outside distance* with respect to each triangle in our experiments as the distance between the adversary's true location and the point of intersection between lines A and B; line A is the one passing through the adversary's true location and the triangle's centroid; line B is the triangle's closest side to the adversary (see Fig. 6.2(a)). Figure 6.2(b) shows a CDF of the 2,301 adversaries' outside distance. Half the adversaries were less than 700 km away from the triangle's closest side (i.e., the triangle encapsulating their fraudulently asserted location), and no adversary was farther than 4,000 km away. For reference, the width of the United States is approximately 4,000 km. The argument is that if CPV rejects relatively nearby adversaries, it will reject more distant ones.

[1]We use the word *legitimates* (i.e., as a noun) to refer to legitimate clients.



(a) ×=centroid of triangle; •=adversary's true location.

(b) CDF of outside adversarial distance

Figure 6.2: Adversaries' distances from the triangles' closest side. (a) How the external distance is calculated with respect to a triangle; (b) A point (x, y) means the proportion y of adversaries were x km away from the closest side. Note: this graph shows experimental design, not results.

Implementation details. Both the CPV server and the client were implemented as Java applications. The server's code was run on the three PlanetLab nodes chosen as CPV servers at each experiment; the client code's was run on the remaining nodes representing the CPV clients. Each CPV client was informed with the server's IP addresses and port numbers. When all three CPV servers are started and waiting for clients to connect, all clients were started in parallel and the verification process begins across all clients simultaneously. Note that in practice, the CPV algorithm requires no specific client-side software because the client side can be implemented using javascript and websockets.

Experiments were run over the course of a month (April 2013) and at different times of the day. The number of iterations, $n_\triangle$ (Algorithm 2 on page 69), was fixed at $n_\triangle = 600$ for all $\triangle$ in the 34-triangle set to study the factors affecting CPV over a relatively long period of time (a total of ∼13.3 million delay measurements were taken between all nodes). Fewer iterations might be sufficient to judge a client, as we show in Section 6.6 below.

Limitations of PlanetLab. Despite being generally used as an experimental testbed representing the global Internet, PlanetLab measurements should not absolutely be deemed as so [16]. Many of PlanetLab nodes are connected through the Global Research and Educational Network (GREN), e.g., Internet 2 [80] and CANARIE [27], in which traffic could be fully routed within the network. Accordingly, all experiments conducted in this thesis are subject to PlanetLab's network

settings [131].

The rest of this chapter is organized as follows. Section 6.1 details an example from the experiments, which involves three clients: one legitimate and two adversaries. Sections 6.2, 6.3, and 6.4 respectively analyze the rates of TIVs, examines the use of the triangular areas as CPV's primary assertion-verification metric, and analyzes CPV's confidence ratio associated with all experimented clients. In Section 6.5, the effect of legitimate clients' closeness to the triangles' sides is examined, and in Section 6.6 the appropriate number of iterations is analyzed. Section 6.7 analyzes hypothetical modifications to CPV, where the OWD-estimation process is modified and CPV's efficacy is reevaluated. Finally, Section 6.8 concludes.

6.1 An Example

We detail the results of one of the triangles in our 34-triangle set, and three of the clients being verified by that triangle. One of the clients was legitimate, the other two were adversaries. Figure 6.3(a) shows the geographic location of the triangle and the three clients, labelled D, E and F. The area difference, δ_i (line 8 of Algorithm 2) for all $1 \leq i \leq 600$, is plotted for the three clients in Fig. 6.3(b).

Number of Triangular Inequality Violations (TIVs). Some iterations have no corresponding values for the area difference (visible in high resolution). Those are the ones where $valid(\mathbf{D}^{mp})$ and $valid(\mathbf{D}^{av})$ (lines 5 and 6 of Algorithm 2) returned false, i.e., the mapped distances resulted in at least one TIV of the four triangles determined by the three verifiers and the client. Of all 600 iterations, the number of iterations where both functions returned false for D, E and F are 114, 11 and 0 respectively. The number of TIVs is high for D likely due to its relatively close position to two of the three triangle's sides (versus one side as with E).

Area difference (δ). From Fig. 6.3(b), the median of δ_i, $\tilde{\delta}$, for clients D, E and F is 30 km^2, 66 km^2 and 209 km^2 respectively. The median corresponding to F is substantially larger than that of D and E because F is relatively far away from the triangle. The smallest recorded area difference for F is $\delta_{325} = 102$ km^2. Therefore, any value for $\epsilon_\triangle$ in the range $\epsilon_\triangle < 102$ keeps the variable $pass = 0$ (line 10, Algorithm 2) for all iterations, resulting in $\Gamma = 0$. Consequently, at $\epsilon_\triangle < 102$, any value for $\tau_\triangle$ (the acceptance threshold, Section 5.4) in the range $\tau_\triangle > 0$ rejects F's assertion. Client E was less than 50 km away from the triangle's nearest side AC, thus the average area difference of E is close to that of D. However, at $\epsilon_\triangle = 45$, there is a

Table 6.1: Results for clients D, E, and F. The "Section" column shows the section where each variable (row) is analyzed further for all experiments.

Variable	Client			Section
	D	E	F	
Number of TIVs	114	11	0	6.2
$\tilde{\delta}$ (km^2)	30	66	209	6.3
Γ (0 to 1)	0.84	0.2	0	6.4

visible distinction between both nodes—there existed a value for $\epsilon_\triangle$ (i.e., 45 km^2) that enabled the verifiers to correctly judge the assertions of both clients, D and E, despite being geographically collocated.

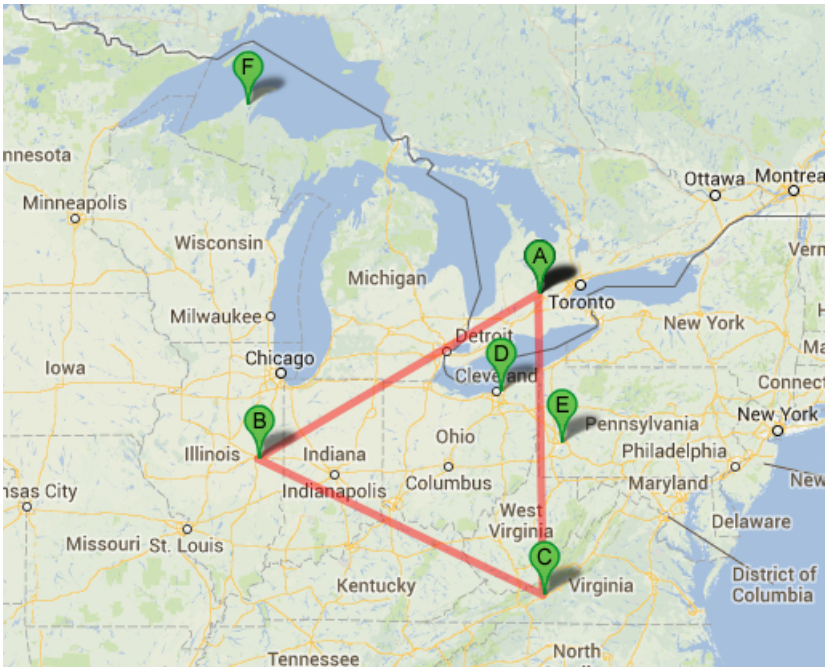
Confidence ratio (Γ). In Algorithm 2, Γ is calculated when all n iterations are performed. Figure 6.3(c) plots Γ (at $\epsilon_\triangle = 45$ km^2), assuming it was calculated at each iteration. Despite the relatively close values of δ_i between D and E in Fig. 6.3(b), their Γ greatly differs. At $i = 100$, Γ is 0.86 and 0.3 for D and E respectively. Therefore, after 100 iterations, any $\tau_\triangle$ in the range $0.3 < \tau_\triangle \leq 0.86$ enables the verifiers to decide that D is a legitimate and E is an adversary. When all 600 iterations are performed, Γ becomes 0.84 and 0.2 for D and E respectively, showing no significant change from the 100^{th} iteration.

Summary. Table 6.1 summarizes the results of this example. The following three sections analyze each of the three variables (rows) in the table for all 2,447 experiments. The respective section is reported in the table.

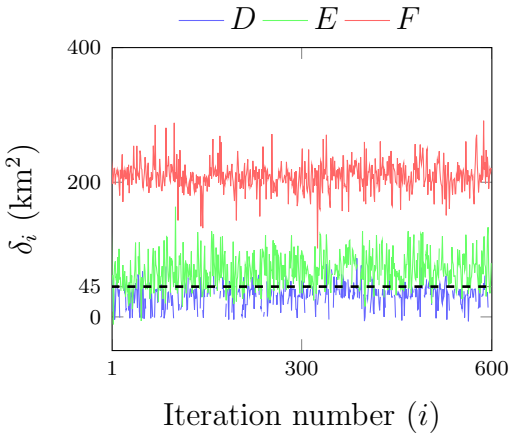
6.2 Triangle Inequality Violations

For each client, four delay-based triangles are calculated at each iteration, three of which have the client as one of the triangle's vertices for a total of $3\times600 =$ 1,800 triangles involving the client. Figure 6.4 shows a CDF of the number of TIVs, resulting from either *mp*-estimated or *av*-estimated delays, for each client (legitimate or adversary). Note that Algorithm 2 does not call $valid(\mathbf{D}^{av})$ if $valid(\mathbf{D}^{mp})$ is true (line 5).[2] We thus counted the number of TIVs for *av* by running a modified version of Algorithm 2 (page 2), where line 5 is removed (and the *else* at the beginning of line 6).

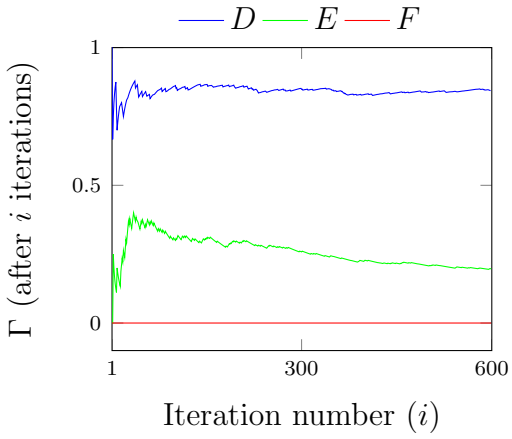
[2]Recall form Section 5.4.2 on page 68, the function $valid(\mathbf{D})$ checks for TIVs in the four triangles whose side lengths are mapped from the six OWDs in $\mathbf{D}$.



(a) The area of the shown triangle is ∼230,000 km^2. Clients E and F are outside, whereas D is inside. Map data: Google, INEGI.



(b)



(c) At $\epsilon_\triangle = 45$ km^2

Figure 6.3: An example from our experiments showing a triangle and three clients (best viewed in color).

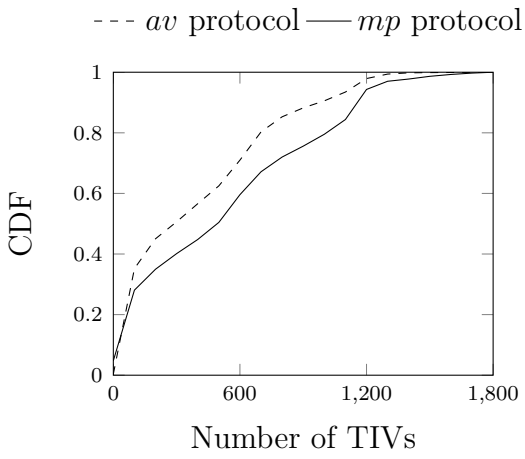


Figure 6.4: Number of TIVs involving the client. A point (x, y) means the proportion y of clients suffered x or fewer TIVs.

For the triangles described by *mp*-estimated delays, very few clients (5%) suffered no TIVs, and 86% suffered at least 10 (of 1,800 possible) TIVs. While these results confirm that TIVs occur frequently in the Internet (*cf.* [161]), they emphasize the importance of iterative delay-measurement to mitigate TIVs. For example, half the clients suffered fewer than 28% (or 500) TIVs in total, enabling CPV to use the remaining 1,300 valid triangles to verify location assertions.

The case was slightly different using *av*-estimated delays; almost all clients suffered at least one TIV and 93% suffered at least 10 of the possible 1,800 TIVs. However, *av* was overall better in avoiding TIVs than *mp*. Half the clients suffered fewer than 300 TIVs (versus 500 for *mp*). Because *av* estimates the OWD of a triangle's side as the average of both directions, it tends to reduce the discrepancy between the three sides, leading to fewer TIVs than *mp*.

6.3 The "Area" as a Discrimination Metric

We analyze the effectiveness of using the areas of triangles (those determined by the verifiers and the client—see Chapter 5) as a metric to distinguish legitimates from adversaries. Figure 6.5 shows a CDF of the median area difference, $\tilde{\delta}$, for all 146 legitimates and 2,301 adversaries. These area differences are either calculated from the *mp* or the *av* protocols (see Algorithm 2). Note that, from Fig. 6.2(b) (Section 6), about one-third of all adversaries were within 400 km of the triangle's sides (e.g., E and F in Fig. 6.3(a) were within 50 km and 850 km of the triangle's side respectively).

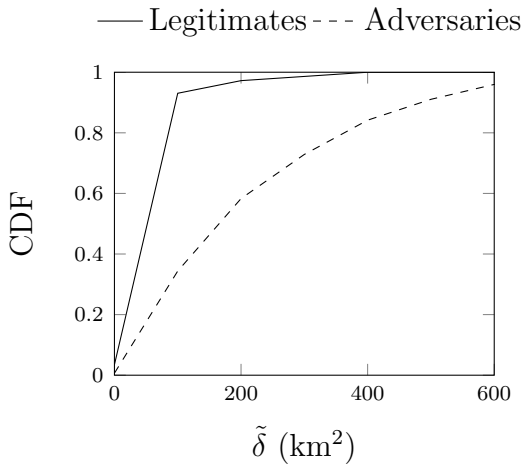


Figure 6.5: Median area difference ($\tilde{\delta}$) for 146 legitimates, and 2,301 adversaries. A point (x, y) means $\tilde{\delta}$ was less than or equal to x km^2 for the proportion y of clients.

The results in Fig. 6.5 show that 93% of all legitimates had $\tilde{\delta}$ <100 km^2, whereas two-thirds of all adversaries had more than that value. The results affirm that, although the experiments involved numerous adversaries that are close to the sides of the triangles encompassing their asserted location, triangular areas distinguished between them. In conclusion, the triangular area served as a successful discrimination metric to distinguish between legitimates and adversaries.

6.4 The Confidence Ratio

Figure 6.6(a) shows the CDF of Γ for legitimates and adversaries; the values of Γ associated with 90% of all adversaries was 0, i.e., certain values for CPV's input parameters led the algorithm to be 100% confident about the absence of those adversaries from the triangles encompassing their asserted location. The case was different with legitimates, where only 30% had a Γ value above 0.5, and half had a value above 0.1. Thus in our experiments, CPV detected falsified location assertions easier than realizing the correctness of true (honest) assertions. The values of $\epsilon_{\triangle}$ that result in this Γ distribution are shown in Fig. 6.6(b).

For FRs and FAs, tolerating one over the other depends on the application using CPV. For example, media broadcasters aiming to assert their legal compliance with license agreements would likely tolerate FAs more than FRs. On the other hand, FRs might be more tolerable for a sensitive banking transactions than FAs. Tuning CPV's input parameters enables applications to control which false decision should

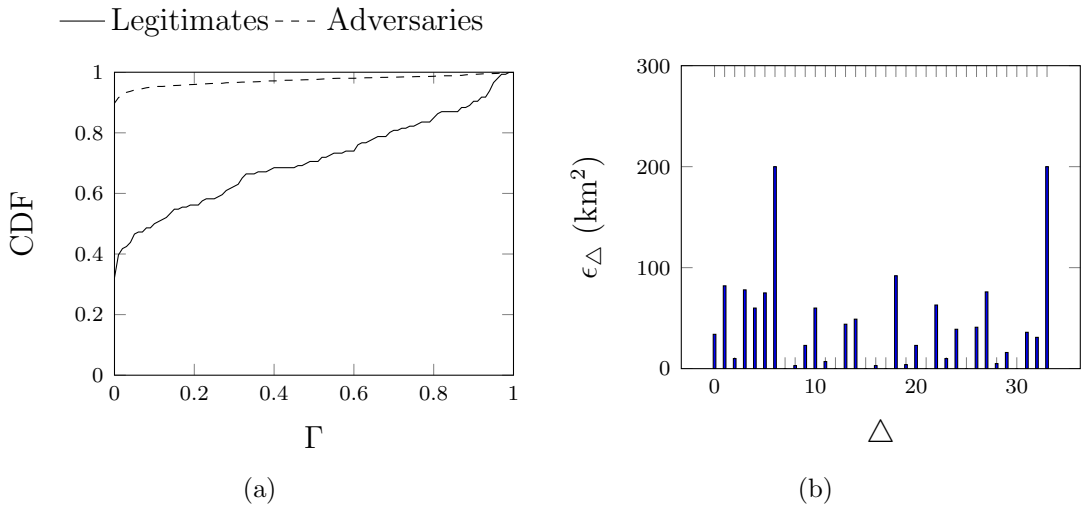


Figure 6.6: (a) Confidence ratios (Γ) for 146 legitimates, and 2,301 adversaries. A point (x, y) means Γ was less than or equal to x for the proportion y of clients. (b) Values of $\epsilon_\triangle$, for each $\triangle$ in the 34-triangles set.

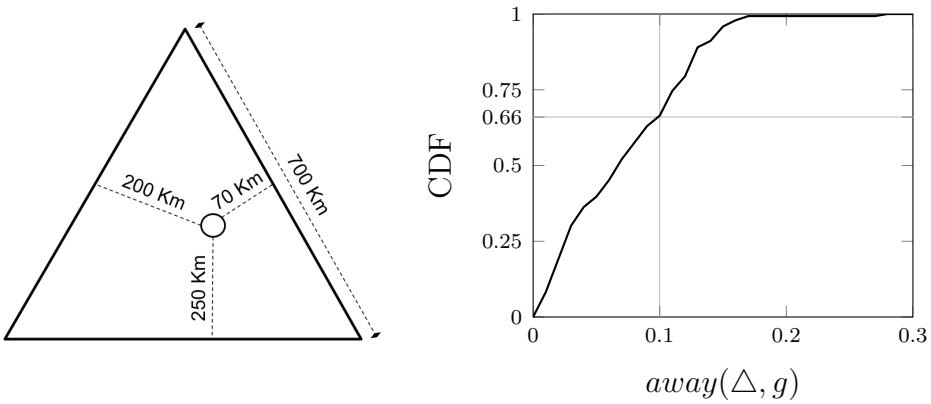
the algorithm tolerate more.

6.5 Proximity to Triangle's Sides

This subsection analyzes the effect of a legitimate's proximity to the sides of its enclosing triangle. Let $away(\triangle, g)$ be the ratio of the distance between a point g inside $\triangle$ and side $z^g_\triangle$ to the length of $z^g_\triangle$, where $z^g_\triangle$ is the closest side to g (see Fig. 6.7(a) for an example). If $away(\triangle, g) = 0$, then g lies on one of the three sides of $\triangle$. We evaluate how CPV's efficacy changes (as expected it improves) as we test with fewer legitimates close to the sides (i.e., with relatively small values of $away()$). Figure 6.7(b) shows a CDF of $away(\triangle, g)$ for all 146 legitimate clients in the experiments with respect to each $\triangle$ in the 34 triangle set. The location g of two-thirds of legitimate clients was such that $away(\triangle, g) \leq 0.1$.

Figure 6.8 shows the number of FRs and FAs after excluding legitimates at locations g, such that $away(\triangle, g) < \lambda$ for all $0 \leq \lambda \leq 0.1$. The number of remaining legitimates is shown on the same chart as the y-axis on the righthand side.[3] All adversaries in our experiments were included in the plot regardless of their triangle proximity. As more legitimates are excluded, the effect of the remaining ones on

[3]Most of the PlanetLab nodes used in our experiments are located within cities, which explains the relatively large number of nodes close to triangles' sides.



(a) $away(\triangle, g) = 70/700 = 0.1$ (b) CDF of $away(\triangle, g)$

Figure 6.7: Legitimates' distances from the triangles' closest side. (a) Calculation of $away(\triangle, g)$; (b) A point (x, y) means the proportion y of adversaries were x km away from the closest side. Note: this graph shows experimental design, not results.

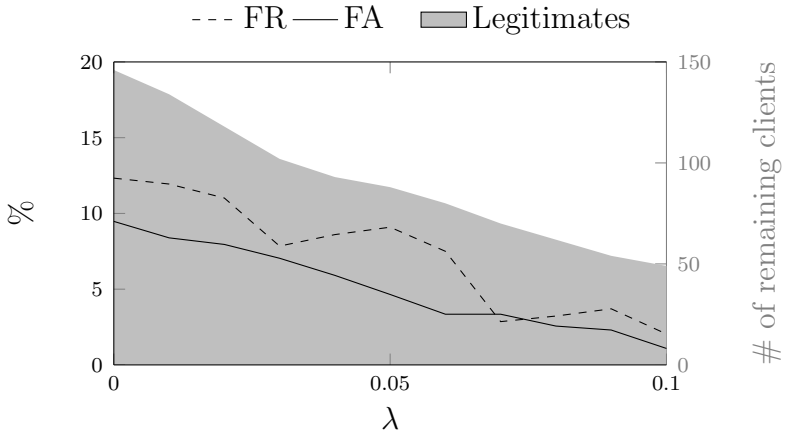


Figure 6.8: FRs and FAs when legitimates at location $g = \{x, y\}$ are excluded from the experiments, such that $away(\triangle, g) < \lambda$. The shaded region is the number of remaining legitimates.

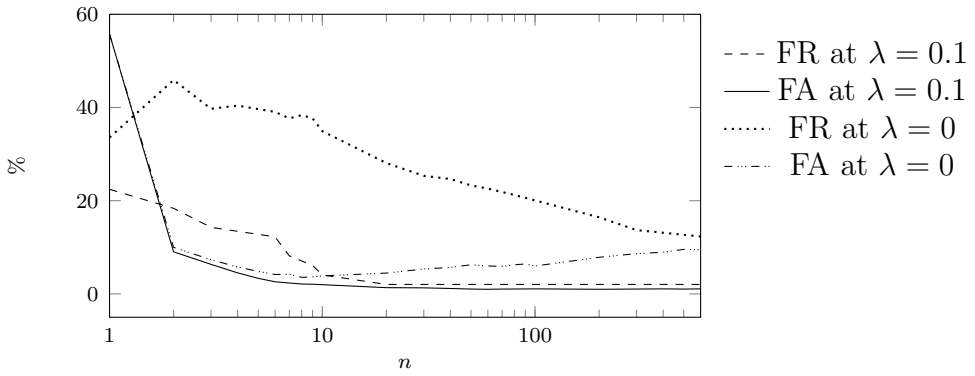


Figure 6.9: FRs and FAs when n iterations in Algorithm 2 (page 69) are performed.

the FRs increases. When the remaining clients suffer relatively high network delays, the FRs oscillate as shown in the plot. Of the chosen PlanetLab nodes, we noticed three nodes suffering exceptionally high delays for unknown reasons. Their distance from the triangle's closest side was such that $0.002 \leq away() \leq 0.28$. Those nodes contribute to the oscillation intensity occurring in Fig. 6.8 as λ increases, and become very hard to partition from adversaries as more legitimates get excluded. At $\lambda = 0.1$, the FRs were 2% versus 12.3% at $\lambda = 0$. This improvement emphasizes the importance of appropriate triangle choice with respect to the asserted location. For an asserted location l, it is recommended that $\triangle_l$ be chosen such that $away(\triangle_l, l) \geq 0.1$.

Although the number of adversaries included in the experiments was unchanged over the spectrum of λ in Fig. 6.8, FAs improve as λ increases; the FAs were 9% at $\lambda = 0$, and dropped to 1.1% at $\lambda = 0.1$. Such improvement stems from the ability to find smaller ϵ values that do not falsely reject legitimates—now far from the triangle's sides, i.e., at $\lambda = 0.1$. Smaller ϵ values reduce FAs.

6.6 Number of Iterations

In this section, we study the effect of the number of CPV iterations, n, on the efficacy of the verification process. Note that large number of iterations comes at the cost of an increased CPV runtime, during which the client is waiting to get its location verified before receiving services.

Figure 6.9 shows the change in FRs and FAs with n ($\log_{10}$ scale). FRs and FAs generally decrease as more iterations are performed, at $\lambda = 0.1$ and $\lambda = 0$. The

results for $\lambda = 0.1$ are quite sensible: FRs and FAs decrease almost monotonically when more iterations are performed. With two iterations, at $\lambda = 0.1$, the FAs dropped to ∼9% from over 50% when only one iteration was performed. Fewer than 10 iterations did not enable the verifiers to identify legitimates appropriately as the FRs were between 6-22%, i.e., no values for $\epsilon_{\triangle}$ and $\tau_{\triangle}$ existed to partition legitimates and adversaries. However, between 10 and 20 iterations, FRs and FAs, at $\lambda = 0.1$, remained at ∼2% and ∼1% respectively.

At $\lambda = 0$, FAs dropped from ∼56% when one iteration was performed, to ∼10% when 9 iterations were performed. It then oscillated between ∼10% and ∼6% when fewer than 100 iterations are performed, climbing steadily to ∼8% for the rest of the iterations. This rise happened simultaneously with an improvement in the FRs (at $\lambda = 0$). As more iterations are performed, it becomes more feasible to find $\epsilon_{\triangle}$ values that partition legitimates from adversaries. To accommodate legitimates that are very close to the triangles' sides, large values of $\epsilon_{\triangle}$ were required, which resulted in falsely accepting more adversaries. This explains the rise in FAs as more iterations were performed, at $\lambda = 0$. Over the entire range of n, the FRs at $\lambda = 0$ decreased from ∼34% at $n = 1$ to ∼12% at $n = 600$. Even when legitimates are highly adjacent to their enclosing triangles' sides, large number of iterations can improve the ability of finding ϵ and τ values that better partition legitimates from adversaries. This highlights the importance of the iterative delay-measurement of CPV (see Algorithm 2 on page 69), especially when the chosen verifiers determine a triangle whose sides are close to the asserted location.

In the conducted experiments, each iteration took six seconds because each verifier sent a probing packet every 2 seconds. Such duration could be modified according to the application's requirements. For example, increasing the duration of each iteration (i.e., increasing the delay between each subsequent probing message) diversifies the network conditions during which the delays are measured. This comes at the cost of increased verification time thus, affecting CPV's usability. In general, a 30 ms delay between network probing/monitoring packets should be sufficient to avoid packet interference [71]. Finding an optimal balance between both ends of the spectrum is left for future investigation.

6.7 *Minimum pairs* versus *Average* protocol

Table 6.2 summarizes PlanetLab results of different CPV evaluation scenarios. The columns represent modified versions of CPV, i.e., different from the behavior given

Table 6.2: Results of modified versions of CPV. The shaded column is the unmodified version—see Algorithm 2 on page 69.

Case	λ	n	*av* only			*mp* only			CPV (*mp* and *av*)		
			FR%	FA%	**FR+FA**	FR%	FA%	**FR+FA**	FR%	FA%	**FR+FA**
1	0	10	45	4.4	**49**	39	3.8	**43**	35	3.9	**39**
2	0	100	25	5.3	**30**	26	4.9	**31**	21	5.1	**26**
3	0	600	14	7.1	**21**	17	6.5	**24**	13	7.3	**20**
4	0.1	10	24	1.7	**26**	10	2.3	**12**	4.1	2.1	**6.2**
5	0.1	100	10	0.7	**11**	2.0	1.0	**3.0**	2.0	1.1	**3.1**
6	0.1	600	2.0	1.7	**3.7**	2.0	1.0	**3.0**	2.0	1.0	**3.0**

λ = legitimates-exclusion threshold (see Section 6.5); n = number of iterations (see Algorithm 2); *av* = the "average" protocol; *mp* = the "minimum pairs" protocol.

in Algorithm 2. In line 4 of Algorithm 2, two OWD-estimation protocols are used (*mp* and *av*) to alleviate the effect of TIVs. Table 6.2 lists the results when only the *av* protocol is used ("*av* only" column), when only *mp* is used ("*mp* only" column), and when both are used ("CPV" column). The results are shown for various combinations of the exclusion threshold, λ (see Section 6.5), and the number of iterations, n. The table shows the FRs, the FAs, and their sum in each respective case.

From Table 6.2, the summation of FRs and FAs when both OWD-estimation protocols are used (right-most column under "CPV") is smaller in four out of six of the cases (table rows) compared to the summation when each protocol is used solely, e.g., 39 is less than 43 and 49 in the first case. Thus, the use of both OWD-estimation protocols tends to enhance the accuracy of the location verification process.

Using the *mp* protocol solely gave better results than *av* solely in four out of six cases. The *av* protocol was better at $\lambda = 0$ and $n \geq 100$. Recall from Section 6.2 that the *mp* protocol results in more TIVs. Since CPV counts the number of TIVs against the client, more TIVs tend to increase FRs, as shown by the results under the "*mp* only" column in Table 6.2. At $\lambda = 0$ and $n \geq 100$, there were 26% and 17% FRs using the *mp* protocol, versus 25 and 14% using *av*. In conclusion, CPV works best when utilizing both delay-estimation protocols to mitigate the unfavorable effect of TIVs.

6.8 Conclusion

Three remarks can be made in conclusion from the evaluation conducted in this chapter.

1. Reducing the factors that negatively affect the delay-to-distance mapping process (such as TIVs [161]) improves the accuracy of the location verification process. CPV leverages several heuristics to reduce such factors, e.g., iterating the delay-measurement process and using multiple delay-estimation protocols. The results in Sections 6.2, 6.6 and 6.7 provide evidence that CPV's accuracy improves upon applying these heuristics.

2. Comparing the areas of triangles projected using the delays between three verifiers and a client enables the verifiers to realize if the client is geographically encapsulated by the triangle determined by the verifiers's locations. Section 6.3 provide evidence supporting this conjecture.

3. The adjacency of a legitimate client to the sides of the triangle enclosing their geographic location can dramatically affect the correctness of CPV's verification. From the analysis in Section 6.5, clients that were away of the triangle's closest side at least 10% of the length of that side were likely to get their assertions correctly accepted.

In summary, the evaluation conducted in this chapter using a real world experimental testbed with wired-connected clients shows that certain CPV parameterization enabled the algorithm to FR and FA rates of 2% and 1% respectively. However, to achieve these results in practice, a sufficient number of verifiers must be available to find the appropriate triangles, ones whose sides are far enough from the asserted location (see Section 6.5).

From a geographic perspective, all triangles used in the experiments conducted in this chapter had side lengths ranging from ∼260 km to ∼1,100 km; the reported results pertain to this range. Due to the increased route circuitousness (see Section 2.1.1, page 9) that happens with short distances over the Internet [93], extremely small triangle sizes are expected to result in higher FR/FA rates. The rate by which the results worsens as triangles become smaller is left for future exploration.

Since the triangle size is the verification granularity, larger triangles may become less practical from the application's perspective. However, some applications may only need coarse verification granularity, e.g., to preserve user's privacy; larger triangles in that case may be beneficial.

In the next chapter, CPV will be similarly evaluated, but with legitimate clients modeled to use 802.11 (wireless) access networks.

Chapter 7

Evaluation with Wireless CPV Clients

The nature of delays in wireless and wired networks is different. This chapter evaluates CPV when legitimate clients (those inside the triangles) are connected through WiFi access networks. In the rest of this thesis, we refer to those clients simply as *wireless clients*. A wireless client is assumed to be one hop away from its access point, which serves as the client's gateway to the Internet. Beyond the gateway, all hops until the verifiers are assumed to be wired. That is, none of the verifiers are assumed to use a wireless access network, e.g., satellite, which is a reasonable assumption since the location verification service provider is assumed to own/control the verifier infrastructure.

In Chapter 6, CPV was evaluated with clients connected through wired access networks. Using the PlanetLab testbed, evaluation was performed by having sets of three verifiers (running on PlanetLab nodes) measure OWDs to/from legitimate clients and adversaries using the *mp* and *av* protocols (Chapter 4). The measured OWDs where logged, and the CPV algorithm (Chapter 5) was run locally on the collected logs. Knowing the ground truth of inside and outside clients (i.e., legitimates and adversaries), CPV's false reject/accept rates were quantified.

To evaluate CPV in wireless networks, we use the OWDs collected in Chapter 6 between the client and the verifiers, and add an additional delay component to each delay value to model wireless transmission. The added component represents the single-hop delay between the wireless client and its access point, and is modeled as a random variable that follows wireless latency-distributions studied in the literature [29].

Table 7.1: Combinations of access networks for a legitimate client and an adversary

Legitimate client	Adversary	Chapter
Wired	Wired	6
Wired	Wireless	–
Wireless	Wired	7
Wireless	Wireless	–

Assume two clients, a legitimate and an adversary, both having their location assertions verified by CPV. Their access networks follow one of the four combinations shown in Table 7.1. The table also shows in which chapter the combination is explored. Wireless adversaries are not modeled in this thesis. The reason is that wireless networks tend to, among other effects, increase delays and the delay variance, which in CPV increase the likelihood of rejecting assertions. Therefore, by modeling wireless legitimates and wired adversaries, we test CPV in the most demanding (to the defender) situation among the four possible combinations in Table 7.1.

This evaluation methodology addresses the effect of delays in wireless networks, while retaining the advantages of PlanetLab, e.g., real-world network delays, logical and geographical network topology, exterior gateway routing policies, congestion behavior. In addition, by using the data logs collected from the wired evaluation phase (Chapter 6), we unify all experimental parameters across wireless and wired testing. Root causes of improvement/retrogression can then be more reliably identified.

This chapter aims to study the impact of the varying wireless delays on CPV, by specifically exploring the following three questions:

1. **Assuming k wireless devices actively competing for the wireless media with the legitimate client, how does k affect CPV?** Here, the number of wireless legitimate clients is varied, and CPV's efficacy is analyzed. We test by modeling clients using IEEE 802.11b as a representative access technology.

2. **For a given triangle verifying assertions of wireless legitimates and a wired adversary, what is the minimum distance the adversary should be away from the triangle's nearest side so that CPV correctly rejects it?** To answer this question, we test CPV when varying the width of the adversary-free region outside the triangle. We do this by progressively excluding nearby adversaries from the experiments and reevaluating CPV.

3. **How many CPV iterations should the verifiers perform in order to**

essentially eliminate the effect of the additional wireless delays? As explained in Chapter 5, the verifiers in CPV estimate the delays iteratively. We derive the number of iterations required to essentially eliminate the effect of the wireless networks, as a function of the number of wireless devices k and the acceptance threshold τ (see Chapter 5).

Chapter Roadmap. Section 7.1 provides background on the mechanisms by which 802.11 networks manage access to the shared medium. Section 7.2 reviews recent literature that models delays of single-hop wireless networks. The reviewed models are then used to evaluate CPV in Section 7.3. Section 7.4 analyzes the effect of the number of iterations on the efficacy of CPV when legitimate clients are using wireless access networks.

7.1 Background on 802.11

Distributed Coordination Function (DCF) is the technique used in IEEE 802.11 (wireless) networks [79] to manage access to the shared wireless media [89]. It employs the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) method.

In DCF, when the Medium Access Control (MAC) layer of a device has a data frame to send, it checks if the medium is busy and starts transmission if it is free for a length of time called the Distributed (coordination function) Interframe Space (DIFS) [89]. If the medium is busy, the device backs off for X time slots, where X is a number chosen uniformly at random in the range $[0, W_{\min}]$. The countdown of the back-off timer is paused whenever a transmission (i.e., from other devices) is sensed. The device transmits only if the media was found vacant for a period equal to DIFS after the back-off time reaches zero. Otherwise, the device backs off for another uniformly-chosen random number of time slots in the range $[0, 2 \cdot W_{\min}]$. The process is repeated as long as the medium is sensed to be busy anytime during the countdown, with the back-off interval doubling on each repetition until it reaches a maximum of $W_{\max} = 2^m \cdot W_{\min}$, for some predefined value m.

Upon successful reception, the receiver sends an Acknowledgement (ACK). Transmitting the ACK follows the DCF procedure described above. If an ACK is not received, the sender of the original data frame attempts several further retransmissions following the DCF procedure, and eventually gives up if those fail.

If two wireless devices, A and B, using one access point are not in the transmission

ranges of each other, they are said to be *hidden terminals*. A and B may thus fail to sense each others' transmission, in which case simultaneously transmitting may cause collision at the access point. To address the hidden terminal problem [55], Request to Send and Clear to Send (RTS/CTS) frames are optionally used. If A is the device with data to send, it first sends an Request to Send (RTS) control frame to the access point. This frame indicates the time A needs to send its data frame and receive the ACK. The access point responds by broadcasting a Clear To Send (CTS) containing such timing information, which would also be received by B. B then refrains from using the medium for the specified period of time. The analysis included throughout this chapter considers the case whereby RTS/CTS frames are used.

7.2 Wireless Delay Models in the Literature

This section reviews two wireless delays models in the literature, both assume a single-hop wireless network with one access point and k wireless devices. The k devices are *saturated*, i.e., always have frames to send. The channel is assumed ideal, meaning that the only source of frame corruption is collision.

Note that the focus of this section is not to compare the two wireless delay models, nor not to evaluate their accuracies. We rather review these models to use them in evaluating CPV later in Sections 7.3 and 7.4 below.

7.2.1 Average back-off time at a stage

Carvalho and Garcia-Luna-Aceves [29] derived the average time a device spends backing off. Recall from Section 7.1 that a device backs-off for $X = \mathcal{U}\{0, 2^m \cdot W_{\min}\}$ time slots. Thus, the expected backing-off time, α, is the time spent while counting down X time slots plus the time where the countdown is paused during a sensed transmission [29]:

$$\alpha = \sigma p_i + t_c p_c + t_s p_s \tag{7.1}$$

The constant σ is the length of the time slot (in μsec); p_i is the probability the channel is idle (i.e., the subscript is not an index, it denotes "idle") during a time slot; and p_c and p_s are the probabilities of collision and successful transmission respectively during a time slot. t_s and t_c are the number of time units a device spends while pausing the countdown during a successful transmission and during a

transmission with collision respectively. Bianchi *et al.* [20] expressed these durations as follows:

$$t_s = \frac{l(\text{RTS}) + l(\text{CTS}) + l(H) + l(P) + l(\text{ACK})}{\text{rate}} + (3 \cdot \text{SIFS} + \text{DIFS}) + 4\delta \quad (7.2)$$

$$t_c = \frac{l(\text{RTS})}{\text{rate}} + \text{DIFS} + \delta \quad (7.3)$$

where the function $l(.)$ indicates the frame (or packet) length in bits; RTS/CTS are the Ready/Clear To Send frames (see Section 7.1); δ is the propagation delay (in μsec); SIFS is a technology-specific amount of time (in μsec); H, P and ACK are the header, data packet, and acknowledgement packets respectively; and *rate* is the media's transmission rate in Mbps.

Using a 2-dimensional discrete-time Markov process, Bianchi *et al.* derived the probability, ψ, that a transmission occurs (successful or with collision) at a time slot as:

$$\psi = \frac{2(1-2p)}{(1-2p)(W_{\min}+1) + pW_{\min}(1-(2p)^m)} \quad (7.4)$$

where p is the probability of collision occurring at a time slot. Note that p is different from p_i, p_c and p_s in (7.1). Bianchi *et al.* [20] then assumed that a packet collides with a constant and independent probability regardless of the number of retransmissions it suffers. Assuming k devices in the network, if one device transmits, the only case that results in no collision is when none of the $k-1$ other devices transmit, i.e., the probability of no collision is $(1-\psi)^{k-1}$. Therefore, p can be expressed in terms of ψ as [20]:

$$p = 1 - (1-\psi)^{k-1} \quad (7.5)$$

Thus, the relationship between p and ψ is non-linear. Carvalho and Garcia-Luna-Aceves [29] linearized this model in order to use ψ to derive the expected total back-off time (see Section 7.2.2 below).

Using ψ and assuming k devices, the probability (P_{tr}) that at least one of the k devices is transmitting, and the probability (P_{suc}) that a transmission for any of the k devices is successful are calculated as follows [19, 20]:

$$P_{\text{tr}} = 1 - (1-\psi)^k$$

$$P_{\text{suc}} = \frac{k\psi(1-\psi)^{k-1}}{P_{\text{tr}}}$$

The probabilities p_i, p_c and p_s in (7.1) are therefore calculated as $p_i = 1 - P_{\text{tr}}$,

$p_c = P_{\text{tr}}(1 - P_{\text{suc}})$, and $p_s = P_{\text{tr}}P_{\text{suc}}$ [29].

7.2.2 Expected total back-off time

Carvalho and Garcia-Luna-Aceves [29] give an approximate solution to the nonlinear relation between ψ in (7.4) and p in (7.5), and reduce ψ to:

$$\psi = \frac{2W_{\min}}{(W_{\min}+1)^2}(1-p) \tag{7.6}$$

Using (7.6), the authors derived p independent of ψ as [29]:

$$p = \frac{2W_{\min}(k-1)}{(W_{\min}+1)^2 + 2W_{\min}(k-1)}$$

Carvalho and Garcia-Luna-Aceves [29] then used this approximation to obtain α in terms of σ, k, $W_{\min}$, t_s and t_c, as explained above. Finally, they derived the expected time a device backs off $\overline{T}_B$ as [29]:

$$\overline{T}_B = \frac{\alpha(W_{\min}F - 1)}{2q} + \left(\frac{1-q}{q}\right)t_c \tag{7.7}$$

where

$$F = \frac{q - 2^m(1-q)^{m+1}}{1-2(1-q)}$$

and $q = 1 - p$ represents the probability of no collision.

7.2.3 Mean delay and jitter, the model of Carvalho *et al.*

Carvalho and Garcia-Luna-Aceves [29] expressed the expected delay $E[T]$ of a frame as the expected time a device backs off $\overline{T}_B$ in (7.7) plus the frame transmission time t_s in (7.2):

$$E[T] = \overline{T}_B + t_s \tag{7.8}$$

The variance of T was derived as:

$$\text{Var}[T] = \left[\frac{\alpha(W_{\min}\gamma - 1)}{2} + t_c\right]^2 \frac{1-q}{q^2}$$

where

$$\gamma = \frac{(2q^2 - 4q + 1 - mq(2q-1))(2-2q)^m + 2q^2}{(2q-1)^2}$$

Thus the jitter (or the standard deviation) is:

$$\mathrm{Std}[T] = \sqrt{\mathrm{Var}(T)} \tag{7.9}$$

7.2.4 CDF of delays

The model of Carvalho and Garcia-Luna-Aceves [29] only provides information about the mean and jitter of the delays given some number of wireless devices k. We assume that delays will follow a Gaussian distribution with mean and variance derived as in (7.8) and (7.9) respectively. However, since the distribution (which would be the delays in that case) goes from $-\infty$ to ∞, the model can result in negative delay values. Thus, we assume a truncated Gaussian [82] in the range $[0, \infty]$.

The mean of the Gaussian distribution truncated from a to b is given by [82]:

$$\mathrm{GausMean}_{\mu,\sigma}(a, b) = \mu - \sigma \cdot Z(\alpha, \beta)$$

where μ and σ are respectively the mean and standard deviation of the parent (non-truncated) Gaussian distribution; $\alpha = (a-\mu)/\sigma$ and $\beta = (b-\mu)/\sigma$; and the function $Z(.)$ is defined as:

$$Z(\alpha, \beta) = \frac{\phi(\beta) - \phi(\alpha)}{\Phi(\beta) - \Phi(\alpha)}$$

The functions $\phi(.)$ and $\Phi(.)$ are respectively the PDF and the CDF of the standard (i.e., with $\mu = 0$ ms and $\sigma = 1$ ms) Gaussian distribution.

The standard deviation of the Gaussian distribution truncated from a to b is [82]:

$$\mathrm{GausStd}_{\mu,\sigma}(a, b) = \sqrt{\sigma^2 \cdot \left(1 - \frac{\beta \cdot \phi(\beta) - \alpha \cdot \phi(\alpha)}{\Phi(\beta) - \Phi(\alpha)} - Z^2(\alpha, \beta)\right)}$$

To obtain a CDF of the wireless delays that has a mean and standard deviation as in (7.8) and (7.9), we need to solve simultaneously for μ and σ:

$$\mathrm{GausMean}_{\mu,\sigma}(0, \infty) = E[T] \tag{7.10}$$

and

$$\mathrm{GausStd}_{\mu,\sigma}(0, \infty) = \mathrm{Std}[T] \tag{7.11}$$

Those are two equations in two unknowns, which can be solved using numerical methods. Finally, using μ and σ, the CDF of the Gaussian distribution truncated

Table 7.2: Mean μ, and standard deviation σ, of the single-hop wireless delays when k devices are simultaneously competing with the media.

Parameters (ms)	Eqn.	k						
		2	3	4	5	10	20	30
$E[T]$	(7.8)	2	3	4	5	12	40	87
Std$[T]$	(7.9)	0.6	1.6	2.9	4.7	21	89	186
μ	–	-110	-159	-208	-246	-691	-2419	-5156
σ	–	15	22	29	36	95	328	700

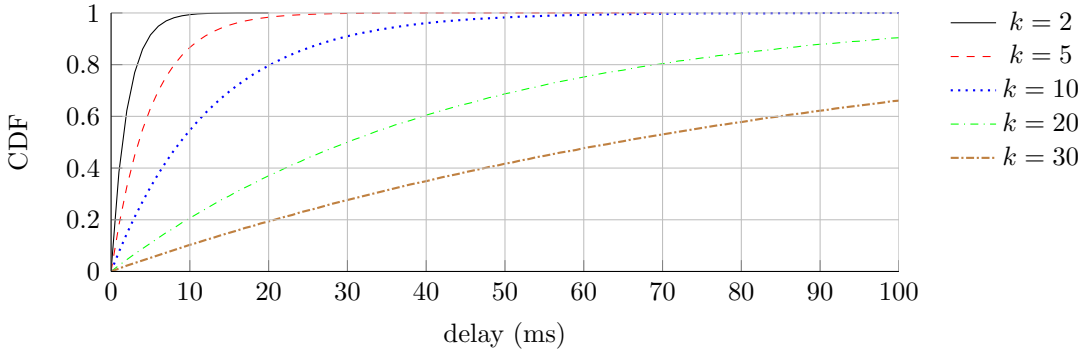


Figure 7.1: Truncated Gaussian CDFs of single-hop wireless delays that a frame endures when there are k saturated wireless devices in the network.

from a to b is [82]:

$$\text{GausCDF}_{\mu,\sigma}(x; a, b) = \frac{\Phi(\zeta) - \Phi(\alpha)}{\Phi(\beta) - \Phi(\alpha)} \tag{7.12}$$

where $\zeta = (x - \mu)/\sigma$. Table 7.2 shows the mean and standard deviations calculated using (7.8) and (7.9) for various values of k, and the corresponding μ and σ of the parent (non-truncated) Gaussian distribution calculated by solving (7.10) and (7.11) simultaneously.

Figure 7.1 plots the delay distribution, $\text{GausCDF}_{\mu,\sigma}(x; 0, \infty)$, using (7.12) for various values of k. Unsurprisingly, the chart shows that the wireless delays generally increase with k. These delay distributions are used in Sections 7.3 and 7.4 to evaluate CPV in wireless networks.

The model of Carvalho and Garcia-Luna-Aceves provides an upper bound on the average delay a frame is expected to suffer [29]; when they compared their model to simulations, delays from the simulations were always smaller. One reason for the simulation delays being smaller is that there is a non-zero probability that a frame backs off indefinitely [29]. However, the DCF standard [79] specifies that the MAC layer must discard the frame if transmission failed after R back off trials, for some

predefined value of R. Transmission retrials from upper layers may then take care of the discarded frames.

7.2.5 CDF of delays, the model of Raptis *et al.*

Similar to Carvalho and Garcia-Luna-Aceves [29], Raptis *et al.* [128] used the basis of Binachi [20] to derive a CDF (and jitter) for the single-hop 802.11 access delays. However, Raptis *et al.* [128] took into consideration the reality that the frame being transmitted will be discarded after failing transmission in R back-off stages. The authors [128] began by deriving the expected delay that a frame suffers after a failed transmission at stage j ($0 \leq j \leq R$) as:

$$U_j = (j+1) \cdot t_c + \alpha \cdot \sum_{i=0}^{j} \frac{W_i - 1}{2} \tag{7.13}$$

where t_c and α are analogous to those in (7.3) and (7.1) respectively, and

$$W_i = \begin{cases} 2^i \cdot W_{\min}, & \text{if } 0 \leq i \leq m \\ 2^m \cdot W_{\min}, & m < i \leq R \end{cases} \tag{7.14}$$

To derive the CDF of delays, Raptis *et al.* [128] first calculated the probability that a frame is successfully transmitted at stage j as:

$$Q_j = \frac{p^j(1-p)}{1-p^{R+1}} \tag{7.15}$$

Since at any stage j, selecting any back-off value in the range $0 \leq i < W_j$ is equiprobable, then the probability of transmitting a frame at stage j after backing off for i stages is (independent of i):

$$P_j = Q_j \cdot \frac{1}{W_j} \tag{7.16}$$

Using (7.16), Raptis *et al.* [128] derive the CDF of delays as follows. Let Ω be a finite set of delays, such that $\Omega_{j,i}$ is the delay a frame suffers before it gets successfully transmitted at stage j, given that i back-off slots were selected at stage j. For any randomly-chosen delay value D, the probability that $D \leq d$ for all $0 \leq d \leq \infty$ is

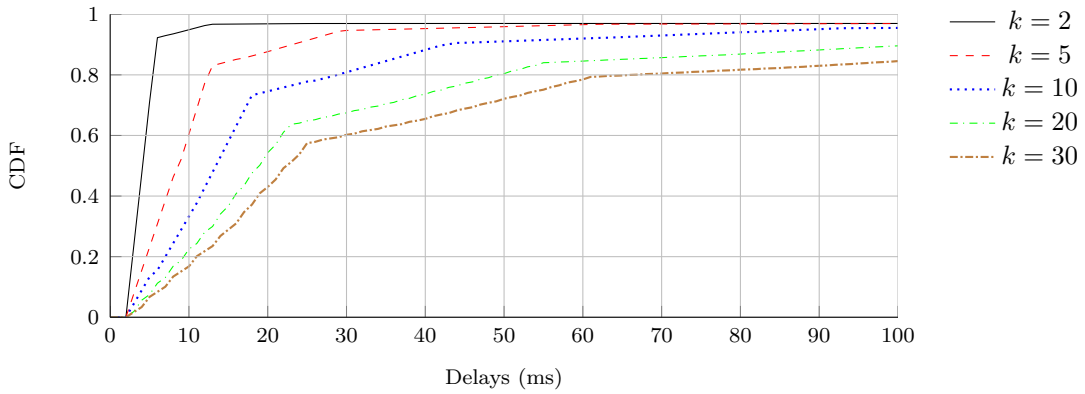


Figure 7.2: CDF of single-hop wireless delays that a frame endures when there are k saturated wireless devices in the network [128].

given by [128]:

$$P\{D \leq d\} = \sum_{j=0}^{R} \sum_{i=0}^{W_j - 1} P_{j,i}(d) \tag{7.17}$$

where

$$P_{j,i}(d) = \begin{cases} P_j, & \text{if } \Omega_{j,i} \leq d \\ 0, & \text{otherwise} \end{cases} \tag{7.18}$$

Using (7.17), Fig. 7.2 plots the wireless delay CDFs of Raptis *et al.* [128] at various values of k. Once again, the model shows that delays generally increase with k, which is unsurprising. However the distributions derived by Raptis *et al.* [128] (Fig. 7.2) are not exactly similar to those derived by Carvalho and Garcia-Luna-Aceves [29] (Fig. 7.1). Differences between both models are discussed in Section 7.2.6 below.

Jitter

Similar to Carvalho and Garcia-Luna-Aceves [29], Raptis *et al.* [128] also derived an expression for the delay jitter in a single-hop wireless network with k devices. To do that, the authors [128] first derived the expected total delay that a frame suffers before being successfully transmitted at stage j as:

$$\omega_j = U_j - t_c + t_s \tag{7.19}$$

Then, using (7.19) and (7.15), the expected delay, $E[T]$, a frame suffers before being successfully transmitted is [128]:

$$E[T] = \sum_{j=0}^{R} (\omega_j \cdot Q_j) \tag{7.20}$$

And the expected value for the square of a delay, T^2, is [128]:

$$E[T^2] = \sum_{j=0}^{R} \left(P_j \cdot \sum_{i=0}^{W_j - 1} (E[\Omega_{j,i}])^2 \right) \tag{7.21}$$

where $E[\Omega_{j,i}]$ is the average of $\{\Omega_{0,0}, .. \Omega_{j,i}\}$, and is calculated as [128]:

$$E[\Omega_{j,i}] = t_s + i \cdot \alpha + U_{j-1} \tag{7.22}$$

Finally, in contrast to the delay jitter of Carvalho and Garcia-Luna-Aceves [29] in (7.9), the jitter of Raptis *et al.* [128] is calculated using (7.21) and (7.20) as:

$$\text{Std}[T] = \sqrt{E[T^2] - (E[T])^2} \tag{7.23}$$

In Sections 7.3 and 7.4, we use the CDFs in (7.12) and (7.17) to evaluate CPV.

7.2.6 Differences between the models

Figure 7.3(a) plots the truncated Gaussian distribution with the parameters obtained from the model of Carvalho and Garcia-Luna-Aceves [29] modeling single-hop wireless delays, and the distribution derived by Raptis *et al.* [128] at $k = 2$ and $k = 10$. The distributions are not drastically different. Their dissimilarities might however stem from differences in their assumptions, e.g., Raptis *et al.* assumes the frame is discarded after failing transmissions in R stages, while Carvalho *et al.* does not make this assumption.

Figure 7.3(b) shows the difference in the jitter between both models, obtained using (7.9) and (7.23) respectively. At first glance, the individual values of the two curves over the region up to $k = 20$ are reasonably similar, but the model of Raptis *et al.* appears almost linear, while that of Carvalho *et al.* gives values lower in the region up to $k = 20$, but rising much faster starting for values shortly beyond $k = 20$.

In the rest of this chapter, both models are used to analyze CPV in wireless networks,

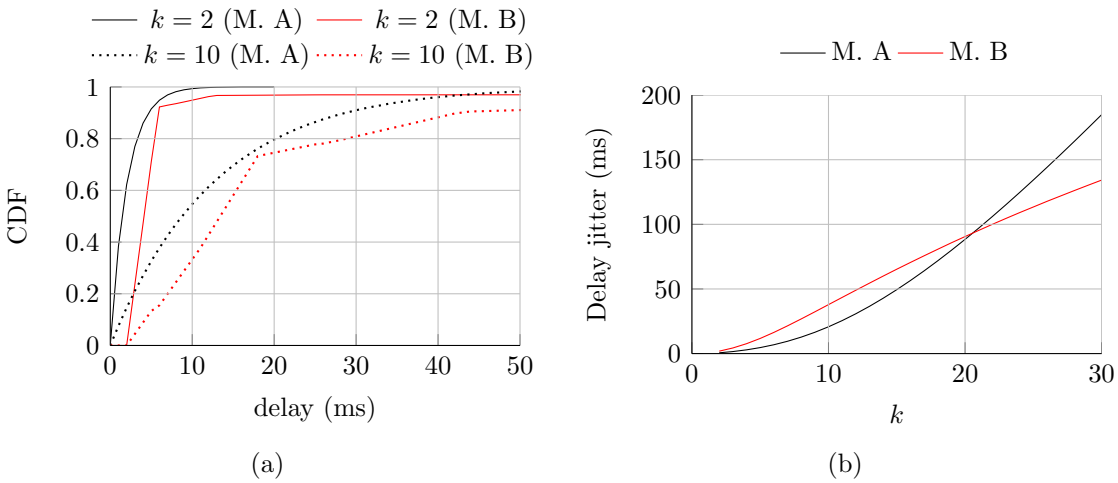


Figure 7.3: Comparison of the reviewed models. M. A means using the model of Carvalho *et al.* [29]; M. B means using the model of Raptis *et al.* [128]. (a) Truncated Gaussian delay distribution with parameters derived from the model of Carvalho *et al.* [29], and the distribution derived by Raptis *et al.* [128] at $k = 2$ and $k = 10$. (b) The jitter follows that derived by the authors [29, 128].

with a truncated Gaussian distribution assumed for the parameters of Carvalho and Garcia-Luna-Aceves.

7.2.7 Summary of reviewed literature on wireless models

All the models reviewed herein, in Section 7.2, consider a wireless network with a single access point and no hidden terminals, typically addressing a small (e.g., home) network. In public places (e.g., coffee shops or hotel rooms), this may not be the case. However, the models already incorporate the additional delays due to the RTS/CTS mechanism of the 802.11 DCF and thus, we believe the existence of hidden terminals is unlikely to result in significant difference in delays.

Another assumption made in the reviewed literature is that the physical media is error-free; in other words, failed transmissions are only caused due to collision. The reviewed literature have compared their analytical models using simulations, which highlighted almost negligible effect of these assumptions in practice [20, 29, 128].

The reviewed literature assumes all k devices are saturated (i.e., always have packets to send). However, k devices are typically expected to alternate between phases of transmission, reception and idle activity. We believe this assumption tends to cause the delays resulting from the derived models to be larger than those in practice.

7.3 Evaluating CPV in 802.11 Networks

We evaluate CPV with wireless clients using the delay models discussed in Section 7.2. All results reported in this chapter follow CPV's recommendation of $\lambda = 0.1$ (see Chapter 6). The area tolerance $\epsilon_{\triangle}$ and the acceptance threshold $\tau_{\triangle}$ are calibrated per triangle. Similar to Chapter 6, the objective is to quantify the FRs and FAs at some values of $\epsilon_{\triangle}$ and $\tau_{\triangle}$ that allow CPV to adequately distinguish legitimates from adversaries.

To analyze CPV with wireless clients, we varied the number of legitimate clients modeled to use wireless access networks.[1] The number of wireless legitimate clients in each $\triangle$ affects the calibration of CPV's input parameters ($\epsilon_{\triangle}$ and $\tau_{\triangle}$), and is thus expected to affect the overall results. Each wireless network was modeled to have k actively-transmitting wireless devices, with one of those k being CPV's legitimate client.

Recall from Chapter 6 that at $\lambda = 0.1$, our PlanetLab experiments had 49 legitimate clients. Thus, we can model a maximum of 49 distinct wireless access networks, with $k \geq 2$ wireless devices in each. For example, if a proportion of $\sim$0.2 of all 49 legitimate clients was using a wireless access network with $k = 4$, this means there are 10 distinct wireless access networks modeled at different geographic regions, and each network has 4 wireless devices (constant across all 10 networks). Fig. 7.4 shows an example of eight legitimate CPV clients; a proportion equal to 0.5 of them is using a wireless access network that has $k = 2$ devices.

7.3.1 Evalution assumptions (wireless access)

Each wireless legitimate client is assumed to be competing for the wireless media with $k - 1$ other wireless devices. All k devices (i.e., including the legitimate client whose assertion is being verified by CPV) use the same wireless access point, which is one hop away. We assume no hidden terminals (recall Section 7.1)—the transmission of any device is sensed by all others.

All k devices are using an 802.11b access network over Direct-Sequence Spread Spectrum (DSSS) on the physical layer with a 11Mbps data rate. Characteristics of DSSS are shown in Table 7.3. Following the reviewed models in Section 7.2, all k devices are assumed *saturated* (i.e., the packet queues of all k device are never

[1] Characteristics of such a network are explained in Section 7.3.1 below.

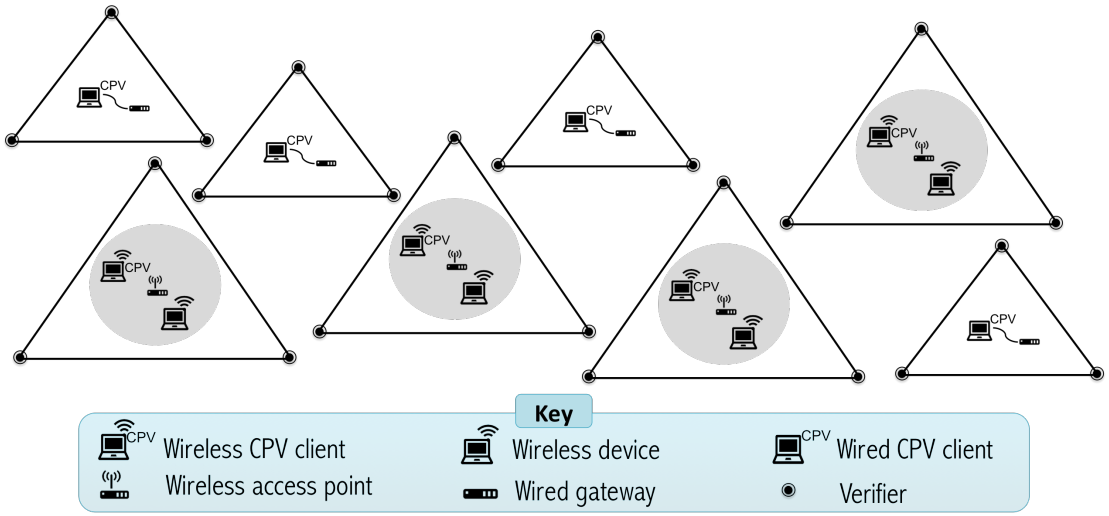


Figure 7.4: An example of eight CPV clients, half of which are using a wireless access network that has $k = 2$ devices.

Table 7.3: DSSS characteristics

Item	Value
W_{min}	32 time slots
W_{max}	1024 time slots
Retransmission limit (R)	6 stages
Physical header (PHY)	192bits at 1 Mbit/s
MAC header	224 bits at 11 Mbit/s
ACK length	112 bits at 11 Mbit/s + PHY
RTS length	160 bits at 1 Mbit/s + PHY
CTS length	112 bits at 1 Mbit/s + PHY
Propagation delay (δ)	1 μsec
Slot time (σ)	20 μsec
SIFS	10 μsec
DIFS	50 μsec

empty), and are transmitting at the same time according to a Constant Bit Rate (CBR) with a packet size equal to 8148 bits.

Finally, because an element of randomness (i.e., the delay component resembling a wireless network) is now introduced to the results, experimentation scenarios were run 10 times and the average result is reported.

7.3.2 Effect of number of wireless devices (k) on CPV

Figure 7.5 shows the mean FRs and FAs of 100 runs resulting from using the models of Carvalho *et al.* [29] and Raptis *et al.* [128]. All 49 legitimate clients were using

Table 7.4: SE and Margin of Error (ME) at 90% confidence level for the rest of the results

Model	Parameter	Std	SE	ME at 90% CI
Carvalho *et al.* [29]	FRs	0.97	0.097	±0.16
	FAs	0.74	0.074	±0.12
Raptis *et al.* [128]	FRs	0.92	0.092	±0.15
	FAs	0.14	0.014	±0.02

Std = Standard deviation; SE = Standard error; ME = Margin of Error.

a wireless access network, and there was a total of $k = 5$ devices in the network of each wireless CPV client. The number of CPV iterations (see Chapter 5) was fixed at $n_{\triangle} = 600$ for all $\triangle$. FRs and FAs for both models lied between ∼1.8% and ∼4.5%.

Because FRs and FAs are estimated empirically from 100 runs, we calculate the error margin of these estimates for a 90% confidence level. To calculate the error margin, we first calculate the critical value as follows [104]:

$$\alpha = 1 - \frac{\text{confidence level}}{100} = 1 - 0.9 = 0.1$$

$$\text{Critical Probability } (p*) = 1 - \frac{\alpha}{2} = 1 - \frac{0.1}{2} = 0.95$$

$$\text{Degree of Freedom } (df) = n - 1 = 100 - 1 = 99$$

From the statistics tables [104], at $df = 99$ and $p* = 0.95$, the critical value is 1.66.

Next, we calculate the standard error (SE). For the FRs obtained using the model of Carvalho *et al.* [29]:

$$\text{SE (FRs)} = \frac{\text{Std}}{\sqrt{n}} = \frac{0.97}{\sqrt{100}} = 0.097. \tag{7.24}$$

Table 7.4 shows the SE for the rest of the results. Finally, the Margin of Error (ME) at 90% confidence level is calculated as:

$$\text{ME (FRs)} = \text{critical value} \times \text{SE} = 1.66 \times 0.097 = 0.16. \tag{7.25}$$

The ME at 90% confidence level for the rest of the results is reported in Table 7.4. The MEs at 90% confidence level are depicted using vertical lines atop the bars in Figure 7.5 for the mean FRs and FAs. None of the MEs exceeds ±0.16%, highlighting that the means estimated from the sample runs are relatively precise.

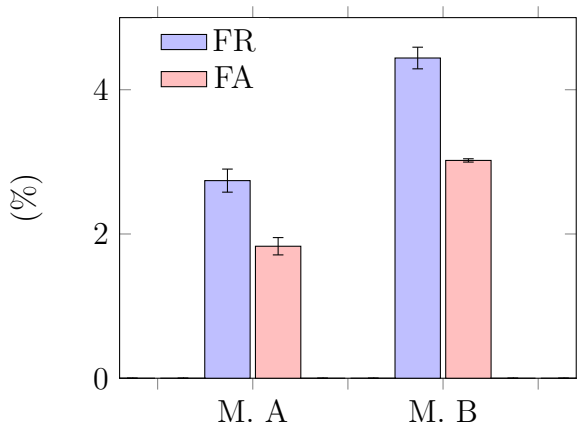


Figure 7.5: Statistical confidence of CPV results in wireless networks. M. A means using the model of Carvalho *et al.* [29]; M. B means using the model of Raptis *et al.* [128].

Note that ideally, the statistical confidence of any results reported thereafter could be measured, although it is not planned in any of the experiments conducted in the remains of this thesis.

Figure 7.6 shows the FRs and FAs when $k = 2$ and $k = 10$. Again, the number of CPV iterations was fixed at $n_{\triangle} = 600$ for all $\triangle$. Using the model of Carvalho and Garcia-Luna-Aceves [29], there was degradation in CPV's efficacy with an increased k, but such degradation was not severe. For example, when all 49 legitimate clients were using a wireless access network (i.e., at $x = 1$ in Fig. 7.6), the sum FR+FA went from ∼4.61% at $k = 2$ to ∼6.22% at $k = 10$. We believe these results stem from the non-zero probability that the wireless delay is (relatively) negligible, e.g., 3 ms. At $k = 10$, the truncated Gaussian distribution in Fig. 7.1 indicates that there is a ∼20% chance the transmitted frame (holding the verifiers' timestamps) suffers <3 ms delay, i.e., if one iteration was performed. As more iterations are performed, the chances that one or more iterations result in such negligible delay increase. Because CPV requires only a proportion τ of the performed iterations to pass the triangular area checks (which is more likely to happen with smaller delays between the verifiers and the client, as discussed in Chapters 5 and 6), it still accepts a client when a proportion of $1 - \tau$ of all iterations result in large delays and area mismatch. The required number of iterations is derived in terms of k and the acceptance threshold τ in Section 7.4 below.

Using the model of Raptis *et al.* [128], and assuming that half the legitimate clients are wireless, the sum FR+FA went from 5.1% at $k = 2$ to 8.3% at $k = 10$. Those numbers are to be compared to 3.1% (2.0% + 1.1%) when none of the legitimate clients are using a wireless access network. In conclusion, under this model, when

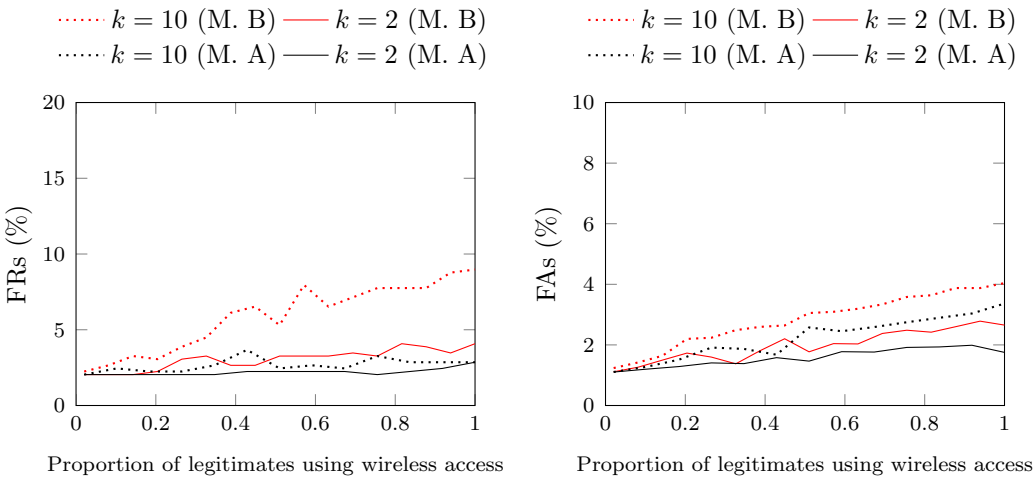


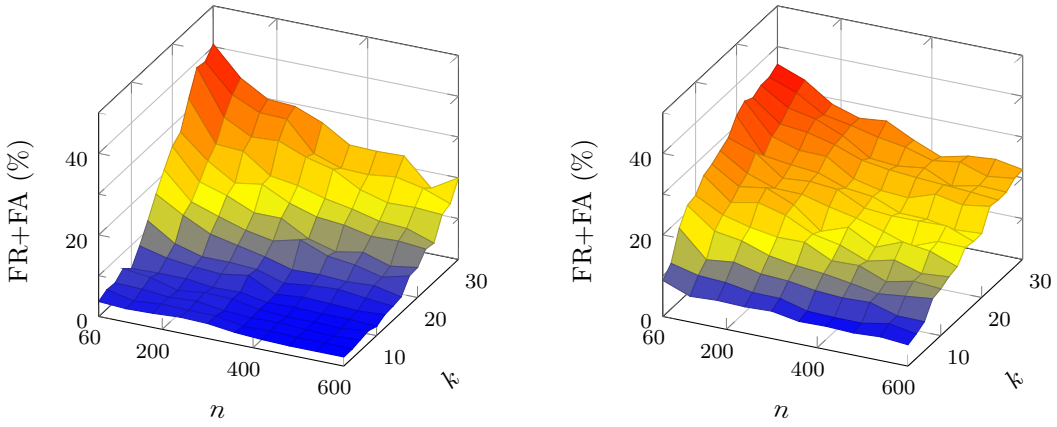
Figure 7.6: FRs and FAs when a proportion of the 49 legitimate clients (i.e., PlanetLab nodes inside triangles) use a wireless access network that has k wireless devices. $n_{\triangle} = 600$ CPV iterations for all $\triangle$. M. A means using the model of Carvalho *et al.* [29]; M. B means using the model of Raptis *et al.* [128].

a wireless CPV legitimate client competes for the media with another device (i.e., $k = 2$), it has double the chances of being falsely rejected compared to a wired legitimate client.

Figure 7.7 shows the summation of FRs and FAs with respect to the number of iterations n (i.e., $n_{\triangle}$ for all $\triangle$), and the number of wireless devices, k, in each wireless network when 25 of the 49 legitimate CPV clients are using a wireless access network.[2] Using the model of Carvalho and Garcia-Luna-Aceves [29], the effect of k on the results begins to manifest starting around $k = 1$. For example, at $k = 2$ the sum FR+FA is almost constant regardless of the performed number of CPV iterations, n. In contrast, at $k = 30$, the impact of n on the sum FR+FA is large. In conclusion, increasing the number of CPV iterations has large impact only when more than $k = 15$ devices are present in each wireless network.

The case is different using the wireless models of Raptis *et al.* [128], where k has a significant impact on the results, for all values of k. For example, at $k = 6$, the sum FR+FA decreases from ∼18% at $n = 60$ to ∼7% at $n = 600$; and at $k = 30$, FR+FA decreases from ∼36% at $n = 60$ to ∼22% at $n = 600$. These results highlight the potential for a larger number of iterations to mitigate the effect of the wireless delays

[2]Recall that the number of wireless legitimate clients being verified by triangle $\triangle$ affects the calibration of $\epsilon_{\triangle}$ and $\tau_{\triangle}$, which is how those 25 wireless clients are expected to influence CPV's decisions on others.



(a) Using Carvalho and Garcia-Luna-Aceves's model [29]

(b) Using Raptis *et al.*'s model [128]

Figure 7.7: FR+FA when half of the evaluated legitimate clients were using a wireless access network with k devices.

on CPV.

Both models agree that CPV's efficacy decreases as k increases, suggesting that CPV may perform poorly in public places where numerous devices are actively competing for the media.

7.3.3 Minimum adversarial distance from the triangle

Figure 7.8 shows the minimum distance, between an (outside-triangle) adversary and the triangle encapsulating the adversary's asserted location, that enables CPV to maintain similar efficacy compared to when all clients are using a wired access network. Recall from Chapter 6, FR+FA when all legitimates were wired-connected is $\sim$3% at $\lambda = 0.1$. Results are obtained when 25 of all 49 legitimate clients are using a wireless access network, and when $n_{\triangle} = 600$ iterations for all $\triangle$.

Using the model of Carvalho and Garcia-Luna-Aceves [29], and at $k = 5$, the sum FR+FA$\approx$3% when (outside-triangle) adversaries were at least $\sim$250 km away from the triangles' sides. At $k = 15$, the minimum distance adversary-free distance outside the triangle that maintains FR+FA$\approx$3% becomes $1,250$ km.

With the model of Raptis *et al.* [128], the minimum adversarial distance is 700 km at $k = 5$ (see Fig. 7.8) and $\sim$1,600 at $k = 10$. In conclusion, the minimum distance clearly increases with k in both models, suggesting that as more saturated devices exist in the network of CPV's legitimate wireless clients, the likelihood of accepting (outside) adversaries close the triangles' sides increases.

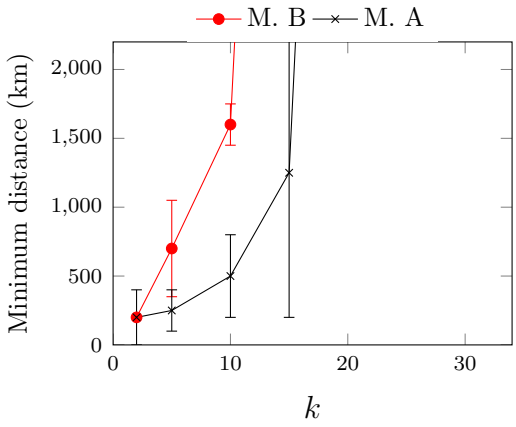


Figure 7.8: The minimum distance, between the (outside) adversary and the triangle, that enables CPV to maintain similar efficacy compared to when all clients are using a wired access network. Results are obtained when 25 of all 49 legitimate clients are using a wireless access network, and when $n_{\triangle} = 600$ CPV iterations, for all $\triangle$. The error bars indicate the smallest and largest y (minimum distance) obtained from 10 runs, and the marker is their average. M. A means using the model of Carvalho *et al.* [29]; M. B means using the model of Raptis *et al.* [128].

7.4 Required Number of CPV Iterations

This section addresses the following question. Assume that the number of wireless devices in the client's access network, k, is known to the verifiers; how many CPV iterations (see Chapter 5) should they perform such that with very high probability the legitimate client gets accepted? It is important to answer this question because, as the results of the previous section show, increasing the number of CPV iterations reduces the impact of the wireless delays on the efficacy of CPV. It is thus important to know what the appropriate number should be in order to mitigate such impact.

To answer this question, let t be a small delay value (i.e., due to the wireless access network) that when added to the (Internet) end-to-end delays of a legitimate client that CPV would typically accept, will not cause CPV to falsely reject this client (i.e., due to the increased delay). Using the wireless delay models in Section 7.2, we can obtain the probability $p_k(t) = P_k\{D < t\}$ that a transmitted frame (carrying the verifiers' signed timestamps) experiences less than t ms additional delay while sharing the wireless media with $k - 1$ other actively participating devices.

If two CPV iterations are performed, the probability that the frames experience $< t$

ms delay in one of them (either the first or the second) is:

$$\begin{aligned}\varrho_1(t,k,2) &= p_k(t)\cdot(1-p_k(t)) + (1-p_k(t))\cdot p_k(t)\\ &= 2\cdot p_k(t)\cdot(1-p_k(t))\end{aligned} \tag{7.26}$$

Note that this equation is similar to the (basic) probability of getting a number x once from a dice that is rolled twice, such that $x < 3$ (i.e., the probability of getting either 1 or 2). This probability would be: either getting x from the first roll but not the second, or from the second roll but not the first; the number of dice rolls is analogous to the number of CPV iterations.

For three iterations:

$$\varrho_1(t,k,3) = 3\cdot p_k(t)\cdot(1-p_k(t))^2 \tag{7.27}$$

In general, the probability that a transmitted frame experiences $< t$ ms in exactly one of n iterations is given by:

$$\varrho_1(t,k,n) = n\cdot p_k(t)\cdot(1-p_k(t))^{n-1} \tag{7.28}$$

Considering more than one iteration, the probability ϱ_2 that the transmitted frames (holding the timestamps) experience $< t$ ms in exactly two of n iterations is given by:

$$\varrho_2(t,k,n) = \left(\frac{n(n-1)}{2}\right)\cdot p_k(t)^2\cdot(1-p_k(t))^{n-2} \tag{7.29}$$

That is because there are $n(n-1)/2$ ways of choosing two of n iterations. In general, there are ${}^nC_r$ ways of choosing r of n iterations, where:

$${}^nC_r = \frac{n!}{r!(n-r)!} \tag{7.30}$$

Accordingly, the probability that the transmitted frames experience $< t$ ms in exactly r of n iterations is given by:

$$\varrho_r(t,k,n) = {}^nC_r\cdot p_k(t)^r\cdot(1-p_k(t))^{n-r} \tag{7.31}$$

And thus, the probability that the wireless delay is $< t$ ms in at least r of n iterations is given by:

$$\rho_r(t,k,n) = \sum_{i=r}^{n}\varrho_i(t,k,n) \tag{7.32}$$

Table 7.5: The probability $p_k(3)$ that an additional delay of < 3 ms is incurred by the wireless network at different values of k.

	Model	k					
		2	5	10	20	25	30
$p_k(3)$	[29]	0.77	0.45	0.21	0.07	0.04	0.03
	[128]	0.24	0.08	0.04	0.02	0.02	0.02

Calculating this probability is fundamental to the operation of CPV. For example, let the number of iterations that CPV performs be $n = 600$, and let CPV be calibrated such that it requires at least 30 of those 600 iterations to pass the triangular area check (explained in Chapter 5). Assuming that $t = 3$, then using (7.32) we can calculate the probability, $\rho_{30}(3, k, 600)$, that the timestamps exchanged between the verifiers and the client are delayed (additionally by the wireless access network) < 3 ms in at least 30 of the 600 iterations. This probability will thus serve as an upper bound probability of that client being correctly accepted. It is "upper bound" because if $\rho_{30}(3, k, 600) = 1$, the client may still get falsely rejected due to other non-wireless related factors (see Chapter 6). Equation (7.32) is used below to derive a function calculating the number of CPV iterations required to mitigate the negative effect of wireless delays.

Note that $p_k(t)$ is calculated using the CDFs in (7.12) and (7.17). For example, for the model of Carvalho *et al.*, we have:

$$p_k(t) = \text{GausCDF}_{\mu,\sigma}(t; 0, \infty) \tag{7.33}$$

where μ and σ are functions of k as discussed in Section 7.2. Example values for $p_k(3)$ are listed in Table 7.5 for various values of k.

Figure 7.9 shows a plot of $\rho_5(3, k, n)$ and $\rho_{20}(3, k, n)$ against n at $k = 2$ and $k = 10$. The charts show that at $k = 2$, the verifiers need to perform 11 (or 45) iterations using the model of Carvalho and Garcia-Luna-Aceves [29] (or that of Raptis *et al.* [128]) to be almost certain (i.e., with probability $\rho_5(3, 2, n) \geq 0.99$) that the transmitted frames will endure < 3 ms delay in at least 5 iterations. To achieve < 3 ms wireless delay in 20 or more iterations, and at $k = 10$, the verifiers will need to perform $\sim$150 and $\sim$700 iterations respectively using the models of Carvalho *et al.* and Raptis *et al.* to satisfy $\rho_{20}(3, 10, n) \geq 0.99$.

CPV requires a proportion $0 \leq \tau_{\triangle} \leq 1$, for each $\triangle$, to pass the triangular area-

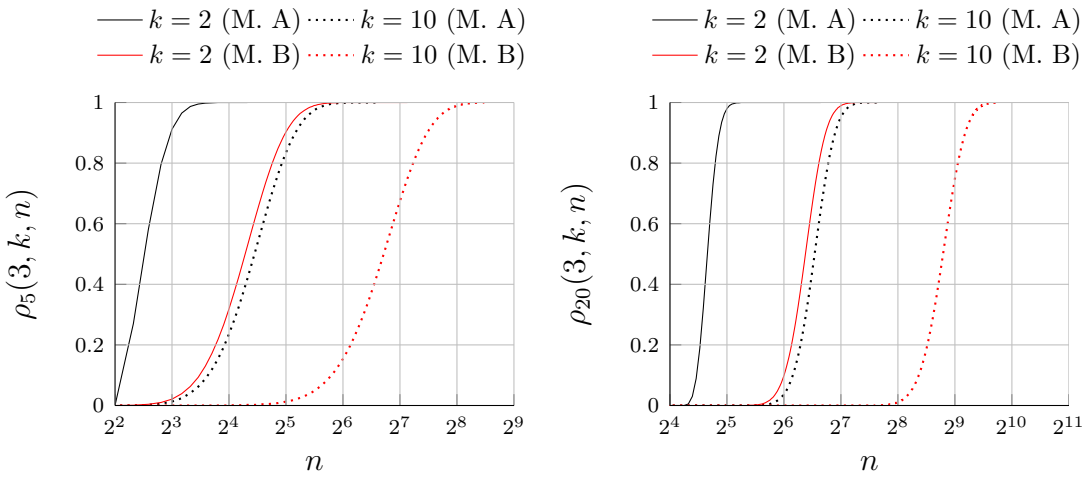


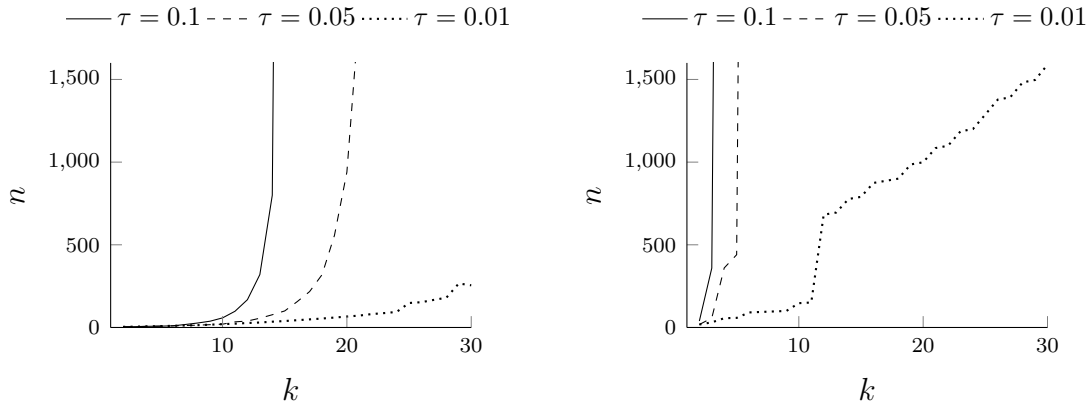
Figure 7.9: The probability that a transmitted frame experiences $< t = 3$ ms of wireless delay in at least 5 and 20 of n iterations, when k wireless devices are sharing the access network. See Table 7.5 (or similarly Figures 7.1 and 7.2 at $x = 3$ ms) for the values of $p_k(t)$. M. A means using the model of Carvalho *et al.* [29]; M. B means using the model of Raptis *et al.* [128].

match checking in order to accept a client.[3] By policy, if $n \cdot \tau_{\triangle}$ of the n iterations pass the area checks, the client gets accepted. To mitigate the effect (on CPV's decisions) of wireless delays with probability $\geq$0.99, the verifiers need to perform n iterations that satisfy:

$$\rho_{n\tau_{\triangle}}(t, k, n) \geq 0.99 \tag{7.34}$$

Using linear iterative root finding [141], we solved (7.34) for n at various values of k. A plot of both variables is shown in Fig. 7.10 for different values of τ. Once again, the differences between the wireless delay models in the reviewed literature manifest in our analysis. For example, using the model of Carvalho and Garcia-Luna-Aceves [29], if $\tau = 0.05$, then only 8 iterations are required to mitigate the effect of the wireless delays on CPV, versus 440 iterations using the model of Raptis *et al.* [128]. At $k = 30$ wireless devices, and $\tau = 0.01$, the required number of iterations is $\sim$250 and $\sim$1590 respectively.

[3]Recall that the CPV algorithm handles triangular inequality violations (TIVs) and area-mismatches similarly, both are treated as area-mismatch.



(a) Using Carvalho and Garcia-Luna-Aceves's model [29]

(b) Using Raptis *et al.*'s model [128]

Figure 7.10: Required number of iterations to essentially eliminate the effect of wireless network delays at different values of τ.

7.5 Conclusion

In this chapter, the efficacy of CPV was evaluated when legitimate clients are using wireless access networks with varying k—the number of saturated wireless devices competing for the media in each network. Evaluation was performed using wireless delay distributions derived in the literature.

The results show that wireless networks are likely to impact the correctness of CPV's decisions. The significance of that impact depends fundamentally on k. For example, the summation of FRs and FAs jumped from 3% (see Chapter 6) when all legitimate clients are using wired access networks to ∼4.5% at $k = 2$ and to ∼7% at $k = 10$ (those numbers are the averages obtained upon using both of the reviewed models).

Moreover, we found that CPV is more likely to falsely accept adversaries close to the triangles' sides when there are wireless legitimate (inside) clients. For example, when $k = 10$, some adversaries within ∼1,000 km of the triangles' sides were falsely accepted, some of which were correctly rejected when no wireless legitimate clients were considered (Chapter 6). Adversaries that are farther away than this distance are unlikely to (illicitly) benefit from the existence of wireless legitimate clients.

Finally, the analysis conducted in this chapter shows that increasing the number of CPV iterations can mitigate the negative effect of wireless delays on CPV. As such, we derived the number of iterations required to achieve that mitigation. Using the derived expressions, we found that the required number of iterations rapidly increases with k. When CPV is calibrated to be more tolerant to high delays between

the client and the verifiers (i.e., at smaller values of τ), the rate for which the required number of iterations increases with k slows down. These results highlight the importance of conducting the appropriate number of iterations, especially when CPV is verifying locations of wireless legitimate clients.

In general, the results in this chapter suggest that the impact of wireless networks on delay-based Internet applications should be given more attention, e.g., most delay-based geolocation techniques in the literature are not evaluated with wireless networks. Investigating the behavior of CPV with clients using other wireless access technologies (e.g., LTE networks) is left for future investigation.

Chapter 8

Hindering Middleboxes from Unauthorized Traffic Relaying

When employed by online content providers, access-control policies can be evaded whenever clients collude with a Middle Box (MB) that meets the policies. A colluding MB, commonly being the gateway of a VPN, typically contacts the content provider on behalf of the clients it colludes with, and relays the provider's outbound traffic to those clients.

To address this problem, we propose a solution to hinder colluding MBs from unauthorized relaying of traffic to a large number of clients. To the best of our knowledge, this is the first work to address this problem. Our solution increases the cost of collusion by leveraging client puzzles in a novel way, and uses network properties to help the content provider detect if its outbound traffic is being further relayed beyond a transport-layer connection. Our evaluation shows that using client puzzles places an upper bound on the number of clients a MB can collude with in parallel. The upper bound follows a hyperbolic decay with the rate of creation of puzzles and the time required to solve a puzzle—both factors are influenced by the content provider, but grows almost linearly with the MB's computational resources.

8.1 Introduction

The content in this chapter was published at the IEEE Communications Letters [12].

Online content providers, such as Hulu [78], often have access-control policies, which either customize or prevent content-delivery to certain classes of clients. By *client*, we mean the software used to communicate with the content provider, e.g., a web browser. For instance, an access policy may only allow access to clients within 300 km of where the site is hosted (e.g, for data sovereignty [118]), or to those with certain IP addresses [40]. Another policy may ban clients at a specific geographic location [18, 25] (see Chapters 3 and 5), or clients whose devices have certain system fingerprints (operating system, user-agent, etc) [112]. A content provider (or *provider* for short) may also classify clients by their access networks [151], or their network distance from the server (in terms of hop counts, network latency, etc) [81].

When access policies are in effect, the motivation to bypass them may arise. A client that does not meet the access policies may try to bypass them using a MB that meets those policies. MBs are commonly transport-layer proxy servers, gateways of VPNs or anonymizing networks. The MB requests the provider's content and grants the client access to it by simply relaying the provider's outbound traffic. Many MBs claim to own thousands of IP addresses, which makes blocking them by enumerating their IP addresses almost infeasible. To detect an intercepting MB, a provider can collaborate with a *cooperative* client [39]. However, this is infeasible within our threat model as we address a client that aims to bypass the provider's access policies; i.e., the client is the provider's adversary. Solutions that aim to prevent MBs from intercepting a connection (such as Secure Socket Layer [130]) fail to prevent those MBs from relaying traffic because the client would be ready to share cryptographic credentials, such as encryption keys, with the MB to deceive the provider.

We propose to use client puzzles [83] to increase the cost of collusion per client on the MB. Our solution leverages network properties (average latency between network hosts) which, together with the puzzles, impose a *limit* on the number of simultaneous clients an MB can collude with. Exceeding the limit divulges the MB's relaying actions to the provider. This chapter makes the following contributions:

- Proposing and studying a solution that uses client puzzles to limit unauthorized traffic relaying (Section 8.2).
- Using a Markovian queueing model to evaluate our solution, and to find the upper limit of the number of clients the MB can collude with at a time (Section 8.3).
- Evaluating the rates of false rejects and false accepts through simulations.

8.2 Proposed Approach

Our objective is to enable a provider detect if a *content recipient*[1] is a *legitimate client* (i.e., connected to the provider without an MB and not relaying the provider's traffic anywhere else) or an MB. To achieve this objective, we use client puzzles [83] to increase the computation required by the MB per client; thus, increasing the RTT the provider observes. The success of detecting an MB is dependent on the number of simultaneous clients receiving the relayed traffic from the MB. As the number increases, the detection success increases. If the number of clients reaches a certain threshold (Section 8.3), the provider realizes that the MB is relaying its traffic. The provider is assumed to be able to:

- Estimate the average RTT from itself to a content recipient [92, 153].
- Estimate the mean time to solve a puzzle with certain difficulty across different client machines spanning a range of computational power (demonstrated in [83]).

For each connection made to provider w from content recipient d, w estimates $N_w(d)$, which is the average network RTT from itself to d. The provider w then periodically creates non-parallelizable puzzles [144], and sends them to d. To solve a puzzle, d must allocate a portion of its resources for some time depending on the puzzle difficulty set by w. The resource demanded by the puzzle depends on the type chosen by w, which could be processing- [83] or memory-type [46] puzzles. We assume w uses processing-type puzzles throughout this chapter. However, any type can be chosen as long as w is able to estimate the client's puzzle-solving time to some degree of certainty (second assumption above). Upon solving a puzzle, d is required to return the solution to w, which verifies it and bans d if the solution was incorrect. Verification happens in constant time independent of the puzzle difficulty [83].

Denoting t_c as the mean time to solve a puzzle across various clients, w expects to see a RTT of:

$$\mathrm{RTT}_e = N_w(d) + t_c \tag{8.1}$$

When w receives a solution, it calculates the actual round-trip time, RTT_a, from the puzzle-arrival time and compares it with RTT_e. If $\mathrm{RTT}_a \leq \mathrm{RTT}_e$, the provider assumes that d is not an MB. Otherwise, it suspects that d is an MB because the existence of an MB between the provider and a client is likely to increase RTT_a—an

[1] We use this term to refer to the machine intended by the provider as the final content destination.

explanation follows.

If d is an MB, it has two options: either relaying all of w's outbound traffic including the puzzles to client c, so that c solves them; or extracting the puzzles from the traffic and solving them on behalf of c. Relaying the puzzles to c costs an additional network RTT, $N_{\text{MB}}(c)$, between the MB and c. An analogous effect occurs if the puzzles were outsourced to a remote party. The actual RTT then becomes:

$$\text{RTT}_a = N_w(d) + N_{\text{MB}}(c) + t_c \tag{8.2}$$

We do not expect w to be able to estimate $N_{\text{MB}}(c)$. To satisfy $\text{RTT}_a \leq \text{RTT}_e$, the MB and c have to satisfy $N_{\text{MB}}(c) + t_c \leq t_c$, which happens when $N_{\text{MB}}(c) = 0$; that is, the colluding client and the MB are one physical machine, or very close to each other. We believe it is not a cost effective (scalable) attack for an MB to be close to a meaningful number of clients. Assuming proper estimations to t_c and $N_w(d)$ (i.e., RTT_e), it would be challenging for the MB to relay the puzzles to c, and satisfy $\text{RTT}_a \leq \text{RTT}_e$. We study the effect of inappropriate estimation of RTT_e in Section 8.3.1 below.

To avoid the additional $N_{\text{MB}}(c)$, the MB will be inclined to choose the second option: solve the puzzles on behalf of the clients. An additional queueing time, q, is expected to contribute to RTT_a because the MB will solve many puzzles, which correspond to the number of clients it simultaneously colludes with. The actual RTT would then be:

$$\text{RTT}_a = N_w(d) + q + t_{\text{MB}} \tag{8.3}$$

where t_{MB} is the MB puzzle-solving time. Recall, the content recipient d is the MB. Again, we do not expect w to be able to estimate t_{MB}. To maintain $\text{RTT}_a \leq \text{RTT}_e$, the MB's computational resources must satisfy:

$$W \leq t_c \tag{8.4}$$

where $W = q + t_{\text{MB}}$, which is the average time a puzzle spends at the MB from the moment it arrives unsolved to the MB until it departs the MB solved. The queueing time q is affected by: the rate at which w sends puzzles to each client connection; the number of clients simultaneously colluding with the MB; the MB's processing capabilities; and the puzzles' difficulty. The last two factors also affect t_{MB}. Although this option seems more appealing to the MB than the previous one, it forces the MB to limit the number of simultaneous clients to avoid being caught by the provider.

Table 8.1: Notation

Notation	Description
δ	the number of clients simultaneously *colluding with* (i.e., being relayed the provider's content from) the MB.
t	(t_c in Section 8.2) the mean of an exponential distribution representing the time required to solve a single puzzle across different client machines, measured in seconds/puzzle. The provider is required to estimate this mean according to the chosen puzzle difficulty.
r	the rate the provider generates puzzles to each client connection, measured in puzzles/second.
b	the proportion of a client's time available to solve puzzles;[2] $b = rt$. If $b = 1$, the average client spends all of its time solving puzzles.
k	the number of distinct puzzles the MB can solve simultaneously. It is possibly influenced by the number of available processing cores to the MB.
g	the factor by which an MB processing core is faster than the average client. It is possibly influenced by the cores' clock rate.

If an MB chooses to combine both options, solving some puzzles by itself and relaying others, the provider will likely observe larger RTT for the relayed puzzles and hence reject the client. The provider may allow some proportion, ρ, of RTTs to be larger than the expected RTT before rejecting a client to account for delay spikes. In such case, the benefit of relaying some puzzles will be limited by the provider's parametrization, which upper bounds the proportion of puzzles the MB can relay, without getting its clients rejected, by ρ.

8.3 Evaluation and Analysis

In this section, we derive W (Section 8.2) as a function of the parameters affecting it. We choose an analytical evaluation method rather than an empirical one to calculate the theoretical maximum number of clients a MB can simultaneously collude with (i.e., relay content to) to maintain W that satisfies equation (8.4).

We use the notation in Table 8.1. Note that of all the variables in the table, a provider needs only estimate k and g, which is left for future investigation.

We focus only on the MB's processing power (k and g) as needed to solve processing-type puzzles, and exclude from consideration resources (e.g., bandwidth, I/O, memory, etc) needed for the MB to relay content to clients. The motivation for this is to allow focus on how the puzzle rate and difficulty constrain the MB; i.e., this is the limiting factor. It follows that if the MB has sufficient resources to solve the puzzles sent to it, then we assume it will have sufficient additional resources to relay content to an arbitrary number of clients. We assume the MB does not store a local copy of the traffic it receives from the provider; it initiates a connection to the provider with each client connection request.

We use the $M/M/k$ queueing model [65] to represent the queueing system at the MB, where we assume the puzzle arrival is modelled by a Poisson process, and the puzzle-solving time is exponentially distributed. This model considers k serving units, which in our case is the number of puzzles the MB is able to solve in parallel. The waiting time of this model is [65]:

$$W = \frac{1}{\mu} + \left(\frac{(k\rho)^k}{k!(1-\rho)} + \sum_{i=0}^{k-1} \frac{(k\rho)^i}{i!} \right)^{-1} \left(\frac{\rho(k\rho)^k}{\lambda(1-\rho)^2 k!} \right) \tag{8.5}$$

where

$$\rho = \frac{\lambda}{k\mu} \tag{8.6}$$

In the queueing terminology, λ is the customer arrival rate to the system and μ is the customer departure rate from each of the k serving units ($\mu = \frac{1}{\text{service time/customer}}$), both measured in customers/time unit. Customers arriving and departing the system resemble, in our case, unsolved puzzles arriving and solved puzzles departing the MB. Customer-service time at each serving unit resembles puzzle-solving time at each of the MB's cores.

To realize the maximum δ that satisfies (8.4), we first need to represent W as a function of δ. We use the waiting time of (8.5), and express λ and μ in terms of δ, r, t and g. Because the provider sends puzzles at a rate of r puzzles/second to each client connection, the puzzle arrival rate at the MB is $\lambda = nr$ puzzles/second. The rate of solving puzzles at each of the k cores is g times faster than that of a client; hence, $\mu = g/t$. Substituting in (8.6), we get:

$$\rho = \frac{nrt}{kg} = \frac{nb}{kg} \tag{8.7}$$

Note that the MB can prevent its queue from growing indefinitely by maintaining $\lambda < k\mu$ [65], which occurs if it keeps the number of simultaneous clients $\delta < kg/b$.

However, only satisfying this inequality can still disclose the MB's relaying actions to the provider, as it does not ensure satisfying (8.4). By substituting ρ obtained as in (8.7) for that in (8.5), we express W in terms of δ, t, r, k and g. Inequality (8.4) (which can be rewritten as $W/t - 1 \leq 0$) then becomes:

$$\frac{1}{g} + \left(\frac{(\frac{nb}{g})^k}{k!(1-\frac{nb}{kg})} + \sum_{i=0}^{k-1} \frac{(\frac{nb}{g})^i}{i!} \right)^{-1} \left(\frac{\frac{nb}{kg}(\frac{nb}{g})^k}{nb(1-\frac{nb}{kg})^2 k!} \right) - 1 \leq 0 \tag{8.8}$$

Using linear iterative root finding [141], we can find the maximum integer value of δ that satisfies (8.8).

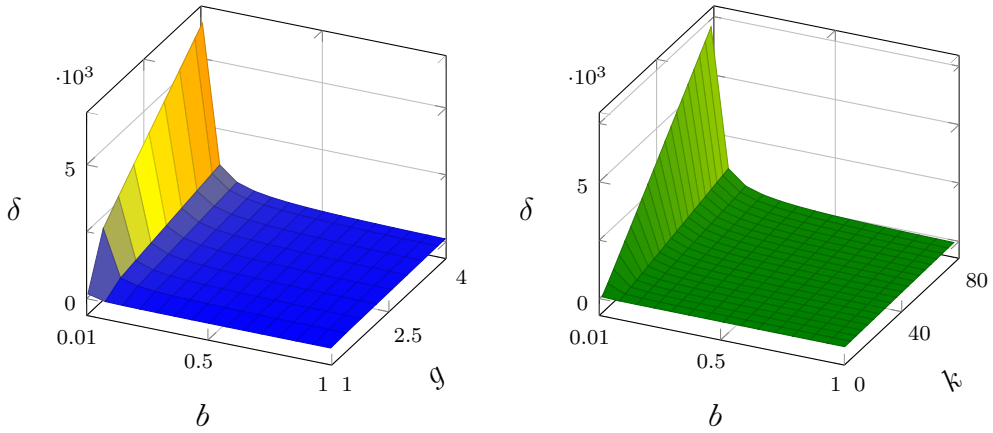
To study the behavior of δ with respect to b, k and g, we consider a range of values for each of those parameters in the intervals $[2^{-6}, 1]$, $[1, 80]$ and $[1, 4]$ respectively. Note that, as of this writing, the fastest clock frequency being manufactured in the industry is the IBM zEC12, which has a frequency of 5.5 GHz [42]. On the other hand, processor speeds of smartphones (i.e., representing slow clients) are generally in the range of 1.2 to 1.9 GHz. As such, the processor speed of a MB is unlikely to exceed four times that of a regular client, hence the upper bound of the selected interval of g. Note that it is still possible for a client to be using a machine slower than 1.2 GHz.[3] As such, values of $g > 4$ could still be of interest to evaluate. However, as we show later below, the selected range does not affect the conclusions drawn about the effect of the puzzles to hinder traffic relaying.

Figure 8.1(a) shows the change of δ at $k = 25$, and Fig. 8.1(b) at $g = 1.5$. We ignore δ when $b > 1$ because the provider should never set b in that range. Otherwise, unsolved puzzles start to accumulate at legitimate clients, increasing the RTT due to additional queueing delay, and falsely rejecting these clients.

From (8.8), we can see that δ and b always occur multiplied together, hence by replacing all occurrences of δb with γ, we can express δ in terms of γ and b as $\delta = \gamma/b$. That is, *δ follows a hyperbolic decay with b (for all $b > 0$) with a scale factor of γ*. The maximum value of γ that makes W satisfy (8.4) grows with k and g. For example, in Fig. 8.1(b)—where $g = 1.5$—every integer value, κ, on the k axis defines the scale factor, $\gamma = f(1.5, \kappa)$, of a hyperbolic decay of δ with respect to b at κ.

The results plotted in Fig. 8.1 show that δ follows an almost linear growth with g

[3]Note—it is likely that such slower devices will already be precluded from enjoying many now-common services that require more powerful processors (e.g., streaming media content, running a browser that is new enough to support web-sockets for CPV, etc).



(a) At $k = 25$.

(b) At $g = 1.5$.

Figure 8.1: Maximum theoretical number of clients that can simultaneously collude with the MB without being detected by the provider. The lines on the surfaces are equally spaced on the b, g and k axes. See Table 8.1 for notation.

and k, versus a hyperbolic decay with b. The provider influences b through t and r, the MB controls k and influences g by investing in hardware. This puts the MB in a critical situation as the provider has a more significant impact on δ than the MB has. These results illustrate the potential of puzzles in limiting the number of colluding clients.

8.3.1 Simulation Results

The analytical evaluation showed how client puzzles affect the number of clients the MB could support in case the MB decides to solve the puzzles on behalf of the clients it colludes with. We now study the case where the MB decides to forward the puzzles to those colluding clients. We use the network simulator (ns-2) [24] to evaluate the rate of False Rejects (FRs), where a legitimate client is rejected by the provider; and False Accepts (FAs), where a client colluding with the MB is accepted. Because wireless access networks have unique latency-estimation issues (see Chapter 7), they are beyond the scope of the evaluation performed in this chapter.

We assume the provider will endure some error while estimating RTT_e in (8.1). This error scales RTT_e by a factor β, such that:

$$\text{RTT}_e = \beta \times \text{RTT}_a \tag{8.9}$$

See (8.2) for RTT_a. FRs tend to increase when $\beta < 1$, FAs tend to increase when

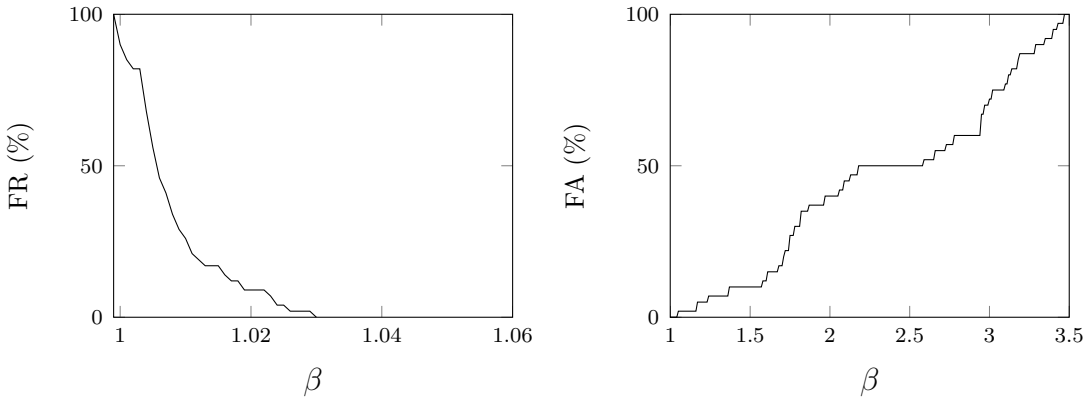


Figure 8.2: FR and FA obtained from simulations; β represents the error of the provider's RTT estimation.

$\beta > 1$.

Our simulation scenarios involved several runs with 100 nodes and random connectivity patterns. Nodes distribution and link latencies were designed to resemble networks distributed over a large geographic region. One node was set to be the provider, another was set to be the MB, while other nodes simulated clients. Some clients were connected directly to the provider (legitimate clients), others (colluding clients) were connected through the MB. FRs and FAs are shown in Fig. 8.2. For the runs we conducted, the error scale in the range $1.03 < \beta < 1.1$ yields 0% FRs and 2% FAs. We believe these results show promising potential for the solution we propose herein.

8.4 Further Considerations

How many puzzles per second should the provider send to a client, and what should their difficulty be? Figure 8.1 showed a tradeoff between allowing more clients to collude with an MB, and overwhelming legitimate clients. To deal with this tradeoff, providers may set b to the value that satisfies a central tendency of δ, such as the mean $\bar{\delta}$, over desired intervals of b, k and g.

One way to calculate $\bar{\delta}$ is to, first, approximate a function that mimics the behavior of δ. This can be done using curve fitting [119]. For example, at $g = 1.5$ and $2^{-6} \leq b \leq 1$, δ_f can mimic the behavior of δ, such that:

$$n_f = k(Ae^{bB+C} + D) \tag{8.10}$$

where A, B, C and D are constants—their values are shown on Fig. 8.3. The mean, $\bar{\delta}_f$, in terms of k is:

$$\bar{\delta}_f = \frac{1}{1-2^{-6}} \int_{2^{-6}}^{1} k(Ae^{bB+C} + D)\,\mathrm{d}b = 8.9k \tag{8.11}$$

Substituting $\bar{\delta}_f$ for δ_f in (8.10), and solving for b, we get:

$$b = \frac{1}{B}\left(\ln\frac{8.9k - Dk}{kA} - C\right) = 0.07 \tag{8.12}$$

That is, considering the abstraction given in Section 8.3 and our queueing model, when b is restricted to the range $2^{-6} \leq b \leq 1$ and $g = 1.5$, the mean of δ_f occurs at $b = 0.07$. Beyond this value of b, puzzles will overwhelm legitimate clients without significant drop in the number of colluding clients δ whereas below, δ rapidly increases with little reduction in the puzzle workload on legitimate clients. This highlights selection of an example value of b which may be of practical interest.

To set b, the provider adjusts r and t such that their product b results in the desired value. Because the network RTT is typically measured in ms [34], a puzzle that takes a relatively long time (e.g., 1 sec) to solve on an average client machine may overshadow the network RTT. Providers need to consider that when setting the puzzle difficulty, as it affects t.

Finally, providers may consider varying the puzzles' difficulty randomly, and discarding the observed RTT of puzzles that are harder than certain undisclosed threshold to avoid having their solving time overshadow the network RTT. This may penalize an MB significantly as it will not be able to distinguish *time-sensitive* puzzles (those where the provider will account for their RTT) from others, and will have to solve them in order of arrival. Having a number of relatively difficult puzzles in the MB's queue will raise the waiting time of all others behind them, making it easier for the provider to capture the highly-delayed responses of timed puzzles, thus, detecting the MB.

8.5 Conclusion

This chapter addressed the problem of unauthorized relaying of a content provider's traffic, commonly performed by a MB to enable colluding clients to bypass access-control policies set by the provider. We proposed to use client puzzles and delay estimation to enable providers hinder such unauthorized relaying of traffic, and used

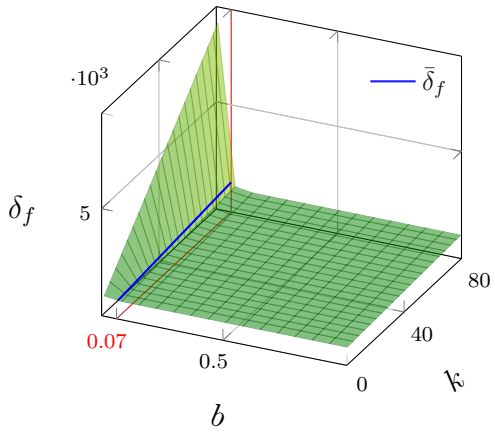


Figure 8.3: Fitted surface at $g = 1.5$ represented by (8.10). The values of the constants in equation (8.10) are: $A = 5.64$, $B = -58.13$, $C = 3.9$ and $D = 4.37$. NRMSD over the displayed b and k intervals is 0.04 (or 4%). The blue line represents the mean $\bar{\delta}_f$.

a queueing-model to evaluate our solution.

The evaluation shows that in the presence of the proposed solution, the maximum number of clients that can simultaneously collude with the MB without being detected by the provider follows a hyperbolic decay with the rate of creation of puzzles and the time required to solve them. Both of these factors are influenced by the content provider. Additionally, the number of colluding clients follows an almost-linear growth with the MB's computational resources, which is rather influenced by the MB. This enables a provider using the proposed solution to have a higher control on the maximum number of simultaneously colluding clients than the MB itself. However the rate of puzzle creation overwhelms legitimate clients too, deriving the need to find an appropriate balance between overwhelming legitimate clients and limiting the MB's collusion. We discussed how this balance could be obtained given the situation.

Chapter 9

Conclusion

As location-oriented service/content providers are emerging over the Internet, verifying the geographic locations of Internet clients is becoming increasingly crucial. A plethora of security applications—such as fraud detection, location-based authentication, and online voting—can benefit from a realtime location-verification tool.

Measurement-based Internet geolocation approaches highlight a strong correlation between the Internet's delays and geographic distances, and provide a strong evidence of the ability to utilize these delays to locate clients, given appropriate delay processing. Despite the achieved accuracy of recent techniques, the process of refining the measured delays could be exploited by an *adversary* motivated to forge its location. Accordingly, any *secure* delay-based geolocation approach has to consider both *menaces*: the adversary and the Internet-added delay uncertainty.

9.1 Satisfying Thesis Objectives

In this thesis, we first investigate the reliability of current state-of-the-art delay-based geolocation techniques in the presence of an adversarial client motivated to deliberately misrepresent its own geographic location (Chapter 3). Our findings illustrate that such techniques are not ready for use in hostile environments yet as they fail to employ an integrity-preserving delay-measurement process, which is the fundamental component relied upon by all such techniques. The difficulty to fix current status quo stems from the challenges of getting community support to modify the default implementation of ICMP-based utilities in the network stack, and disseminate the modifications for the sole purpose of hardening geolocation.

We then proceed to devise CPV (Chapter 5), a delay-based algorithm designed to provide a higher level of assurance about the correctness of a device's location, compared to the assurance provided by current state-of-the art geolocation techniques. To reduce potential false rejects/accepts, we support CPV by a novel OWD-estimation protocol that requires similar amount of client cooperation as in estimating RTTs, yet achieves higher accuracy in many cases (Chapter 4). To identify these cases, we derived the probability distribution of absolute error for both protocols as a function of the underlying delay distribution. CPV has been extensively evaluated in wired (Chapter 6) and wireless networks (Chapter 7), and the results show its potential to be adopted in practice.

We show how the CPV algorithm can be further reinforced against a customized MB, which is specifically designed to defeat CPV by exchanging the algorithm's control messages with the verifiers on behalf of the adversary (Chapter 8). By attaching a cryptographic puzzle to these control messages and verifying the solution each time the messages are echoed, we force the MB to solve all the puzzles destined to all the adversaries it colludes with. We proved how this technique enables CPV to place a ceiling on the number of adversaries the MB can collude with in parallel, without being detected by CPV.

The bigger picture

Table 9.1 shows solutions designed to ensure the integrity of location calculation against common adversarial threats. The table shows where the location verification mechanisms contributed by this thesis (two right-most columns) stand with respect to current state-of-the-art mechanisms. A check mark (✓) means the solution is sufficient to ensure location integrity against the respective adversarial threat. Note that the threat at row i means the adversary is capable of imposing this threat and all previous threats in upper table rows. Accordingly, a check mark at row i means the respective solution (column) is sufficient to ensure location integrity against an adversary capable of imposing all threats from 1 to i inclusive, or any combination thereof.

The table categorizes these solutions by their susceptibility to evasion rather than, e.g., by their cost of operation or the magnitude of their accuracy. The "user-declared location" (column 1) is the mechanism by which the LSP simply asks the human user about his/her location. In the absence of any adversarial threats, including the absence of the threat that the user falsifies (or lies about) their location

Table 9.1: Solutions designed to ensure the integrity of location calculation against common adversarial threats. The location verification mechanisms contributed by this thesis are in the two right-most columns.

		Solutions					
	Adversarial threat	User-declared location	Client self-geolocation	Inference-based or measure.-based IP geoloc.	App-layer measurement-based geoloc.*	CPV	CPV + Proof-of-Work
1	*Absence of threats*	✓	✓	✓	✓	✓	✓
2	Falsifying declared location		✓	✓	✓	✓	✓
3	Forging transmitted coordinates			✓	✓	✓	✓
4	Modifying location hints				✓	✓	✓
5	Manipulating delays					✓	✓
6	Colluding with a public MB						✓

*This class of solutions encompasses all measurement-based geolocation techniques when measurements are performed on the application layer of the TCP/IP protocol stack.

(row 2 in the table), this mechanism is sufficient to ensure location integrity (hence the checkmark in row 1 column 1).

As discussed in Chapter 2, the "client self-geolocation" category (column 2 in the table) includes any means by which the client geolocates itself and informs the Location-Sensitive Provider (LSP), e.g., using GPS or WPS. Also recall, from Section 2.1.4 on page 13, that the client's location could be "inferred" from its IP address (column 3). This is different from measurement-based IP geolocation, which is when the delay-measurement probes are destined to the client's observed IP address. However, we place both, inference-based geolocation (including IP-address based inference) and measurement-based IP geolocation techniques, together under a single solution category (column 3) since they are affected by the same threat: modifying location hints. This threat includes not only modifying browser-based hints (e.g., preferred language—see Section 2.2 on page 13), but also using a MB to modify the IP address observed by the geolocating party. If measurement-based geolocation is to be used, with delay measurements performed over the application layer (column 4), e.g., through the browser [30, 107] or using websockets [50, 95], it would be sufficient to ensure the integrity of location calculation against the threats in rows 1 to 4 in the table.

The table shows that CPV combined with a Proof-of-Work (PoW) mechanism, as we explain in Chapter 8, is sufficient to ensure location integrity against all the listed adversarial threats in the table. The absence of CPV leaves two threats (rows 5 and 6) unaddressed by current state-of-the-art geolocation techniques. Note however that this list is not exhaustive. For example, there is also the threat of the adversary colluding with a MB customized to evade CPV, and that MB is not colluding with other adversaries. This is the case when, for example, the adversary has its own private MB physically located where it wants to fraudulently appear to be. However, it may not be scaleable for an adversary to own a MB at every possible geographic location it intends to forge its location to. To that end, we believe the mechanisms for location verification of Internet clients contributed to the literature by this thesis are of practical value to many location-sensitive applications.

9.2 Future Research Directions

We now discuss possible future extensions to the work conducted in this thesis.

Enhancing the accuracy of delay-based geolocation techniques. The advantages provided by the *minimum pairs* protocol of Chapter 4 can be leveraged to enhance the accuracy of delay-based geolocation techniques. The protocol requires three cooperating servers to exchange messages among themselves and the client; in delay-based techniques, sets of three landmarks can cooperate to implement the *minimum pairs* protocol, thus estimating OWDs instead of RTTs across *all* the links between the landmarks and the client. No further cooperation would be required from the client beyond echoing the messages, which is similar to what the client does when the landmarks estimate RTTs. For example, the client would not be required to synchronize its clock with the landmarks, nor to calculate and report its view of the delays.

Server location verification. Verifying the geographic locations of servers, e.g., webservers, may provide security benefits to mitigate server impersonation, typically done through phishing, pharming or Man in the Middle (MitM) attacks. We believe that some of the ideas in this thesis, including the heuristics used to enhance the delay-measurement process, can be adapted to address the problem of verifying the geographic locations of servers.

References

[1] "OS X 10.9.2—Source/Source Browser—ping," http://www.opensource.apple.com/source/network_cmds/network_cmds-433/ping.tproj/ping.c. 23, 24

[2] "OS X 10.9.2—Source/Source Browser—traceroute," http://www.opensource.apple.com/source/network_cmds/network_cmds-433/traceroute.tproj/traceroute.c. 24, 28

[3] "Ubuntu—Details of source package tcptraceroute in lucid," http://packages.ubuntu.com/source/lucid/tcptraceroute. 22

[4] "Hping - Active Network Security Tool," http://www.hping.org/download.php, 2005. 24, 27

[5] "fping 3 Homepage," http://fping.org/dist/fping-3.10.tar.gz, 2014. 27

[6] "Freebsd Source. [base] Index of release/9.3.0/contrib/traceroute," https://svnweb.freebsd.org/base/, 2014. 24, 28

[7] "FreeBSD Source. [base] Index of release/9.3.0/sbin/ping," https://svnweb.freebsd.org/base/, 2014. 23, 24

[8] "GNU Project Archives (/inetutils-1.9.2/src/traceroute.c)," http://ftp.gnu.org/gnu/inetutils/, 2014. 24, 27, 28

[9] A. M. Abdou, A. Matrawy, and P. C. van Oorschot, "CPV: Delay-based Location Verification for the Internet," *IEEE Trans. Dependable and Secure Computing, TDSC (to appear; accepted June 14, 2015.* 45, 62, 77

[10] ——, "Location Verification on the Internet: Towards Enforcing Location-aware Access Policies Over Internet Clients," in *IEEE CNS*, 2014. 42, 45, 77

[11] ——, "Accurate One-Way Delay Estimation with Reduced Client-Trustworthiness," *IEEE Commun. Lett.*, vol. 19, no. 5, pp. 735–738, 2015. 45

[12] ——, "Taxing the Queue: Hindering Middleboxes from Unauthorized Large-Scale Traffic Relaying," *IEEE Commun. Lett.*, vol. 19, pp. 42–45, 2015. 19, 115

[13] M. Arif, S. Karunasekera, and S. Kulkarni, "GeoWeight: Internet Host Geolocation Based on a Probability Model for Latency Measurements," in *Australian Computer Society, Inc. ACSC*, 2010. 11, 19, 63

[14] M. Arif, S. Karunasekera, S. Kulkarni, A. Gunatilaka, and B. Ristic, "Internet Host Geolocation Using Maximum Likelihood Estimation Technique," in *IEEE AINA*, 2010. 11, 19, 43

[15] V. A. Balasubramaniyan, A. Poonawalla, M. Ahamad, M. T. Hunter, and P. Traynor, "PinDr0p: Using Single-ended Audio Features to Determine Call Provenance," in *ACM CCS*, 2010. 8

[16] S. Banerjee, T. Griffin, and M. Pias, "The Interdomain Connectivity of PlanetLab Nodes," in *Springer PAM*, 2004. 79

[17] D. Berbecaru, "LRAP: A Location-Based Remote Client Authentication Protocol for Mobile Environments," in *Euromicro PDP*, 2011. 62

[18] E. Bertino, B. Catania, M. L. Damiani, and P. Perlasca, "GEO-RBAC: a spatially aware RBAC," in *ACM SACMAT*, 2005. 18, 116

[19] G. Bianchi, L. Fratta, and M. Oliveri, "Performance evaluation and enhancement of the CSMA/CA MAC protocol for 802.11 wireless LANs," in *IEEE PIMRC*, 1996. 95

[20] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," *Selected Areas in Communications, IEEE Journal on*, vol. 18, pp. 535–547, 2000. 95, 99, 102

[21] C. Bovy, H. Mertodimedjo, G. Hooghiemstra, H. Uijterwaal, and P. Van Mieghem, "Analysis of end-to-end delay measurements in Internet," in *Springer PAM*, 2002. 58

[22] R. Braden, "Requirements for Internet Hosts - Communication Layers," RFC 1122 (Internet Standard), 1989. 24

[23] S. Brands and D. Chaum, "Distance-Bounding Protocols," in *Advances in Cryptology–EUROCRYPT'93*. Springer, 1994, pp. 344–359. 15

[24] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu, "Advances in network simulation," *Computer*, vol. 33, no. 5, pp. 59–67, 2000. 122

[25] "BBC News - US employee outsourced job to China," http://www.bbc.com/news/technology-21043693, British Broadcasting Corporation, January 2013. 116

[26] J. Burnett, "Geographically Restricted Streaming Content and Evasion of Geolocation: the Applicability of the Copyright Anticircumvention Rules," *HeinOnline MTTLR*, vol. 19, p. 461, 2012. 19, 63

[27] CANARIE, "CANARIE — Advancing Canada's knowledge and innovation infrastructure," http://www.canarie.ca, 2015. 79

[28] S. Capkun and J.-P. Hubaux, "Secure positioning of wireless devices with application to sensor networks," in *IEEE INFOCOM*, 2005. 15, 63

[29] M. Carvalho and J. Garcia-Luna-Aceves, "Delay analysis of IEEE 802.11 in single-hop networks," in *IEEE Network Protocols*, 2003. 91, 94, 95, 96, 97, 98, 99, 100, 101, 102, 104, 105, 106, 107, 108, 109, 111, 112, 113

[30] M. Casado and M. J. Freedman, "Peering Through the Shroud: The Effect of Edge Opacity on IP-based Client Identification," in *USENIX NSDI*, 2007. 14, 19, 34, 63, 72, 75, 128

[31] C. Castelluccia, M. A. Kaafar, P. Manils, and D. Perito, "Geolocalization of Proxied Services and Its Application to Fast-flux Hidden Servers," in *ACM IMC*, 2009. 19, 43

[32] J.-H. Choi and C. Yoo, "One-way delay estimation and its application," *Computer Communications*, vol. 28, pp. 819–828, 2005. 45

[33] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "PlanetLab: An Overlay Testbed for Broad-coverage Services," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 33, pp. 3–12, 2003. 5, 32, 64, 77

[34] M. Crovella and B. Krishnamurthy, *Internet Measurement: Infrastructure, Traffic and Applications*. John Wiley & Sons, 2006. 9, 51, 124

[35] I. Cunha, R. Teixeira, D. Veitch, and C. Diot, "DTRACK: A System to Predict and Track Internet Path Changes," *IEEE/ACM Trans. Netw.*, vol. 22, pp. 1025–1038, 2014. 64

[36] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A decentralized network coordinate system," in *ACM SIGCOMM*, 2004. 42, 65

[37] C. Davis, I. Dickinson, T. Goodwin, and P. Vixie, "A Means for Expressing Location Information in the Domain Name System," RFC 1876 (Experimental), 1996. 12, 42

[38] L. De Vito, S. Rapuano, and L. Tomaciello, "One-Way Delay Measurement: State of the Art," *IEEE Trans. Instrum. Meas.*, vol. 57, pp. 2742–2750, 2008. 51

[39] G. Detal, B. Hesmans, O. Bonaventure, Y. Vanaubel, and B. Donnet, "Revealing middlebox interference with tracebox," in *ACM IMC*, 2013. 14, 116

[40] C. Dietrich and C. Rossow, "Empirical research of IP blacklists," in *ISSE 2008 Securing Electronic Business Processes*, 2009, pp. 163–171. 116

[41] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *USENIX Security*, 2004. 14, 63

[42] I. Dobos, H. Chu, L. Fadel, W. Fries, O. Lascu, F. Nogal, F. Packheiser, E. Palacio, M. Raave, V. R. Jr., A. Spahni, and C. Zhu, "IBM zEnterprise System Technical Introduction," http://www.redbooks.ibm.com/redbooks/pdfs/sg248050.pdf, IBM Redbooks, Tech. Rep., 2014. 121

[43] "IP Address Lookup Hostip.info," http://www.hostip.info/, Domains By Proxy, LLC. 13, 19

[44] Z. Dong, R. D. Perera, R. Chandramouli, and K. Subbalakshmi, "Network measurement based modeling and optimization for IP geolocation," *Elsevier Computer Networks*, vol. 56, pp. 85–98, 2012. 10, 11, 16, 19, 20, 21, 32, 39, 43, 63, 64, 72, 146

[45] B. Donnet, B. Gueye, and M. A. Kaafar, "A survey on network coordinates systems, design, and security," *Communications Surveys & Tutorials, IEEE*, vol. 12, pp. 488–503, 2010. 65

[46] S. Doshi, F. Monrose, and A. D. Rubin, "Efficient memory bound puzzles using pattern databases," in *Springer ACNS*, 2006. 117

[47] B. Eriksson, P. Barford, B. Maggs, and R. Nowak, "Posit: A Lightweight Approach for IP Geolocation," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 2, pp. 2–11, 2012. 11, 19

[48] B. Eriksson, P. Barford, J. Sommers, and R. Nowak, "A Learning-Based Approach for IP Geolocation," in *Springer PAM*, 2010. 11, 19

[49] B. Eriksson and M. Crovella, "Understanding Geolocation Accuracy using Network Geometry," in *IEEE INFOCOM Miniconference*, 2013. 19

[50] I. Fette and A. Melnikov, "The WebSocket Protocol," RFC 6455 (Proposed Standard), 2011. 49, 128

[51] "HBO is cracking down on Canadians accessing streaming service HBO Now," http://business.financialpost.com/fp-tech-desk/hbo-is-cracking-down-on-canadians-accessing-streaming-service-hbo-now?__lsa=b293-7dd0, Financial Post, April 2015. 62

[52] "Netflix Is Under Pressure To Ban VPN Use," http://www.forbes.com/sites/ianmorris/2014/09/17/netflix-is-under-pressure-to-ban-vpn-use/, Forbes, Sep 2014. 19

[53] "Food, Nightlife, Entertainment." https://foursquare.com/, Foursquare Labs, Inc. 17

[54] "GNU Project Archives (/inetutils-1.9.2/ping)," http://ftp.gnu.org/gnu/inetutils/, Free Software Foundation, 2014. 19, 23, 144, 145

[55] C. L. Fullmer and J. Garcia-Luna-Aceves, *Solutions to hidden terminal problems in wireless networks.* ACM, 1997. 94

[56] S. Gambs, M.-O. Killijian, M. Roy, and M. Traoré, "PROPS: A PRivacy-preserving lOcation Proof System," in *IEEE SRDS*, 2014. 16

[57] J. Garcia-Alfaro, M. Barbeau, and E. Kranakis, "Secure geolocalization of wireless sensor nodes in the presence of misbehaving anchor nodes," *annals of telecommunications*, vol. 66, no. 9-10, pp. 535–552, 2011. 15

[58] "Geodetic Calculation Methods," http://www.ga.gov.au/earth-monitoring/geodesy/geodetic-techniques/calculation-methods.html, Geoscience Australia. 30

[59] P. Gill, Y. Ganjali, B. Wong, and D. Lie, "Dude, where's that IP? Circumventing measurement-based IP geolocation," in *USENIX Security*, 2010. 3, 4, 5, 14, 15, 18, 19, 25, 29, 30, 31, 35, 39, 41, 42, 43, 63, 72, 73, 75

[60] F. Girlich, M. Rossberg, G. Schaefer, T. Boehme, and J. Schreyer, "Bounds for the Security of the Vivaldi Network Coordinate System," in *IEEE NetSys*, 2013. 42, 65

[61] S. Goldberg, D. Xiao, E. Tromer, B. Barak, and J. Rexford, "Path-quality Monitoring in the Presence of Adversaries," in *ACM SIGMETRICS*, 2008. 42

[62] M. Gondree and Z. N. Peterson, "Geolocation of data in the cloud," in *ACM CODASPY*, 2013. 19, 43

[63] A. González-Tablas, K. Kursawe, B. Ramos Álvarez, and A. R. Garnacho, "Survey on Location Authentication Protocols and Spatial-Temporal Attestation Services," in *EUC Workshops*. Springer, 2005, vol. 3823, pp. 797–806. 16

[64] A. I. González-Tablas Ferreres, B. Ramos Álvarez, and A. R. Garnacho, "Guaranteeing the authenticity of location information," *Pervasive Computing, IEEE*, vol. 7, no. 3, pp. 72–80, 2008. 16

[65] D. Gross, J. Shortle, J. Thompson, and C. Harris, *Fundamentals of Queueing Theory*. Wiley, Hoboken, NJ, 2008. 120

[66] B. Gueye, S. Uhlig, A. Ziviani, and S. Fdida, "Leveraging Buffering Delay Estimation for Geolocation of Internet Hosts," in *Networking*, 2006, vol. 3976, pp. 319–330. 11

[67] B. Gueye, A. Ziviani, M. Crovella, and S. Fdida, "Constraint-based geolocation of Internet hosts," *IEEE/ACM Trans. Netw.*, vol. 14, pp. 1219–1232, 2006. 9, 10, 19, 20, 21, 32, 33, 39, 43, 64, 65

[68] C. Guo, Y. Liu, W. Shen, H. Wang, Q. Yu, and Y. Zhang, "Mining the Web and the Internet for Accurate IP Address Geolocations," in *IEEE INFOCOM*, 2009. 12, 19

[69] O. Gurewitz, I. Cidon, and M. Sidi, "One-way delay estimation using network-wide measurements," *IEEE/ACM Trans. Netw.*, vol. 14, pp. 2710–2724, 2006. 51

[70] A. Hernandez and E. Magana, "One-way Delay Measurement and Characterization," in *Networking and Services. ICNS*, 2007. 4, 46

[71] P. Holleczek, R. Karch, R. Kleineisel, S. Kraft, J. Reinwand, and V. Venus, "Statistical characteristics of active IP one way delay measurements," in *IEEE ICNS*, 2006. 88

[72] M. Honda, Y. Nishida, C. Raiciu, A. Greenhalgh, M. Handley, and H. Tokuda, "Is It Still Possible to Extend TCP?" in *ACM IMC*, 2011. 14

[73] W.-B. Hsieh and J.-S. Leu, "A Time and Location Information Assisted OTP Scheme," *Wireless Personal Communications*, vol. 72, no. 1, pp. 509–519, 2013. 62

[74] P. Hsu and H. Robbins, "Complete convergence and the law of large numbers," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 33, p. 25, 1947. 72

[75] D. Hu and C.-L. Wang, "GPS-Based Location Extraction and Presence Management for Mobile Instant Messenger," *LNCS Embedded and Ubiquitous Computing*, vol. 4808, pp. 309–320, 2007. 3, 12

[76] Y.-C. Hu, A. Perrig, and D. Johnson, "Packet Leashes: A Defense against Wormhole Attacks in Wireless Networks," in *IEEE INFOCOM*, 2003. 16

[77] J. Hua, Y. Cui, Y. Yang, and H. Li, "Analysis and prediction of jitter of Internet one-way time-delay for teleoperation systems," in *IEEE INDIN*, 2013, pp. 612–617. 59

[78] "Hulu. Watch TV. Watch Movies. Online. Free." http://www.hulu.com/, Hulu. 2, 19, 66, 116

[79] IEEE, "EEE 802.11, The Working Group Setting the Standards for Wireless LANs." [Online]. Available: http://www.ieee802.org/11/ 93, 98

[80] Internet2, "Home — Internet2," http://www.internet2.edu, 2015. 79

[81] C. Jin, H. Wang, and K. G. Shin, "Hop-count Filtering: An Effective Defense Against Spoofed DDoS Traffic," in *ACM CCS*, 2003. 116

[82] N. Johnson, S. Kotz, and N. Balakrishnan, *Continuous Univariate Distributions*. John Wiley, 1994, vol. 1. 97, 98

[83] A. Juels and J. G. Brainard, "Client Puzzles: A Cryptographic Countermeasure Against Connection Depletion Attacks." in *NDSS*, 1999. 116, 117

[84] M. A. Kaafar, L. Mathy, C. Barakat, K. Salamatian, T. Turletti, and W. Dabbous, "Securing Internet Coordinate Embedding Systems," in *ACM SIGCOMM*, 2007. 42

[85] E. Katz-Bassett, J. P. John, A. Krishnamurthy, D. Wetherall, T. Anderson, and Y. Chawathe, "Towards IP geolocation using delay and topology measurements," in *ACM IMC*, 2006. 11, 29, 35, 65

[86] S. M. Kay, *Fundamentals of statistical signal processing, Vol. II: Detection Theory.* Pearson Education, 1998, vol. 3. 11

[87] R. Khan, S. Zawoad, M. M. Haque, and R. Hasan, "OTIT: Towards Secure Provenance Modeling for Location Proofs," in *ACM ASIA CCS*, 2014. 17

[88] R. Khan, S. Zawoad, M. Haque, and R. Hasan, "'Who, When, and Where?' Location Proof Assertion for Mobile Devices," in *Data and Applications Security and Privacy XXVIII.* Springer, 2014, vol. 8566, pp. 146–162. 16

[89] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach*, 4th ed., M. Hirsch, Ed. Addison-Wesley, 2007, vol. 1. 9, 93

[90] S. Laki, P. Mátray, P. Hága, I. Csabai, and G. Vattay, "A model based approach for improving router geolocation," *Computer Networks*, vol. 54, pp. 1490–1501, 2010. 11

[91] S. Laki, P. Mátray, P. Hága, T. Sebók, I. Csabai, and G. Vattay, "Spotter: A Model Based Active Geolocation Service," in *IEEE INFOCOM*, 2011. 3, 19, 34, 43, 63, 64, 65

[92] R. Landa, R. G. Clegg, J. T. Araújo, E. Mykoniati, D. Griffin, and M. Rio, "Measuring the Relationships between Internet Geography and RTT," in *IEEE ICCCN*, 2013. 9, 64, 65, 72, 117

[93] D. Li, J. Chen, C. Guo, Y. Liu, J. Zhang, Z. Zhang, and Y. Zhang, "IP-Geolocation Mapping for Moderately Connected Internet Regions," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, pp. 381–391, 2013. 3, 12, 19, 90

[94] M. Li, S. Salinas, and P. Li, "LocaWard: A security and Privacy Aware Location-Based Rewarding System," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 2, pp. 343–352, 2014. 17

[95] W. Li, R. K. Mok, R. K. Chang, and W. W. Fok, "Appraising the Delay Accuracy in Browser-based Network Measurement," in *ACM IMC*, 2013. 49, 128

[96] X. Lin and W. He, "WiLoVe: A WiFi-coverage based Location Verification System in LBS," *Procedia Computer Science*, vol. 34, no. 0, pp. 484–491, 2014. 16

[97] H. Liu, S. Saroiu, A. Wolman, and H. Raj, "Software Abstractions for Trusted Sensors," in *ACM MobiSys*, 2012. 13

[98] J. Liu, "A novel method for estimating the variable and constant components of one-way delays without using the synchronized clocks," in *IEEE ICNC*, 2014. 51

[99] C. Lumezanu, R. Baden, N. Spring, and B. Bhattacharjee, "Triangle inequality and routing policy violations in the Internet," in *Springer PAM*, 2009. 70

[100] W. Luo and U. Hengartner, "Proving Your Location Without Giving up Your Privacy," in *ACM HotMobile*, 2010. 16

[101] ——, "VeriPlace: A Privacy-Aware Location Proof Architecture," in *ACM SIGSPATIAL*, 2010. 16

[102] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "iPlane: An Information Plane for Distributed Services," in *USENIX OSDI*, 2006. 32

[103] "MaxMind - IP Geolocation and Online Fraud Prevention," https://www.maxmind.com, MaxMind, Inc. 3, 13, 19, 63

[104] I. Miller, J. E. Freund, and R. A. Johnson, *Probability and statistics for Engineers*. Prentice-Hall Englewood Cliffs, NJ, 1965, vol. 1110. 105

[105] D. Mills, J. Martin, J. Burbank, and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification," RFC 5905 (Proposed Standard), 2010. 78

[106] H. Moritz, "Geodetic Reference System 1980," *Springer-Verlag Geodesy*, vol. 74, pp. 395–405, 2000. 31

[107] J. A. Muir and P. C. van Oorschot, "Internet geolocation: Evasion and counterevasion," *ACM Comput. Surv.*, vol. 42, pp. 4:1–4:23, 2009. 3, 5, 8, 12, 13, 19, 42, 63, 72, 128

[108] A. Mukherjee, "On the Dynamics and Significance of Low Frequency Components of Internet Load," *Internetworking: Research and Experience*, vol. 5, pp. 163–205, 1992. 58

[109] "Nanjee," http://www.nanjee.net/, Nanjee, Inc. 19

[110] A. Nezhad and Y. Azizi, "GPS clock based one way delay measurement and modeling in web environment," in *Computer and Knowledge Engineering (ICCKE)*, 2014. 59

[111] B. Ngamwongwattana and R. Thompson, "Measuring one-way delay of VoIP packets without clock synchronization," in *IEEE I2MTC*, 2009. 59

[112] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, "Cookieless monster: Exploring the Ecosystem of Web-based Device Fingerprinting," in *IEEE S&P*, 2013. 116

[113] V. N. Padmanabhan and L. Subramanian, "An investigation of geographic mapping techniques for Internet hosts," in *ACM SIGCOMM*, 2001. 9, 10, 19, 20, 21, 32, 39

[114] "PANDORA," http://www.pandora.com/, Pandora Media, Inc. 33, 66

[115] B. Parno, "Bootstrapping Trust in a "Trusted" Platform." in *USENIX HotSec*, 2008. 13

[116] A. Pathak, H. Pucha, Y. Zhang, Y. C. Hu, and Z. M. Mao, "A Measurement Study of Internet Delay Asymmetry," in *Springer PAM*, 2008, pp. 182–191. 4, 46, 64, 67

[117] R. Percacci and A. Vespignani, "Scale-free behavior of the Internet global performance," *Springer EPJ B—Condensed Matter and Complex Systems*, vol. 32, pp. 411–414, 2003. 12, 29

[118] Z. N. J. Peterson, M. Gondree, and R. Beverly, "A position paper on data sovereignty: The importance of geolocating data in the cloud," in *USENIX HotCloud*, 2011. 43, 116

[119] J. Philips, *Online Curve and Surface Fitting at ZunZun.com*, 2011. 123

[120] S. Pidcock and U. Hengartner, "Zerosquare: A privacy-Friendly Location Hub for Geosocial Applications," in *IEEE MoST*, 2013. 16

[121] I. Poese, S. Uhlig, M. A. Kaafar, B. Donnet, and B. Gueye, "IP geolocation databases: unreliable?" *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, pp. 53–56, 2011. 13, 19

[122] I. Polakis, S. Volanis, E. Athanasopoulos, and E. P. Markatos, "The Man Who Was There: Validating Check-ins in Location-based Services," in *ACM ACSAC*, 2013. 16, 19, 63

[123] A. Popescu, "Geolocation API Specification," http://www.w3.org/TR/geolocation-API/, 2013. 12

[124] J. Postel, "User Datagram Protocol," RFC 768 (Internet Standard), 1980. 27

[125] ——, "Internet Control Message Protocol," RFC 792 (Internet Standard), 1981. 19, 22, 23, 24, 27, 41, 144

[126] ——, "Internet Protocol," RFC 791 (Internet Standard), 1981. 27

[127] A. Ranganathan, N. O. Tippenhauer, B. Škorić, D. Singelée, and S. Čapkun, "Design and Implementation of a Terrorist Fraud Resilient Distance Bounding System," in *Computer Security–ESORICS*. Springer, 2012, vol. 7459, pp. 415–432. 42

[128] P. Raptis, V. Vitsas, and K. Paparrizos, "Packet Delay Metrics for IEEE 802.11 Distributed Coordination Function," *Mobile Networks and Applications*, vol. 14, pp. 772–781, 2009. 99, 100, 101, 102, 104, 105, 106, 107, 108, 109, 111, 112, 113

[129] ReLuc, "Geolocater :: Add-ons for Firefox," https://addons.mozilla.org/en-us/firefox/addon/geolocater/, April 2013. 13

[130] E. Rescorla, *SSL and TLS: Designing and Building Secure Systems*. Addison-Wesley Reading, 2001. 116

[131] M. Santos, S. Fernandes, and C. Kamienski, "Conducting network research in large-scale platforms: Avoiding pitfalls in planetlab," in *Advanced Information Networking and Applications (AINA), 2014 IEEE 28th International Conference on*, 2014. 80

[132] S. Saroiu and A. Wolman, "Enabling New Mobile Applications with Location Proofs," in *ACM HotMobile*, 2009. 16

[133] N. Sastry, U. Shankar, and D. Wagner, "Secure Verification of Location Claims," in *ACM WiSec*. ACM, 2003. 15, 63

[134] S. Shalunov, B. Teitelbaum, A. Karp, J. Boote, and M. Zekauskas, "A One-way Active Measurement Protocol (OWAMP)," RFC 4656 (Proposed Standard), 2006. 4, 46, 51, 59

[135] Y. Shavitt and N. Zilberman, "A geolocation databases study," *IEEE J. Sel. Areas Commun.*, vol. 29, pp. 2044–2056, 2011. 13, 19

[136] M. Shin, M. Park, D. Oh, B. Kim, and J. Lee, "Clock Synchronization for One-Way Delay Measurement: A Survey," in *Advanced Communication and Networking*. Springer, 2011, vol. 199, pp. 1–10. 59

[137] S. Siwpersad, B. Gueye, and S. Uhlig, "Assessing the Geographic Resolution of Exhaustive Tabulation for Geolocating Internet Hosts," in *Springer PAM*, 2008. 13, 19

[138] S. Son and V. Shmatikov, "The hitchhiker's guide to DNS cache poisoning," *LNCS Security and Privacy in Communication Networks*, vol. 50, pp. 466–483, 2010. 41

[139] L. Subramanian, V. N. Padmanabhan, and R. H. Katz, "Geographic properties of Internet routing," in *USENIX ATC*, 2002. 31, 66, 72

[140] "Hulu Blocks VPN Users Over Piracy Concerns," https://torrentfreak.com/hulu-blocks-vpn-users-over-piracy-concerns-140425/, TorrentFreak, April 2014. 19

[141] J. F. Traub, *Iterative Methods for the Solution of Equations*. AMS Bookstore, 1982. 112, 121

[142] I. Trestian, S. Ranjan, A. Kuzmanovic, and A. Nucci, "Measuring Serendipity: Connecting People, Locations and Interests in a Mobile 3G Network," in *ACM IMC*, 2009. 3

[143] M. Trimble, "The Future of Cybertravel: Legal Implications of the Evasion of Geolocation," *HeinOnline Fordham Intell. Prop. Media & Ent. LJ*, vol. 22, pp. 567–657, 2011. 19, 63

[144] S. Tritilanunt, C. Boyd, E. Foo, and J. Gonzlez Nieto, "Toward Non-Parallelizable Client Puzzles," in *LNCS Cryptology and Network Security*. Springer, 2007, vol. 4856, pp. 247–264. 117

[145] A. Vakili and J. Gregoire, "Accurate One-Way Delay Estimation: Limitations and Improvements," *IEEE Trans. Instrum. Meas.*, vol. 61, pp. 2428–2435, 2012. 51, 61

[146] P. C. van Oorschot and S. Stubblebine, "Countering Identity Theft Through Digital Uniqueness, Location Cross-Checking, and Funneling," in *Springer FC*, 2005, pp. 31–43. 17

[147] G. Wang, B. Zhang, and T. Ng, "Towards network triangle inequality violation aware distributed systems," in *ACM IMC*, 2007. 64, 70, 72

[148] Y. Wang, D. Burgener, M. Flores, A. Kuzmanovic, and C. Huang, "Towards street-level client-independent IP geolocation," in *USENIX NSDI*, 2011. 12, 19

[149] B. R. Water and E. W. Felten, "Secure, Private Proofs of Location," Princeton University, Tech. Rep., 2003. 16

[150] "Free VPN Service — Free VPN Software - Hotspot Shield VPN," http://www.hotspotshield.com/, Web Hosting Logic, Inc. 6, 14

[151] W. Wei, B. Wang, C. Zhang, J. Kurose, and D. Towsley, "Classification of access network types: Ethernet, wireless LAN, ADSL, cable modem or dialup?" *Computer Networks*, vol. 52, pp. 3205–3217, 2008. 116

[152] B. Wong, A. Slivkins, and E. G. Sirer, "Meridian: A lightweight network location service without virtual coordinates," in *ACM SIGCOMM*, 2005. 42

[153] B. Wong, I. Stoyanov, and E. G. Sirer, "Octant: a comprehensive framework for the geolocalization of Internet hosts," in *USENIX NSDI*, 2007. 9, 12, 48, 64, 71, 117

[154] Y. Xie, F. Yu, K. Achan, E. Gillum, M. Goldszmidt, and T. Wobber, "How Dynamic Are IP Addresses?" in *ACM SIGCOMM*, 2007. 14

[155] S.-H. Yook, H. Jeong, and A.-L. Barabási, "Modeling the Internet's large-scale topology," *National Academy of Sciences*, pp. 13 382–13 386, 2002. 65

[156] I. Youn, B. Mark, and D. Richards, "Statistical Geolocation of Internet Hosts," in *IEEE ICCCN.*, 2009. 10, 11, 19

[157] D. J. Zage and C. Nita-Rotaru, "On the accuracy of decentralized virtual coordinate systems in adversarial networks," in *ACM CCS*, 2007. 42

[158] P. A. Zandbergen, "Accuracy of iPhone Locations: A Comparison of Assisted GPS, WiFi and Cellular Positioning," *Transactions in GIS*, vol. 13, pp. 5–25, 2009. 3, 12

[159] A. Zeitoun, C.-N. Chuah, S. Bhattacharyya, and C. Diot, "An AS-level Study of Internet Path Delay Characteristics," in *IEEE GLOBECOM*, 2004. 46, 59

[160] Y. Zeng, J. Cao, J. Hong, S. Zhang, and L. Xie, "Secure localization and location verification in wireless sensor networks: a survey," *Springer The Journal of Supercomputing*, vol. 64, pp. 685–701, 2013. 42

[161] Y. Zhang and H. Zhang, "Triangulation Inequality Violation in Internet Delay Space," in *Advances in Computer Science and Information Engineering*. Springer, 2012, vol. 169, pp. 331–337. 83, 90

[162] Y. Zhang and N. Duffield, "On the Constancy of Internet Path Properties," in *ACM IMW*, 2001. 70, 71

[163] Z. Zhu and G. Cao, "APPLAUS: A Privacy-Preserving Location Proof Updating System for location-based services," in *IEEE INFOCOM*, 2011. 16

[164] A. Ziviani, S. Fdida, J. F. de Rezende, and O. C. M. B. Duarte, "Demographic placement for Internet host location," in *GLOBECOM*, 2003. 10

[165] ——, "Similarity models for internet host location," in *IEEE ICON*, 2003. 9

[166] ——, "Toward a Measurement-Based Geographic Location Service," in *Springer PAM*, 2004. 10, 19

[167] ——, "Improving the accuracy of measurement-based geographic location of Internet hosts," *Elsevier Computer Networks*, vol. 47, pp. 503–523, 2005. 9, 10, 19, 64, 72

Appendix A

RTT Measuring Tools

To illustrate the ease of manipulating delays as measured by common network utilities, we show code snippets of example utilities lacking delay-measurement integrity.

Recall from Section 3.3 that the sender in the stateless implementation places the timestamp s_i (packet-creation time) in the DATA field of the ICMP packet.[1] From GNU's *ping* (*ping.c*) [54]:

```
502 if (PING_TIMING (data_length))
503 {
504   struct timeval tv;
505   gettimeofday (&tv, NULL);
506   ping_set_data (ping, &tv, 0, sizeof (tv), USE_IPV6);
507 }
```

The variable `tv` represents our s_i. When the echo-reply is received, the sender observes the receiving time r_i, reads s_i from the echoed packet, and uses them to calculate the RTT using (3.3). From GNU's *ping* (*ping_echo.c*) [54], when the sender receives the echo-reply:

```
181 struct timeval tv;
182 int timing = 0;
183 double triptime = 0.0;
184
185 gettimeofday (&tv, NULL);
```

⋮

```
196 struct timeval tv1, *tp;
197
```

[1] ICMP types 13 and 14 (timestamp, and timestamp reply), can also be used to measure RTTs; RFC 792 specifies recording sending and receiving timestamps in their DATA field [125]. However, we did not notice many implementations of these types.

```
198 timing++;
199 tp = (struct timeval *) icmp->icmp_data;
200
201 /* Avoid unaligned data: */
202 memcpy (&tv1, tp, sizeof (tv1));
203 tvsub (&tv, &tv1);
204 triptime = ((double) tv.tv_sec) * 1000.0 + (double) tv.tv_usec) / 1000.0;
```

⋮

```
227 if (timing)
228   printf (" time=%.3f ms", triptime);
```

The variable `timing` is true if `datalen - PING_HEADER_LEN >= sizeof (struct timeval)`. Thus, from line 502 in *ping.c* and 227-228 in *ping_echo.c* above, such implementation of *ping* fails to calculate the RTT if the packet size was less than the size of the *timeval* struct.[2]

For the stateful echo-request/reply implementation, recall that the sender records s_i in its local memory. These stateful utilities commonly fill the DATA field using a fixed predefined pattern; e.g., from GNU's *traceroute* [54] (*src/traceroute.c*):

```
664 char data[] = "SUPERMAN";
665
666 len = sendto (t->udpfd, (char *) data, sizeof (data),0, (struct sockaddr *) &t->
        to, sizeof (t->to));
```

⋮

```
679 if (gettimeofday (&t->tsent, NULL) < 0)
680   error (EXIT_FAILURE, errno, "gettimeofday");
```

where `t` is a struct (locally) holding information about an issued *traceroute* packet. When an echo-reply is received [54] (*src/traceroute.c*):

```
383 gettimeofday (&now, NULL);
384
385 now.tv_usec -= trace->tsent.tv_usec;
386 now.tv_sec -= trace->tsent.tv_sec;
```

⋮

```
417 triptime = ((double) now.tv_sec) * 1000.0 + ((double) now.tv_usec) / 1000.0;
```

[2]The *timeval* struct could either be 8 or 16 bytes depending on the platform. The packet size is commonly set by the *-s* option.

⋮

```
438 printf (" %.3fms ", triptime);
```

The snippets provided herein are only examples of a wide range of utilities adopting similar behaviors. They show how predictable packet contents of commonly-used utilities could be, and provide evidence of lack of integrity in delay measurement. We assert that, at their current state, none of these tools are ready for use in security-sensitive systems. Unfortunately, many such systems either rely on these tools [44], or fail to propose integrity-preserving alternatives.

Appendix B

Proofs

In this appendix, the three claims made in Chapter 5 are proved.

Notation. The notation $\bigcirc XY(k)$ refers to the ellipse determined by the foci X and Y whose major axis is k meters long; $\overline{AB}$ for the *length* of line segment AB; and $\overleftrightarrow{XY}$ refers to the straight line passing by the points X and Y. Consider $\triangle XYZ$ in Fig. B.1. Regions A_1, A_2 and A_3 are those outside $\triangle XYZ$ delimited by the pairs $(\overleftrightarrow{XZ}, \overleftrightarrow{YZ})$, $(\overleftrightarrow{XY}, \overleftrightarrow{XZ})$ and $(\overleftrightarrow{XY}, \overleftrightarrow{YZ})$ respectively, such that none of $\triangle XYZ$'s exterior angles belong to A_1, A_2 or A_3. Regions B_1, B_2 and B_3 are those outside $\triangle XYZ$ delimited by the region pairs (A_1, A_2), (A_2, A_3) and (A_3, A_1) respectively. A point P outside $\triangle XYZ$ will either fall in region $A = A_1 \cup A_2 \cup A_3$ or $B = B_1 \cup B_2 \cup B_3$.

Proof of Claim 1

Recall Claim: *Let P be a point in the Cartesian plane, and let $\triangle XYZ$ be the triangle determined by the points X, Y and Z. If P is strictly outside $\triangle XYZ$, then the sum of the areas of $\triangle XYP$, $\triangle XPZ$ and $\triangle PYZ$ is greater than the area of $\triangle XYZ$.*

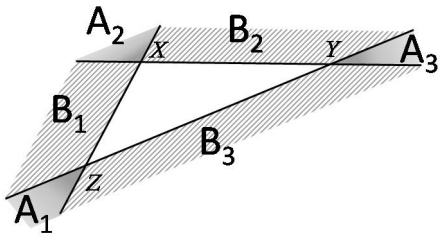
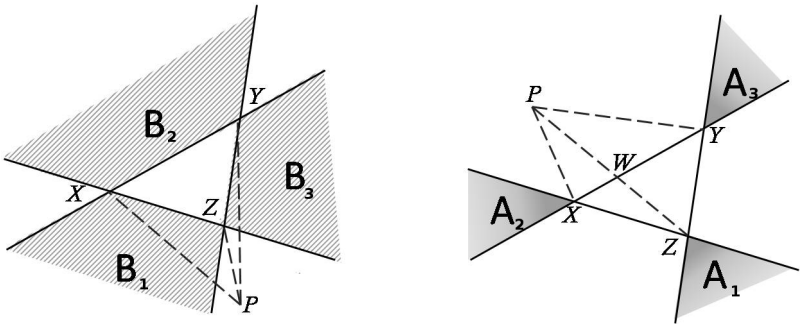


Figure B.1: Regions $A = A_1 \cup A_2 \cup A_3$ and $B = B_1 \cup B_2 \cup B_3$ outside $\triangle XYZ$.



(a) (b)

Figure B.2: If P is outside $\triangle XYZ$, the sum of the areas of $\triangle XYP$, $\triangle XPZ$ and $\triangle ZPY$ will be larger than the area of $\triangle XYZ$.

First, assume that P is in region A; then:

Claim 4 *If P is in region A, then the area of one of the triangles $\triangle XYP$, $\triangle XPZ$ or $\triangle PYZ$ will be larger than the area of $\triangle XYZ$.*

Proving claim 4 suffices to prove claim 1 for region A because if the area of only one triangle by itself exceeds the area $\triangle XYZ$, then the sum of the areas of the three triangles ($\triangle XYP$, $\triangle XPZ$ and $\triangle PYZ$) will definitely exceed the area of $\triangle XYZ$. To prove claim 4, assume that P is in region A_1, as shown in Fig. B.2(a). In this case, the one triangle (referred to in claim 4) whose area is larger than that of $\triangle XYZ$ is $\triangle XYP$. The proof follows.

Proof:
Since region A_1 is bound by the straight line pair $(\overleftrightarrow{XZ}, \overleftrightarrow{YZ})$.
Therefore $\angle YXP > \angle YXZ$ and $\angle XYP > \angle XYZ$.
Therefore Z is inside $\triangle XYP$.
Since line segment XY is shared between $\triangle XYZ$ and $\triangle XYP$.
Therefore $\triangle XYZ \subset \triangle XYP$.
Therefore $area(\triangle XYZ) < area(\triangle XYP)$. ■

Note that an analogous proof holds if P is in A_2 or A_3. For region B:

Claim 5 *If P is in region B, then the sum of the areas of two of the three triangles $\triangle XYP$, $\triangle XPZ$ or $\triangle PYZ$ will be larger than the area of $\triangle XYZ$.*

Again, proving claim 5 suffices to prove claim 1 for region B because the sum of the areas of the three triangles ($\triangle XYP$, $\triangle XPZ$ and $\triangle PYZ$) will definitely exceed the

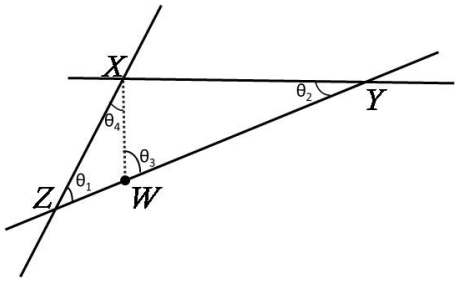


Figure B.3: If $\overline{XZ} \leq \overline{XY}$ and W is inside $\triangle XYZ$, then $\overline{XW} \leq \overline{XY}$.

area of $\triangle XYZ$ if the areas of two of the three triangles together exceed the area $\triangle XYZ$. To prove claim 5, assume that P is in region B_2, as shown in Fig. B.2(b); line segment PZ intersects XY in W. In this case, the two triangles (referred to in claim 5) are $\triangle XPZ$ and $\triangle ZPY$. The proof follows.

Proof:
Since P, W and Z are collinear, W is between P and Z, and
Since line segment XZ is shared between $\triangle XWZ$ and $\triangle XPZ$
Therefore $\triangle XWZ \subset \triangle XPZ$
Similarly, $\triangle ZWY \subset \triangle ZPY$
Therefore $(\triangle XWZ \cup \triangle ZWY) \subset (\triangle XPZ \cup \triangle ZPY)$
Therefore $\triangle XYZ \subset (\triangle XPZ \cup \triangle ZPY)$.
Therefore $area(\triangle XYZ) < area(\triangle XPZ) + area(\triangle ZPY)$. ■

Analogous proof holds if P is in B_1 or B_3. This concludes the proof to Claim 1.

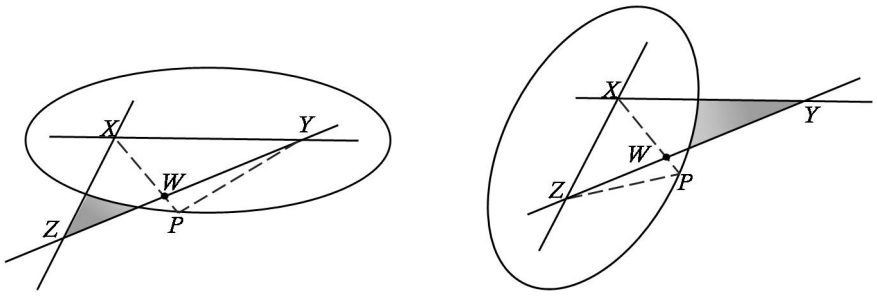
Proof of Claim 2

Recall Claim: *Let W be a point in the Cartesian plane, and let $\triangle XYZ$ be the triangle determined by the points X, Y and Z such that $\overline{XZ} \leq \overline{XY}$. If $\overline{XW} > \overline{XY}$, then W is strictly outside of $\triangle XYZ$.*

This Claim can be rewritten as:

Claim 6 *Let W be a point in the Cartesian plane, and let $\triangle XYZ$ be the triangle determined by the points X, Y and Z; $\overline{XZ} \leq \overline{XY}$. If W is inside $\triangle XYZ$, then $\overline{XW} \leq \overline{XY}$.*

which is the logical transposition $(P \rightarrow Q) \vdash (\neg Q \rightarrow \neg P)$ of Claim 2, where P is



(a) (b)

Figure B.4: When $P \in B_3$, then $\triangle XYZ \subset \{\bigcirc XY(\overline{XP}+\overline{PY}) \cup \bigcirc XZ(\overline{XP}+\overline{PZ})\}$.

the event "$\overline{XW} > \overline{XY}$", and Q is the event "W is strictly outside of $\triangle XYZ$". The following proves that $\overline{XW} \leq \overline{XY}$ holds when $\overline{XW}$ is the maximum that maintains W inside $\triangle XYZ$, which is when W lies on line segment YZ (see Fig. B.3).

Proof:
Since $\overline{XZ} \leq \overline{XY}$
Therefore $\theta_2 \leq \theta_1$.
Since W lies on line segment YZ
Therefore $\theta_1 + \theta_4 = \theta_3$.
Therefore $\theta_1 \leq \theta_3$.
Therefore $\theta_2 \leq \theta_3$.
Therefore $\overline{XW} \leq \overline{XY}$. ■

Proof of Claim 3

Recall Claim: *Let P be a point in the Cartesian plane, and let $\triangle XYZ$ be the triangle determined by the points X, Y and Z. If P is strictly outside $\triangle XYZ$, then increasing the sums $\overline{XP}+\overline{PZ}$, $\overline{XP}+\overline{PY}$ or $\overline{YP}+\overline{PZ}$ without reducing at least one of the other sums cannot place P inside $\triangle XYZ$.*

Similar to the proof of Claim 1, the proof of Claim 3 is split into two parts: when $P \in A$ and when $P \in B$. For part one, first assume that $P \in A_1$. In this case, according to the isoperimetric inequality, $\overline{XP}+\overline{PY}$ must be greater than $\overline{XZ}+\overline{ZY}$ because they both have the same starting and ending points, X and Y. Therefore, it is impossible to move P inside $\triangle XYZ$ without decreasing $\overline{XP}+\overline{PY}$. Analogous argument applies for regions A_2 and A_3.

Now to the case where $P \in B$. First assume that $P \in B_3$ as shown in Fig B.4. If

$\triangle XYZ \subset \{\bigcirc XY(\overline{XP}+\overline{PY}) \cup \bigcirc XZ(\overline{XP}+\overline{PZ})\}$,[1] is proved, then P cannot move to inside $\triangle XYZ$ without reducing $\overline{XP}+\overline{PY}$ or $\overline{XP}+\overline{PZ}$ because the sum of the lengths from any point on the ellipse to its pair of foci is constant; hence, the sum of the lengths from any point inside the ellipse to its pair of foci is less than that to any point on the ellipse.

Assume that $\triangle XYZ$ is split into two: $\triangle XYW$ and $\triangle XWZ$, where W is the intersection of line segments XP and YZ. Then, proving that $\triangle XYW \subset \bigcirc XY(\overline{XP}+\overline{PY})$ is as follows (see Fig. B.4(a)).

Proof:
Since X is a focus of the ellipse; P is a point on the ellipse; X, W and P are collinear; and $P \notin \triangle XYZ$
Therefore W is inside the ellipse.
Since Y is a focus of the ellipse
Therefore line segments XW, WY and XY are inside the ellipse.
Therefore $\triangle XYW \subset \bigcirc XY(\overline{XP}+\overline{PY})$. ■

Analogous proof applies to $\triangle XWZ \subset \bigcirc XZ(\overline{XP}+\overline{PZ})$ (Fig. B.4(b)). Therefore, when $P \in B_3$, it is impossible to move P inside $\triangle XYZ$ without reducing the summation $\overline{XP}+\overline{PY}$ or $\overline{XP}+\overline{PZ}$. The remaining regions of B can be proved in the same manner. Therefore, whenever $P \in B$, then $\triangle XYZ \subset \{\bigcirc XY(\overline{XP}+\overline{PY}) \cup \bigcirc YZ(\overline{YP}+\overline{PZ}) \cup \bigcirc XZ(\overline{XP}+\overline{PZ})\}$. This concludes the proof.

[1] Note that $\triangle \subset \bigcirc$ if $\forall p \in \triangle$, $p \in \bigcirc$.