Accurate Manipulation of Delay-based Internet Geolocation

AbdelRahman Abdou*‡ Ashraf Matrawy† Paul C. van Oorschot‡

‡School of Computer Science, †School of Information Technology
Carleton University
Ottawa, ON, Canada

ABSTRACT

Delay-based Internet geolocation techniques are repeatedly positioned as well suited for security-sensitive applications, *e.g.,* location-based access control, and credit-card verification. We present new strategies enabling adversaries to accurately control the forged location. Evaluation showed that using the new strategies, adversaries could misrepresent their true locations by over 15000km, and in some cases within 100km of an intended geographic location. This work significantly improves the adversary's control in misrepresenting its location, directly refuting the appropriateness of current techniques for security-sensitive applications. We finally discuss countermeasures to mitigate such strategies.

CCS Concepts

•**Security and privacy** → *Authentication;* •**Networks** → *Protocol correctness;*

Keywords

Location-aware Authentication; Location-based Services; Location Verification; Internet Measurements; Geolocation

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 [11], or restrict operations by clients' geographic locations. Examples include media streaming [6], online voting/gambling, location-based social networking [35], fraud prevention [7], and geography based routing [26]. Nanjee.net is one example that provides commercial geolocation services based on active network (delay) measurements [45]. 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.,* hostip.info, are unreliable [34] and evadable [30].

Delay-based IP geolocation techniques have accuracy advantages over other techniques [40]. They are also resilient to attacks that other techniques fall to [16], such as clients submitting false location information [45, 30]. The W3C geolocation API [36] defines an interface for LBSs to obtain a client's location from the browser, which uses technologies such as WiFi Positioning Systems (WPS) [48], Global Positioning System (GPS) [21], or IP address to location mapping, to determine its location. Technologies that allow users to hide their IP addresses, such as Virtual Private Networks (VPNs) and proxies, can be detected [7] and thwarted [3]; some LBSs, *e.g.,* Hulu,[1] have recently started employing these practices [43]. For these reasons, delay-based techniques are gaining increasing community support [14], specifically advocated [34], and repeatedly positioned as well suited for security-aware contexts, *e.g.,* ensuring legitimate storage of data in the cloud [19], or locating hidden servers [8].

Since 2001, more than 10 delay-based geolocation techniques have been proposed [20, 4, 13, 24, 27]. The round-trip time (RTT) delay is first measured between the client and a set of landmarks with known locations. These delays then get mapped to distances according to some predefined and usually landmark-specific function, and the client's location is estimated relative to the landmarks. Delay-based geolocation requires some way of measuring delays; because Internet Control Message Protocol (ICMP) [38] utilities, *e.g., ping* and *traceroute*, are ubiquitous and facilitate delay measurements, they are commonly used for that purpose [47, 12].

Gill *et al.* [16] studied an adversary's ability to distort delay-based techniques by selectively delaying response messages thereby increasing the observed RTTs. They modeled an adversary that uses the speed of light in fiber as an estimate to the traffic propagation speed over the Internet. Although such an adversary succeeded to misrepresent its location, it failed to control the country where it appears to be at [16], which is the granularity of general interest in practice [44, 33]. To the best of our knowledge, none of the delay-based geolocation techniques published to date have addressed these adversarial manipulations.

In this paper, we explain new attacks that enable an adversary to accurately control the forged location, with adversarial errors less than one-fifth of those achieved in previous literature [16],[2] allowing country-level control of the forged location. To achieve such an accuracy, we devise new strategies whereby adversaries can model the speed of traffic propagation, *without knowing the predefined delay-to-distance mapping function* as assumed in previous literature [16]. Additionally, our attacks exploit how delay-based geolocation techniques often fail to measure delays in an integrity-preserving manner. As we show, delay-measuring utilities commonly used by geolocation techniques lack in-

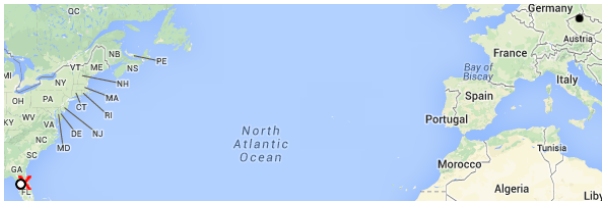
* Email: abdou@scs.carleton.ca

The final version of this paper is published in the proceedings of ACM Asia Conference on Computer and Communications Security (AsiaCCS 2017).

[1] http://www.hulu.com/

[2] See the distance error of adversaries A and B in Table 3.

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



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

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

Figure 1: Examples of adversarial accuracy in evading geolocation. • = true location of adversary; × = intended location of adversary; ○ = locations calculated by (a) CBG [20], (b) GeoPing [31], and (c) SegPoly [12]; *attempted dist* is that between • and ×; *dist error* for the adversary is that between × and ○. Map data: Google, INEGI, Basarsoft.

tegrity checking, as they are subject to (1) modifying and/or (2) predicting packet contents, enabling an adversary to *fully manipulate*, *i.e.,* increase and decrease (versus increase only [16]), the observed RTTs.

We analyze the effectiveness of the modeled adversaries on delay-based geolocation—the class of techniques shown to be most resilient to attacks [16]. We implemented three delay-based techniques, CBG [20], GeoPing [31] and segmented polynomial (*SegPoly* for short) [12], and evaluated adversarial location-forging accuracy. Some modeled adversaries forged their locations with *distance errors* <100 km (see Section 6.1). The distance error is that between the adversary's intended location and the one calculated by the geolocation technique. This relatively fine-grained location control was possible even for some who attempted fraudulent relocation more than 15,000 km away from their true locations. Figure 1 shows an example while manipulating each technique. No prior scientific literature clearly demonstrates that this level of adversarial location control is possible. We make the following contributions:

1. We devise strategies that enable an adversary to accurately forge the location calculated by delay-based geolocation techniques.
2. We show how an adversary can fully manipulate the delays measured by common utilities, and make available a proof-of-concept implementation.
3. We evaluate the manipulation effectiveness on three delay-based techniques, which is the class of techniques previously believed to be the most resilient to attacks [16], showing how powerful an adversary can be upon conducting such attacks on delay-based geolocation.

The rest of this paper is organized as follows. Section 2 reviews delay-based geolocation, and common delay measuring utilities. Section 3 discusses related work. Section 4 explains how RTTs can be fully manipulated (increased and decreased). The adversarial models and attack strategies are explained in Section 5. Section 6 analyzes the effect of manipulating RTTs on delay-based geolocation, and Section 7 compares that across different adversarial models. Section 8 suggests countermeasures, and Section 9 concludes.

2. BACKGROUND

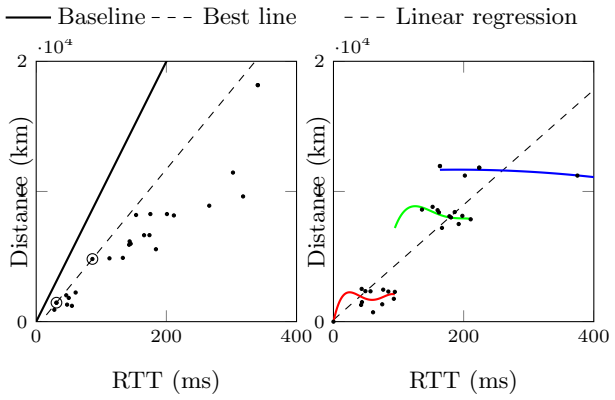
2.1 Delay-based IP Geolocation

In delay-based IP geolocation, the geographic location of the client machine is determined based on the observed network delays between the machine and a set of landmarks with known locations. These techniques assume the client is able to receive and respond to delay-measurement probes, but fail to use/propose mechanisms for preserving the integrity of the measured delays. Common ICMP-based utilities, like *ping* and *traceroute*, are often relied upon for that purpose [47, 12].

Despite a plethora of factors that affect the Internet delays between two nodes [46], numerous studies established that there is a strong correlation between delays and geographic distances [31, 20, 25]. The main characteristic relied on is the propagation delay. Most, if not all, delay-based geolocation techniques mitigate the effect of other undesired delay factors, such as queueing due to congestion, by using the minimum of multiple delays measurements to the client from each landmark. Typical values lie in the range of 10 to 20 RTT measurements. The research question then addressed by most techniques becomes finding the best function to map these delays to distances.

One exception is GeoPing [31]. Instead of mapping delays to distances, GeoPing matches the location of the client to a location where the most similar delay behavior would be observed. Assuming n landmarks and m reference nodes with known locations, 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 client's location is then matched to the node with the *nearest* delay vector, calculated as the n-dimensional Euclidean distance between the two vectors.

A key contribution of delay-based techniques is a function that accurately maps delays to distances. CBG [20] uses a linear mapping function called the *best line*, which is the one closest to all {RTT, distance} coordinates, lies above all of them, and has a non-positive intersection with the y-axis (the distance). An example of the best line from our experiments is shown in Fig. 2(a). The *baseline* function, which represents the speed of data in fiber [32], is also plotted on the same chart for reference. 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



(a) CBG [20] calibration (b) SegPoly [12] calibration

Figure 2: An example of delay-distance calibration from our experiments. (a) circled markers determine the best line; (b) 3-cluster polynomial fitting.

then estimated as the centroid of the intersection of circles whose centers are the landmarks and radii are the distances.

Dong *et al.* [12] 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 exploits that delay-to-distance ratios vary according to the spanned geographic distance [12]. Figure 2(b) shows an example from our experiments of segmented polynomial regression with $k = 3$. Coordinates in the first two clusters in Fig. 2(b) are fitted using a quartic polynomial (degree 4), and in the third using a quadratic one (degree 2). A least-square linear fitting is also plotted on the same chart for referencing.

2.2 Common RTT Measurement Techniques

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 packets $i \geq 0$ is:

$$s_i = T + i \cdot t \tag{1}$$

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

$$m_i = s_i + \gamma_1 = T + i \cdot t + \gamma_1 \tag{2}$$

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:

$$RTT_i = r_i - s_i = \gamma_1 + \gamma_2 \tag{3}$$

To measure RTTs, network utilities commonly use the ICMP protocol [38],[3] as it is implemented by default in most systems' protocol stack. An ICMP packet gets wrapped by

[3]Some utilities rely on TCP messages, e.g., *tcptraceroute*.

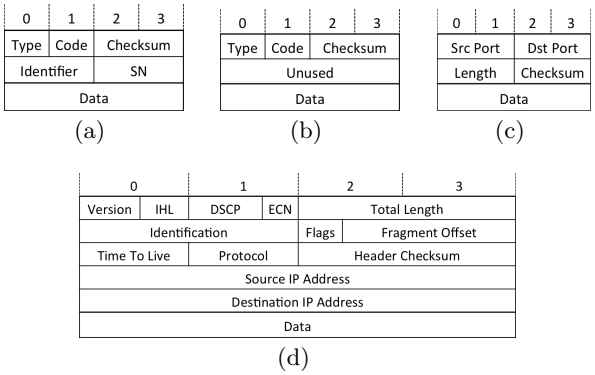


Figure 3: Packet/segment formats for (a) ICMP echo-request/reply; (b) ICMP destination-unreachable; (c) UDP; and (d) IPv4.

an IP packet for delivery. Eleven ICMP types are specified by 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 options commonly used to measure RTTs. The RFC does not specify a mechanism to calculate RTTs for either types [38].

Echo-request/reply. To construct an echo-request, the sender sets the TYPE and CODE fields to 8 and 0 respectively (see Fig. 3(a)), 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. The implementer can choose any values for the DATA, IDENTIFIER and SEQUENCE NUMBER fields [38]. We found that many, if not all, *ping* implementations on Linux,[4] BSD[5]and Mac OS[6] use the echo-request/reply, and place the process ID (PID) of the issuing process in the IDENTIFIER field. When echoing the message, RFC 792 specifies that the receiver should only change the TYPE field to 0 and recalculate the checksum, without defining a mechanism to ensure the described behavior, thus no integrity checking.

To calculate the RTT using the echo-request/reply options, two common implementations exist: *stateless* and *stateful*. In the former, 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). Other examples of such stateless implementation include, but not limited to, *ping* on FreeBSD and Mac OS.

In the stateful echo-request/reply implementation, the sender records s_i in its local memory. The RTT is calculated also using (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* using the ICMP option (*i.e.,* `traceroute -I <host>`), and *hping*[7] again using the ICMP option (*i.e.,* `hping3 -1 <host>`). These utilities commonly fill the

[4]http://ftp.gnu.org/gnu/inetutils/
[5]https://svnweb.freebsd.org/base/
[6]http://www.opensource.apple.com/source/network_cmds/network_cmds-433/ping.tproj/
[7]http://www.hping.org/download.php

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 this option, the sender issues a UDP segment destined to a port that is unlikely to be open on the receiver's machine. It then records s_i in its local memory, and commonly fills the DATA field with some fixed predefined patterns. If the port was actually closed, the receiver is expected to respond with an ICMP destination-unreachable message [5]. The sender records the receiving time r_i, and calculates the RTT using (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.

To construct a destination-unreachable message (Fig. 3(b)), the TYPE and CODE fields are set to 3, and the UNUSED field to 0. To enable the sender to match responses with their corresponding processes, RFC 792 specifies that the IP header and the first 8 payload bytes of the originally received IP packet are to be placed in the DATA field of the destination-unreachable message [38].

3. RELATED WORK

Gill *et al.* [16] studied the effect of delay increases on topology-aware and delay-based geolocation techniques, choosing one representative technique for each. The former, such as Octant [46], leverages the network topology to generate a richer set of constraints to geolocate clients. The authors [16] 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 5 below) and the latter was able to increase the number of hops to the landmarks. Against delay-based techniques, both adversaries were found to have limited control over the forged location [16]. Additionally, the authors found that delay-based techniques are generally more resilient to attacks than topology-aware ones [16], making them more suitable for security-sensitive applications than others. We thus focus on analyzing the effect of manipulations on delay-based geolocation techniques.

Contrary to what Gill *et al.* [16] concluded, we found that country-level control of forged locations is indeed possible while attacking delay-based geolocation, and that detecting these attacks is not trivial. Even when the adversary is only able to increase delays, we show below that devious traffic modeling strategies can almost double adversarial accuracy, compared to the strategies of Gill *et al.* [16].[8]

Muir *et al.* [30] 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 like *whois* databases. 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.* [18] 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 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. Thus, none of their solutions fit the problem studied herein.

Delay-based location verification techniques have been proposed [2]. However, proposals for single-hop wireless networks [49] cannot be directly applied to the Internet because of the difference in delay nature between both domains [16].

In Network Coordinate Systems (NCSs) [10], 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 to disrupt an NCS were explored [17], and proposals for securing NCSs addressed adversarial delay-increase [22].

[8]See the distance error of adversaries B and C in Table 3.

Table 1: Properties of ICMP-based utilities, and the effects of exploiting them on the observed RTTs. A bullet (•) means Method i has property j.

	Property (vulnerability)	Effect		Method			Discovered
		↑ RTT	↓ RTT	1	2	3	
1	Suspendable responses	✓		•	•	•	[16]
2	Modifiable pkt contents	✓	✓	•			herein
3	Predictable pkt contents		✓		•	•	herein

4. FULL DELAY MANIPULATION

From Section 2.2, the methods whereby a sender can measure RTTs using ICMP are:

- M1: stateless using echo-request/reply.
- M2: stateful using echo-request/reply.
- M3: stateful using destination-unreachable.

Table 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 1 as *vulnerabilities*. Note that despite having the same effect on RTTs, the first and second vulnerabilities in Table 1 enable an adversary to increase RTTs in a different way; likewise, the second and third decrease differently. The table also shows which of the three commonly-used RTT-measuring methods listed above has which vulnerability (third column). In each method, there are two ICMP vulnerabilities (•) that enable and adversary to increase and decrease RTTs.

Exploiting the first vulnerability in Table 1 enables the adversary to increase RTTs in all three methods, because the adversary needs only hold on to the response messages to increase the RTTs. Decreasing RTTs, in each method, is achieved as follows.

M1. *stateless-echo:* The packet-creation time, s_i, is recorded in the ICMP echo-request in this method. To decrease RTTs, the adversary increases the value of s_i before including it in the echo-reply. Changing s_i to $s_i + \delta$ decreases the observed RTTs by δ. Using (3), the sender calculates the manipulated

RTT of packet i as:

$$\text{RTT}'_i = r_i - (s_i + \delta) = RTT_i - \delta \quad (4)$$

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 its choosing, τ, by setting:

$$\delta = RTT_i - \tau \quad (5)$$

causing the sender to calculate the manipulated RTT as:

$$\text{RTT}'_i = r_i - (s_i + \delta) = RTT_i - \delta = \tau \quad (6)$$

To demonstrate a proof-of-concept, we set up a machine which manipulates RTTs when measured as in Method 1. When this machine is *ping*ed from most Linux, BSD or Mac machines, it behaves as follows: for the first 5 packets, the actual RTT is returned; for the next 25 packets, the actual RTT keeps decreasing by 2 ms. The same procedure repeats starting the 30^{th} packet, 60^{th} packet, etc.

M2. *stateful-echo:* To decrease RTTs, the adversary first estimates the sender *waiting* time, t in (1), 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 (2), from m_{i+1} for all $0 \leq i < n - 1$ (Section 2.2). 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:

$$t = \frac{1}{n-1} \sum_{i=0}^{n-2} (m_{i+1} - m_i) \quad (7)$$

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

$$m_i = m_{i-1} + t \quad (8)$$

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:

$$m'_i = m_i - \delta = s_i + \gamma_1 - \delta \quad (9)$$

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

$$r'_i = m'_i + \gamma_2 = s_i + \gamma_1 - \delta + \gamma_2 \quad (10)$$

and hence, calculate the RTT of packet i as:

$$\text{RTT}'_i = r'_i - s_i = \gamma_1 + \gamma_2 - \delta \quad (11)$$

If the adversary knows the actual RTT, it can use (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 1 enables the adversary achieve this because the values in the header of an echo-reply message, Fig. 3(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, *e.g., fping*[9] and *hping*. After receiving the first echo-request, which the adversary ignores, it predicts the values of those two fields for subsequent requests. Because the previous stateless approach requires the adversary to only modify echo reply messages, it might be easier for the adversary to manipulate delays if the stateless approach was used.

M3. *stateful-unreachable:* Similar to the previous method, the adversary decreases RTTs by sending early fake destination-unreachable messages. Timing analysis is, thus, similar to that of M2. From the destination-unreachable header, Fig. 3(b), we see that the ICMP header constitutes no difficulties for the adversary to predict; the TYPE and CODE fields are set to 3, the UNUSED bytes must be set to 0 [38], 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(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 measured RTTs due to extra processing and transmission delays of large packets. The protocol number will be set to 17 for UDP [37]. The following fields are already prone to changes by intermediate systems (*e.g.,* routers) [39]: differentiated services (DSCP), 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(c)). On many implementations, including the *traceroute* utility of GNU, FreeBSD, and Mac OS X, the source and destination port numbers are fixed over a single session, or incremented by one in each UDP segment. Similar to the stateful echo-request utilities (Section 2.2), 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 responses with corresponding UDP segments.

5. ADVERSARIAL MODELS

5.1 Common Capabilities

The adversary is a client that tries to misrepresent its own location by manipulating geolocation. The adversary's ob-

[9]http://fping.org/dist/fping-3.10.tar.gz

Table 2: Capabilities and assumptions of five 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	§
A	✓	✓	✓			(1/3)**c**	herein	6
B	✓		✓			(2/3)**c**	[16]	7
C	✓		✓			(1/3)**c**	herein	7
D	✓	✓	✓	✓		Variable	herein	7
E	✓		✓		✓	Variable	[16]	—

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

jective is to have the technique return a location as close as possible to its intended location, rather than its true location. We assume the LBS uses a delay-based geolocation technique that relies on ICMP messages to measure delays. The adversary can lead the LBS to rely on ICMP-responses simply by filtering all TCP ports, *i.e.,* no TCP response messages are sent on attempted connections to any port.

The adversary has full control over its own machine, but no other machines. It cannot influence the process of calibrating the delay-to-distance mapping function of the landmarks (see Section 2.1), nor infer the calibrated 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 4. We assume the adversary knows the geographic locations of the landmarks (*e.g.,* the information got leaked over time), but does not know the RTT between each landmark and its *intended* location, which is where the adversary wants to appear to be, in terms of the result computed by the geolocation technique.

5.2 Strategies for Modeling Traffic Speed

We model four adversary situations, three of which are newly introduced herein—namely adversaries A, C and D (see Table 2). Adversary B was introduced by Gill *et al.* [16], which we involve in the discussion for comparative assessment. All the modeled adversary situations have similar assumptions (see Section 5.1), except for the factors in Table 2. Note that in contrast to the *achieved ability* in the second column of the table, the third column presents an *assumed* knowledge. Adversary class E in the table was proposed by Gill *et al.* [16], and was assumed to have access to each landmark's calibration function; it then uses this function to directly calculate the traffic propagation speed. We only list this adversary in the table for completeness; we do not explore it further.

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 $\alpha(a, l)$.

Adversary *A*. This adversary does not know both $\alpha(a, l)$ and $\alpha(a', l)$. For example, the landmarks could be taking measures to prevent anyone from *ping*ing their addresses except, perhaps, themselves. However, Adversary A is able to fully manipulate, *i.e.,* increase and decrease RTTs, *e.g.,* by exploiting the properties discussed in Section 4. If it guesses the speed of traffic propagation, it can estimate the RTT at current time because it knows the distances between itself and the landmarks. Katz-Bassett *et al.* [23] found that a speed between (2/9)**c** and (4/9)**c** describes the one-way delay nature of the typically multi-hop Internet routes, where **c** is the speed of light in vacuum. 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 estimates the RTT between its true/intended location and l as:

$$\beta(a, l) = \frac{2 \times dis(a, l)}{(1/3)\mathbf{c}} \tag{12}$$

and

$$\beta(a', l) = \frac{2 \times dis(a', l)}{(1/3)\mathbf{c}} \tag{13}$$

where $dis(a, l)$ and $dis(a', l)$ are the great circle geographic distances [15] between the adversary's true/intended location and landmark l. A great circle is one whose center and radius are those of the Earth. The distance in the numerator is doubled because $\beta(.)$ is a round-trip, rather than one-way, delay. To forge its location from a to a', the adversary sets δ in (5) as:

$$\delta = \beta(a, l) - \beta(a', l) \tag{14}$$

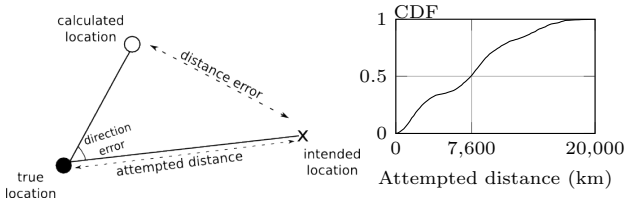
The difference between the adversary's estimated RTT, $\beta(.)$, and the actual, $\alpha(.)$, contributes to the adversary's errors in forging location.

Adversary *B*. Gill *et al.* [16] studied the effect of an adversary that is only able to increase the RTT observed by a measuring party by delaying response messages, exploiting the first property in Table 1. They modeled an adversary that uses the constant (2/3)**c**, which is the speed of light in fiber [32], as an estimate to the traffic propagation speed [16]. We implemented the manipulation tactic of Gill *et al.* [16],[10] which is equivalent to adversary class B in Table 2, to compare it with the new adversary situations modeled herein.

Adversary *C*. Adversary C is assumed the ability of only increasing RTTs, *e.g.,* by exploiting the first property in Table 1. It is similar to adversary class B in that sense. However, C uses (1/3)**c** to model traffic speed. We model this adversary to understand the reasons behind any retrogressions/improvements of B over A, and the impact of the delay-manipulation ability versus the traffic speed parameterization on the accuracy of the forged location.

Adversary *D*. This adversary is assumed to have the ability of full delay manipulation (see Section 4). It also knows the RTT between itself and each landmark $l \in L$, $\alpha(a, l)$, *e.g.,* by *ping*ing l or a nearby server. However, it does not know the RTT between the landmarks and its intended location, $\alpha(a', l)$. To estimate it, D benefits from its knowledge of $\alpha(a, l)$, and calculates the traffic speed between

[10]The results obtained from our implementation closely match those reported by Gill *et al.* [16]; we believe that any dissimilarities arise from differences in the data sets and the experimental environment.



(a) (b)

Figure 4: (a) 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. (b) CDF of 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.

itself and each $l \in L$ as follows:[11]

$$\lambda_l = max\left(\frac{2 \times dis(a,l)}{\alpha(a,l)}, (2/9)\mathbf{c}\right) \quad (15)$$

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 (recall Section 2.1) 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 (15) is applied to avoid the effect of increased circuitousness, or indirectness, and highly varying delay-to-distance ratios occurring with short distances over the Internet [42]. D then estimates the RTT that l should observe at the intended location, a', using the speed λ_l:

$$\beta(a', l) = \frac{2 \times dis(a', l)}{\lambda_l} \quad (16)$$

And finally sets δ in (5) to:

$$\delta = \alpha(a, l) - \beta(a', l) \quad (17)$$

Summary of adversarial categories. We consider Adversary A to be the most realistic one of all five listed in Table 2. It makes use of the vulnerabilities discovered herein, enabling it to increase and decrease the measured RTTs; it also uses a relatively accurate constant to represent the speed of data propagation over the Internet. As such, we evaluate this adversary independently first in the next section. The next three Adversaries in Table 2 (B, C, and D) are introduced to enable us to study the impact of their dissimilar abilities on the location-forging accuracy. Their efficacy is compared to Adversary A later in Section 7.

6. EVALUATING ADVERSARY "A"

The primary evaluation metrics we use are the adversary's distance error and direction error (Fig. 4(a)). 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.,* $\leq 180^\circ$, between the lines passing through both locations and the adversary's true location. This metric is equally important because a large distance error may still be in the same direction, *i.e.,* small direction error (see Fig. 4(a)). Evaluating the direction error independently highlights the cases where the adversary's manipulated location was not only far away from its intended one, but also at a different direction. We used spherical trigonometry to calculate direction errors, where we adopted 6,371 km as an approximation to the Earth's radius [29].

We implemented three prominent delay-based techniques as representatives: GeoPing [31], CBG [20], and SegPoly [12]. We choose more than one technique to study similarities and/or differences in the results, if any. To evaluate the manipulations, we used PlanetLab [9], selecting as many of the available nodes as possible, and in diverse regions in the world. This is important in order to evaluate location-forging attempts from arbitrary locations in the world to other near-by or remote countries. Ground-truth locations of PlanetLab nodes are available on the testbed's website.

In total, experiments were conducted using 144 nodes (Fig. 5), which represented 122 landmarks and 51 adversaries; some nodes acted as both. The delays between the chosen nodes were obtained from the iPlane project [28]. Each client made 50 location-forging attempts, marked by $\times$ in Fig. 5, giving a total of 2,550 attempts.

Figure 4(b) shows a Cumulative Distribution Function (CDF) of the *attempted distances*; 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.

6.1 Manipulation Accuracy

Figure 6(a) shows a CDF of A's distance errors; more than 50 adversarial attempts to manipulate CBG resulted in distance errors <100 km, and two-thirds of all 2,550 manipulations to CBG resulted in <1,700 km. This is less than half the width of the US. For example, if Pandora[12] used CBG to enforce US geographic restriction policies, at least two-thirds of non US-based clients are expected to bypass these restrictions. These adversarial errors arise in part due to inherent inaccuracies of the geolocation methods themselves, making the relatively smaller errors more noteworthy.

The adversary's distance errors were larger while manipulating GeoPing; one-fifth of all manipulations resulted in errors <850 km, and half had errors <1,800 km. The difference between CBG and GeoPing, however, partly stems from CBG being generally more accurate than GeoPing [20]. Using the manipulation strategies of adversary A, such a higher accuracy may unfortunately help adversaries more accurately control the calculated location.

For SegPoly, 80% of manipulation attempts resulted in >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 mapping, 44% of A's manipulations resulted in <2,000 km distance error.

The CDF of the direction error for the three techniques is shown in Fig. 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

[11] $max(x, y)$ returns the larger between x and y.

[12] http://www.pandora.com/

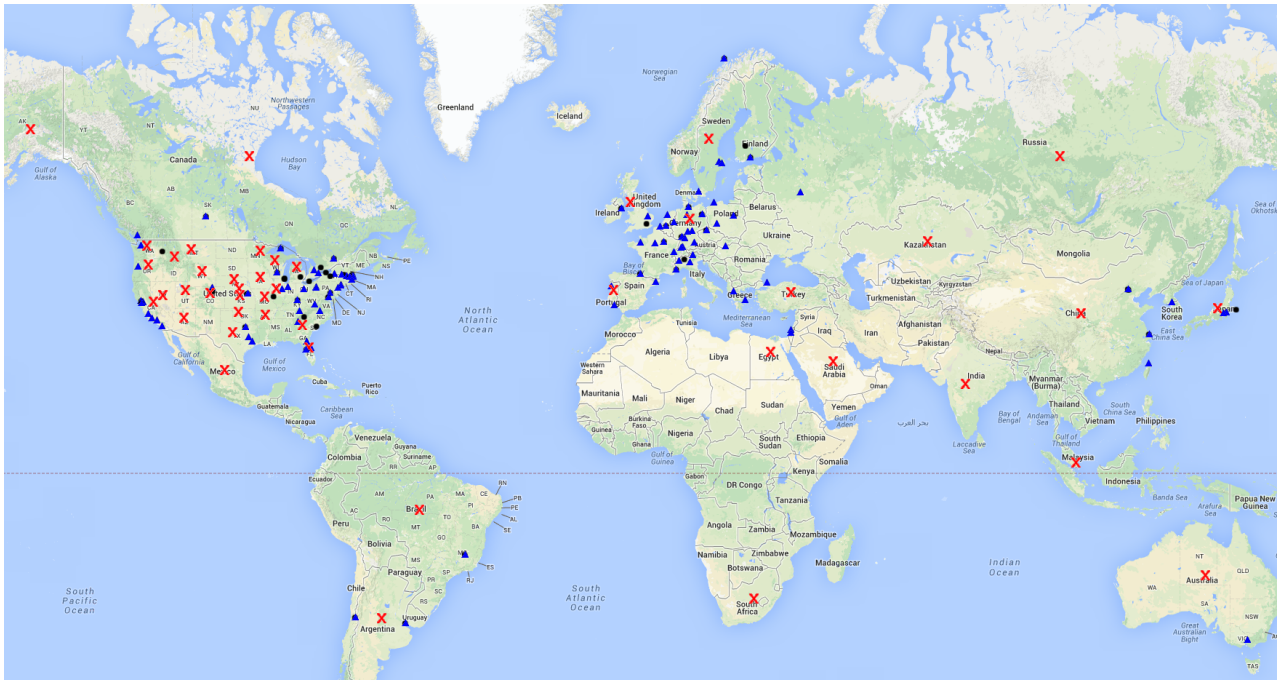


Figure 5: True locations of the 51 modeled adversaries. Each adversary attempted to forge its location to 50 intended locations, for a total of 2,550 modeled attempts to manipulate geolocation. A total of 122 landmarks are used. ▲ = landmarks; • = true locations of adversaries; × = intended locations of adversaries. (best viewed in color) Map data: Google, INEGI.

result, one can think of a US bank restricting credit card transactions to the US, *e.g.,* for fraud prevention [7, 24]. 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 <50° enables them achieve their objective (provided the distance error remains <5,000 km—the country's approximate width). Figure 7 shows the spherical angle at the intersection point, close to the extreme west of Europe, of two lines enclosing the contiguous US.

Next we explore the relationship between A's attempted distance and its distance error. We use the Pearson Correlation Coefficient, which ranges from -1 to +1, such that 0 = no correlation, and ±1 = extreme +/-ve correlation. A powerful adversary should exhibit lower correlation between both variables, meaning that its accuracy does not degrade when its intended location is far away from its true one.

The correlation between the two variables is 0.55 for both CBG and GeoPing. This relatively moderate correlation means that the adversary was able to accurately control the location calculated by the geolocation technique, even when that location is extremely remote. Note that the correlation is positive because manipulations of small attempted distances result in small distance errors.[13]

The correlation for SegPoly is 0.68; it is higher because of the discrepancy between SegPoly's segmented polynomial mapping function and the linear function that A uses. For example, in the third blue cluster of Fig. 2(b) (Section 2.1),

[13]See technical report [1] for further discussion.

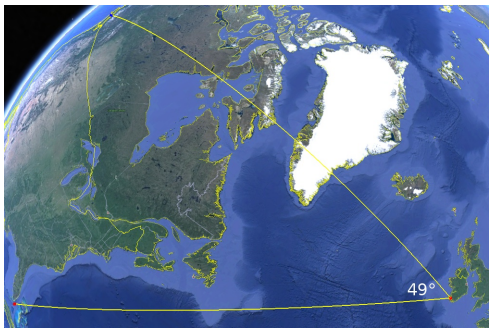


Figure 7: The spherical angle at the intersection point, close to the extreme west of Europe, of two lines enclosing the contiguous US is ∼49°. Map data: Google, SIO, NOAA, U.S. Navy, NGA, GEBCO.

the difference in the mapped distances between both functions at 400 ms is 6,706 km, which is more than double 3,286 km—the value at 100 ms (second cluster in green).

6.2 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.* [16] 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 adversary A. GeoPing generates no intersection regions; we are not immediately aware of a method to precisely detect manipulations against GeoPing.

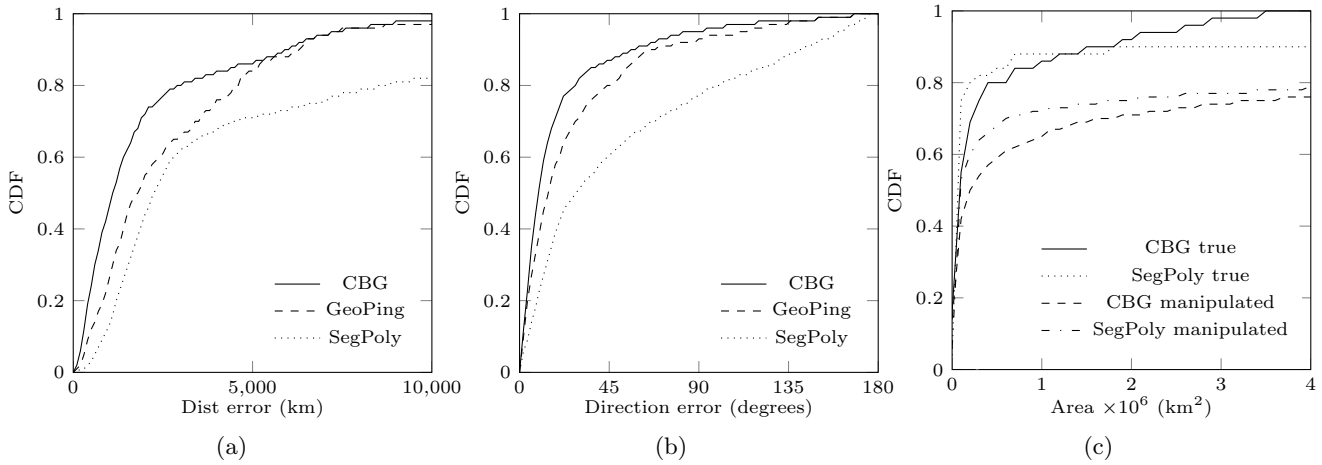


Figure 6: CDFs of (a) distance and (b) direction errors for adversary *A* upon manipulating geolocation; a point (x,y) means a fraction y of all manipulation attempts resulted in error of $\leq x$ (km or deg.). (c) Intersection-region areas while determining the 51 *true* and 2,550 *manipulated* locations; a point (x,y) means a fraction y had areas of $\leq x$ km^2. Higher *manipulated* curves indicate less detectable manipulations.

Figure 6(c) shows a CDF of the intersection-region areas while operating CBG and SegPoly to calculate the forged locations of adversary A, and its true locations. These true locations are calculated from the original delays between the landmarks and the 51 PlanetLab nodes, before changing these delays to model A's manipulations. The speed that adversary A uses, $(1/3)\mathbf{c}$, is slow relative to the average traffic propagation speed [23]. This results in relatively large RTT estimates to the intended location, $\beta(a', l)$ in (13), increasing the distances the landmarks calculate from mapping those RTTs. This explains the large areas depicted by the low curves in Fig. 6(c) while manipulating geolocation.

However, 92% of the areas that CBG calculated while geolocating the true positions 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. As such, detecting manipulations based on the intersection-region areas is not trivial.

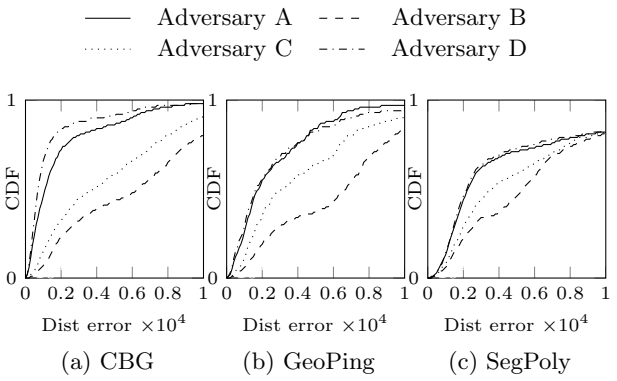


Figure 8: Distance errors (km) for the adversaries in Table 2.

7. ADVERSARIAL MODEL COMPARISON

We compare the adversaries modeled in Table 2 (Section 5). The same delay dataset was used across them to establish a comparable experimental set up.

7.1 Manipulation Accuracy

From Fig. 8, ∼80% of the manipulations of adversary B by Gill *et al.* [16] to the three techniques resulted in distance errors >1,900 km; only 29%, 48% and 59% of A's manipulations to CBG, GeoPing, and SegPoly respectively resulted in errors >1,900 km. This highlights significant adversarial improvements achieved herein compared to adversaries proposed in previous literature [16].

The charts also show that C is more accurate than B; one-quarter of C's manipulations to CBG and GeoPing resulted in <1,500 km distance error and <1,800 km for SegPoly, whereas three-quarters of B's manipulations resulted in >2,200 km error. This highlights the effect of different traffic speed models on location-forging accuracy.

As for adversary D, 66% of its manipulations result in errors <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. For SegPoly, almost unnoticeable distance error improvements were made by adversary D's manipulations over A.

Figure 9 compares the direction errors; 50%, 38% and 53% of adversary B's manipulations to CBG, GeoPing and SegPoly respectively resulted in direction errors <45°, versus 87%, 80% and 60% of 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.

7.2 Manipulation Detection

Figure 10 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 much 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. 10(b) than in Fig. 10(a)). This is because B uses double the traffic modeling speed used by A. Nonetheless, the average distances resulting from B's modeling will be relatively large since B can only increase RTTs. This, combined with B's ability to only increase delays, explains B's 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 area-increasing factors: only increasing RTTs and modeling a slow traffic speed. It is thus the most exposed to being detected based on the intersection region area. Finally, the chart shows that D is least detectable compared to all others.

7.3 Summary

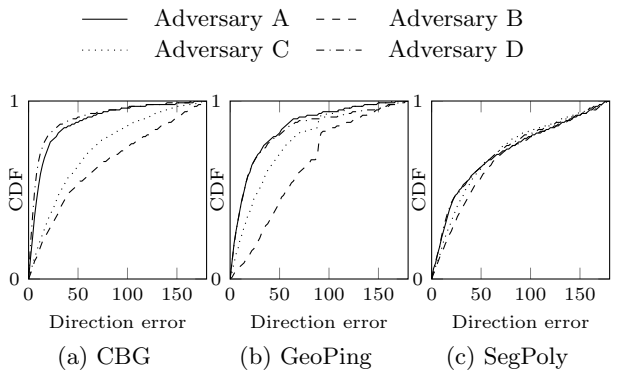
Table 3 summarizes the differences between the four modeled adversaries. For any pair of adversaries X and Y in the table, we can calculate X's achieved percentage reduction to the median distance error over Y as follows:

$$\frac{\text{Median error of } Y \text{ - Median error of } X}{\text{Median error of } Y} \times 100$$

Accordingly, Adversary A achieves 83%, 73% and 58% reductions 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.

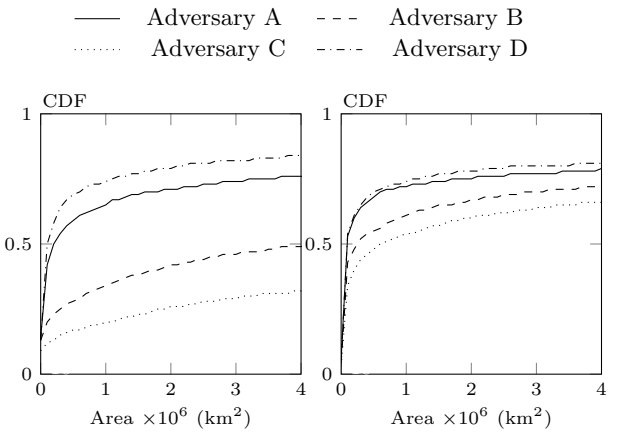
Compared to B, C achieves 40%, 54%, and 33% improvement to the median distance error while manipulating CBG, GeoPing, and SegPoly respectively (see Table 3). The difference between both adversaries is traffic speed modeling, thus a devious strategy alone can increase the adversary's location-forging 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 its location forging accuracy. Furthermore, knowing this information does not provide significant advantages in hiding attempts to manipulate SegPoly since all attempts resulted in very small areas. Thus, the accuracy that SegPoly gains using polynomial regression comes at the cost of lower ability to detect manipulations by the constrained region area.



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

Figure 9: Direction errors (degrees) for the adversaries in Table 2.



(a) CBG (b) SegPoly

Figure 10: CDFs of the intersection-region areas for the adversaries in Table 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.

Finally, adversaries A and D have the lowest correlation between attempted distances and distance errors (Table 3), enabling them to fraudulently relocate themselves at extremely remote locations accurately. Thus, the combined ability of increasing and decreasing delays reduces the impact on distance errors when large distances are attempted.

8. COUNTERMEASURES

We analyzed above the effectiveness of one possible factor, the area of the constrained region, in detecting if the geolocation mechanisms are being manipulated. Although this factor was found to be potentially effective in some cases with an adversary that can only increase RTTs [16], our analysis shows it becomes largely unreliable with an adversary that fully controls RTTs. It is thus crucial to discuss possible countermeasures that specifically aim to preserve the integrity of delay measurements used by a geolocation technique, which is a crucial step prior to geolocation itself.

We stress that the root cause of the vulnerabilities lies not in the ICMP utilities themselves but rather in improperly

Table 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 2; Adversary *B* is similar to that of Gill *et al.* [16]. Smaller values indicate a more powerful adversary.

Geolocation	Dist error (km)				Direction error (deg)				Area $\times 10^6$ (km^2)				Correlation			
method	*A*	*B*	*C*	*D*	*A*	*B*	*C*	*D*	*A*	*B*	*C*	*D*	*A*	*B*	*C*	*D*
CBG	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	1,800	6,700	3,100	1,630	14	58	29	14	–	–	–	–	0.55	0.86	0.64	0.57
SegPoly	2,250	5,350	3,600	2,200	28	41	34	29	<1	<1	<1	<1	0.68	0.85	0.8	0.64

leveraging them for geolocation—a task 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 4, 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(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 [38]. 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 local utilities like *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 [41]. For all practical purposes, ample unpredictability should prevent the adversary from successfully issuing fraudulent echo replies, forcing it to wait for echo requests first.

9. CONCLUSION

This paper shows how to manipulate delay-based geolocation techniques using new strategies that enable an adversary to not only misrepresent its location, but also accurately control the forged location mostly at the country-level.[14] This may defeat location-aware security systems, *e.g.,* a cloud provider violating service level agreements [19], especially given that such geolocation techniques are increasingly being advocated for use in security-aware contexts [8].

We show that *full* delay manipulation, *i.e.,* both increasing and decreasing the measured delays, is possible under certain conditions; these are present in conventional network utilities such as *ping* and *traceroute*, as they fail to check the integrity of the measured RTTs. Unfortunately, delay-based geolocation techniques commonly misuse such utilities for delay-measurements [47, 12]. We also show that better modeling to the traffic propagation speed can enhance the adversary's accuracy in controlling the forged location, even when the adversary is only capable of increasing RTTs; *e.g.,* an adversary that uses $(1/3)\mathbf{c}$ as an estimate to the traffic speed as shown herein is 40% more accurate in forging a CBG-calculated location [20] than the one using $(2/3)\mathbf{c}$ studied in previous literature [16].

The analysis herein provides some useful insights. For example, landmarks in CBG [20] 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 [12] is used, the areas of the constrained region cannot be relied upon for detecting manipulations since they become almost indistinguishable from the constrained regions' areas of legitimate clients. In fact, security-sensitive applications [19] should not rely on the constrained region areas for detecting manipulations because, while geolocating adversaries who can fully manipulate delays, these regions become almost indistinguishable from those of legitimate clients.

For completeness, technically simple countermeasures are discussed as a minor point. However, these add overhead to core ICMP utilities, and thus we expect will likely face deployment resistance since they might be unnecessary for many 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 [20, 24, 4, 12].

Finally, our work shows the importance of not relying on measurement primitives that lack integrity checking. We hope this work raises awareness of the importance of employing integrity-protection in geolocation mechanisms, and encourages further research in the area of secure geolocation.

[14] Related to geolocating cloud data, Peterson *et al.* [33] emphasize: "Of particular interest is establishing data location at a granularity sufficient for placing it within the borders of a particular nation-state."

Acknowledgements

We thank members of the Carleton Computer Security Lab for discussions on this topic. This research is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC)—the second author through a Discovery Grant; the third through a Discovery Grant and as Canada Research Chair in Authentication and Computer Security.

10. REFERENCES

[1] A. Abdou, A. Matrawy, and P. C. van Oorschot. On the Evasion of Delay-Based IP Geolocation. Technical report, Carleton University TR-14-03, June 2014. For further information, see: A. Abdou, *Internet Location Verification: Challenges and Solutions*, PhD thesis, 2015, Carleton University, Canada.

[2] A. 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.

[3] A. Abdou, A. Matrawy, and P. C. van Oorschot. Taxing the Queue: Hindering Middleboxes from Unauthorized Large-Scale Traffic Relaying. *IEEE Commun. Lett.*, 19(1):42–45, 2015.

[4] M. Arif, S. Karunasekera, S. Kulkarni, A. Gunatilaka, and B. Ristic. Internet Host Geolocation Using Maximum Likelihood Estimation Technique. In *AINA*, pages 422–429. IEEE, 2010.

[5] R. Braden. Requirements for Internet Hosts - Communication Layers. RFC 1122 (Internet Standard), Oct. 1989.

[6] J. Burnett. Geographically Restricted Streaming Content and Evasion of Geolocation: the Applicability of the Copyright Anticircumvention Rules. *HeinOnline MTTLR*, 19(2):461, 2012.

[7] M. Casado and M. J. Freedman. Peering Through the Shroud: The Effect of Edge Opacity on IP-based Client Identification. In *NSDI*. USENIX, 2007.

[8] C. Castelluccia, M. A. Kaafar, P. Manils, and D. Perito. Geolocalization of Proxied Services and Its Application to Fast-flux Hidden Servers. In *IMC*, pages 184–189. ACM, 2009.

[9] Chun et al. PlanetLab: An Overlay Testbed for Broad-coverage Services. *ACM SIGCOMM Comput. Commun. Rev.*, 33(3):3–12, 2003.

[10] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. In *SIGCOMM*, pages 15–26. ACM, 2004.

[11] M. L. Damiani, E. Bertino, B. Catania, and P. Perlasca. Geo-rbac: A spatially aware rbac. *ACM Trans. Inf. Syst. Secur.*, 10(1), 2007.

[12] Z. Dong, R. D. Perera, R. Chandramouli, and K. Subbalakshmi. Network measurement based modeling and optimization for IP geolocation. *Elsevier Computer Networks*, 56(1):85–98, 2012.

[13] B. Eriksson, P. Barford, J. Sommers, and R. Nowak. A Learning-Based Approach for IP Geolocation. In *PAM*, pages 171–180. Springer, 2010.

[14] B. Eriksson and M. Crovella. Understanding Geolocation Accuracy using Network Geometry. In *INFOCOM Miniconference*, pages 75–79. IEEE, 2013.

[15] Geoscience Australia. Geodetic Calculation Methods. http://www.ga.gov.au/earth-monitoring/geodesy/geodetic-techniques/calculation-methods.html, 2015.

[16] P. Gill, Y. Ganjali, B. Wong, and D. Lie. Dude, where's that IP? Circumventing measurement-based IP geolocation. In *USENIX Security*. USENIX, 2010.

[17] F. Girlich, M. Rossberg, G. Schaefer, T. Boehme, and J. Schreyer. Bounds for the Security of the Vivaldi Network Coordinate System. In *NetSys*, pages 66–75. IEEE, 2013.

[18] S. Goldberg, D. Xiao, E. Tromer, B. Barak, and J. Rexford. Path-quality Monitoring in the Presence of Adversaries. In *SIGMETRICS*, pages 193–204. ACM, 2008.

[19] M. Gondree and Z. N. Peterson. Geolocation of data in the cloud. In *CODASPY*, pages 25–36. ACM, 2013.

[20] B. Gueye, A. Ziviani, M. Crovella, and S. Fdida. Constraint-based geolocation of Internet hosts. *IEEE/ACM Trans. Netw.*, 14(6):1219–1232, 2006.

[21] D. Hu and C.-L. Wang. GPS-Based Location Extraction and Presence Management for Mobile Instant Messenger. *LNCS Embedded and Ubiquitous Computing*, 4808:309–320, 2007.

[22] M. A. Kaafar, L. Mathy, C. Barakat, K. Salamatian, T. Turletti, and W. Dabbous. Securing Internet Coordinate Embedding Systems. In *SIGCOMM*, pages 61–72. ACM, 2007.

[23] E. Katz-Bassett, J. P. John, A. Krishnamurthy, D. Wetherall, T. Anderson, and Y. Chawathe. Towards IP geolocation using delay and topology measurements. In *IMC*, pages 71–84. ACM, 2006.

[24] S. Laki, P. Mátray, P. Hága, T. Sebók, I. Csabai, and G. Vattay. Spotter: A Model Based Active Geolocation Service. In *INFOCOM*, pages 3173–3181. IEEE, 2011.

[25] R. Landa, R. G. Clegg, J. T. Araújo, E. Mykoniati, D. Griffin, and M. Rio. Measuring the Relationships between Internet Geography and RTT. In *ICCCN*, pages 1–7. IEEE, 2013.

[26] D. Levin, Y. Lee, L. Valenta, Z. Li, V. Lai, C. Lumezanu, N. Spring, and B. Bhattacharjee. Alibi Routing. In *ACM SIGCOMM*, pages 611–624. ACM, 2015.

[27] Li et al. IP-Geolocation Mapping for Moderately Connected Internet Regions. *IEEE Trans. Parallel Distrib. Syst.*, 24(2):381–391, 2013.

[28] Madhyastha et al. iPlane: An Information Plane for Distributed Services. In *OSDI*, pages 367–380. USENIX, 2006.

[29] H. Moritz. Geodetic Reference System 1980. *Springer-Verlag Geodesy*, 74(1):395–405, 2000.

[30] J. A. Muir and P. C. van Oorschot. Internet geolocation: Evasion and counterevasion. *ACM Comput. Surv.*, 42(1):4–26, 2009.

[31] V. N. Padmanabhan and L. Subramanian. An investigation of geographic mapping techniques for Internet hosts. In *SIGCOMM*, pages 173–185. ACM, 2001.

[32] R. Percacci and A. Vespignani. Scale-free behavior of the Internet global performance. *Springer EPJ B—Condensed Matter and Complex Systems*, 32(4):411–414, 2003.

[33] Peterson et al. A position paper on data sovereignty: The importance of geolocating data in the cloud. In *HotCloud*. USENIX, 2011.

[34] I. Poese, S. Uhlig, M. A. Kaafar, B. Donnet, and B. Gueye. IP geolocation databases: unreliable? *ACM SIGCOMM Comput. Commun. Rev.*, 41(2):53–56, 2011.

[35] I. Polakis, S. Volanis, E. Athanasopoulos, and E. P. Markatos. The Man Who Was There: Validating Check-ins in Location-based Services. In *ACSAC*, pages 19–28. ACM, 2013.

[36] A. Popescu. Geolocation API Specification. http://www.w3.org/TR/geolocation-API/, Oct 2013.

[37] J. Postel. User Datagram Protocol. RFC 768 (Internet Standard), Aug. 1980.

[38] J. Postel. Internet Control Message Protocol. RFC 792 (Internet Standard), Sept. 1981.

[39] J. Postel. Internet Protocol. RFC 791 (Internet Standard), 1981.

[40] S. Siwpersad, B. Gueye, and S. Uhlig. Assessing the Geographic Resolution of Exhaustive Tabulation for Geolocating Internet Hosts. In *PAM*, pages 11–20. Springer, 2008.

[41] S. Son and V. Shmatikov. The hitchhiker's guide to DNS cache poisoning. *LNCS Security and Privacy in Communication Networks*, 50:466–483, 2010.

[42] L. Subramanian, V. N. Padmanabhan, and R. H. Katz. Geographic properties of Internet routing. In *ATC*, pages 243–259. USENIX, 2002.

[43] TorrentFreak. Hulu Blocks VPN Users Over Piracy Concerns. http://bit.ly/1839GT2, Apr 2014.

[44] M. Trimble. The Future of Cybertravel: Legal Implications of the Evasion of Geolocation. *HeinOnline Fordham Intell. Prop. Media & Ent. LJ*, 22, 2011.

[45] Y. Wang, D. Burgener, M. Flores, A. Kuzmanovic, and C. Huang. Towards street-level client-independent IP geolocation. In *NSDI*, pages 27–27. USENIX, 2011.

[46] B. Wong, I. Stoyanov, and E. G. Sirer. Octant: a comprehensive framework for the geolocalization of Internet hosts. In *NSDI*, pages 23–23. USENIX, 2007.

[47] I. Youn, B. Mark, and D. Richards. Statistical Geolocation of Internet Hosts. In *ICCCN.*, pages 1–6. IEEE, 2009.

[48] P. A. Zandbergen. Accuracy of iPhone locations: A Comparison of Assisted GPS, WiFi and Cellular Positioning. *Blackwell Transactions in GIS*, 13(s1):5–25, 2009.

[49] 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*, 64(3):685–701, 2013.