Location Verification on the Internet:
Towards Enforcing Location-aware Access Policies Over Internet Clients

AbdelRahman M. Abdou
Carleton University, Ottawa
Email: abdou@sce.carleton.ca

Ashraf Matrawy
Carleton University, Ottawa
Email: amatrawy@sce.carleton.ca

Paul C. van Oorschot
Carleton University, Ottawa
Email: paulv@scs.carleton.ca

***Abstract*—Over the Internet, location-sensitive content/service providers are those that employ location-aware authentication or location-aware access policies in order to prevent fraud, comply with media streaming licencing, regulate online gambling/voting, etc. An adversary can configure its device to fake geolocation information, such as GPS coordinates, and send this information to the location-sensitive provider. IP-address based geolocation is circumvented when the adversary's device employs a non-local IP address, which is easily achievable through third party proxy and Virtual Private Network providers. To address the issue that existing Internet geolocation techniques were not designed with adversaries in mind, we propose *Client Presence Verification* (CPV), a delay-based verification technique designed to verify an assertion about a device's presence inside a prescribed triangular geographic region. CPV does not identify devices by their IP addresses, thus hiding the IP does not evade it. Rather, the device's location is corroborated in a novel way by leveraging geometric properties of triangles, which prevents an adversary from manipulating the delay-sampling process to forge the location. To achieve high accuracy, CPV mitigates path asymmetry by introducing a new method to deduce one-way application-layer delays to/from the adversary's participating device, and mines these delays for evidence supporting/denying the asserted location. We implemented CPV, and conducted real world extensive experimental evaluation on PlanetLab. Our results to date show false reject and false accept rates of 2% and 1.1% respectively.**

I. Introduction

Over the Internet, *Location-Sensitive Providers* (LSPs) are content/service providers that customize their services based on the geographic locations of their *clients* (the software that communicates with the LSP, typically a web-browser). Some LSPs make their services available only to certain geographic regions, such as media streaming [1] (e.g. hulu.com); others restrict certain operations to a specific location, such as online voting (e.g. placespeak.com), online gambling (e.g. ballytech.com), location-based social networking [2] (e.g. foursquare.com), or fraud prevention (e.g. optimal-payments.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 contents based on region [3]. The nature of the provided services may motivate clients to forge their locations to gain unauthorized access.

Various technologies exist for geolocating clients, such as measurement-based techniques [4] or requesting the user's GPS coordinates [5]. Measurement-based geolocation techniques exploit the correlation between Internet delays and geographic distances to *determine* clients' locations [6]. It was shown that those techniques can be evaded through IP address hiding [7], submitting forged information [2], or corrupting the delay-measuring process [8]. Because of the stochastic nature of Internet delays [9], location-verification solutions designed for single-hop wireless networks [10], [11] may not be directly adopted for the Internet.

Considering the mentioned circumvention tactics, verifying locations of Internet clients becomes a challenging problem [12]. To devise a practical verification approach, a number of questions must be answered, such as: *How will IP address-masking be handled? How will the false rejects and false accepts be quantified?* To answer these questions, we present and evaluate *Client Presence Verification* (CPV), a delay-based technique designed to verify a client's geographic location to $\sim 10^4$-$10^5 km^2$ granularity. CPV resists traditional geolocation-circumvention 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 client's favour. CPV takes as an input the client's asserted location, and maps it to a $[0, 1]$ confidence scale which, under appropriate calibration, can then be translated to an accept/reject decision. To mitigate path asymmetry, CPV introduces a new method to deduce one-way delays (OWDs) to/from a potentially dishonest client. To the best of our knowledge, CPV is the first technique that leverages the relationship between delays and distances to *corroborate* (instead of determine) clients' locations, taking into account adversarial environments.

In §II, we provide a summary of the literature that studied delay behavior over the Internet, and its relationship to geographic distances. We discuss the threat model in §III, and explain CPV in §IV. Our empirical evaluation of CPV and a discussion about its security are presented in §V and §VI respectively. Other work related to location verification is discussed in §VII. We conclude in §VIII.

II. Background

Delay characterization between Internet hosts plays a prominent role in numerous applications such as distributed web-caching, server placement in Content Distribution Networks, clock synchronization, overlay Peer-to-Peer networks, geolocation, application-layer mutlicast, and timeout estimations in TCP. Due to its inherent sensitivity, factors affecting

The final version of this paper is published in the proceedings of IEEE Conference on Communications and Network Security (CNS), October 2014. This is the authors' copy for personal use. ©2014 IEEE.

delays between Internet nodes have been well studied including the spanned geographic distances, routing policies, etc.

Delay-based IP geolocation is a broad class of techniques that aim at calculating the geographic location of a client based on the observed delays between the client and a set of landmarks with known locations [13]. Most techniques apply regression analysis to find a function that best describes the relationship between the measured delays and geographic distances [4]. Others use machine learning to achieve the same objective [14]. Multilateration is then used on the pairwise estimated distances between the landmarks and the client to constrain the region where the client is located. Some efforts were proposed to enhance the delay sampling process by removing buffering delays in the routers along the path [15], or utilizing that—even in regions with moderate Internet connectivity—the shortest delay comes from the closest distance [16], [4]. Recent delay-based IP geolocation techniques incur a median error of as low as a few kilometres [17]. To infer distances from delays, the speed at which packets are transmitted over the Internet has been approximated to 4/9 the speed of light in vacuum, a ratio coined as the Speed of the Internet (SOI) [18]. However, the actual speed is affected by several factors such as time of the day, region and the underlying network. Landa *et al.* [19] evaluated the conditional entropy of the delays between two hosts in a study derived from 19 million round-trip time (RTT) measurements all over the globe. They found that the knowledge of the geographic distance between two nodes, their /8 IP prefixes, and their countries scopes down their expected delay to within $\sim 22ms$.

Network topology has been extensively leveraged to devise delay estimators without active probing [20]. Network Coordinates System (or NCS), such as GNP [21], Vivaldi [22], and Meridian [23], 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 space*, i.e. according to its delay to/from them. The Euclidean distance can then be calculated between pairs of nodes to estimate their delays. 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 [24]. Unfortunately, NCSs are vulnerable to an adversary falsifying its coordinates [25], [26].

The aforementioned delay studies reflect solid evidence of a strong correlation between Internet delays and geographic distances [27], which is commonly speculated to be stemming from the improved global network connectivity [13]. We take these studies a step forward, and devise CPV to address the problem of location verification. We now detail the threat model considered in CPV.

III. Threat Model

The adversary is a (human) user that programs its client (software) to evade a geolocation process, thus misrepresenting its location. We assume the adversary is in physical possession of the device running the client, and that the client contacts the LSP through a regular Internet connection. CPV is designed to verify the output of a geolocation technique. However, we assume that the geolocation step prior to the operation of CPV is an unverified assertion, e.g., from the client. CPV is then to verify this assertion. Note that such assumption is one of convenience, and does not pose implications on the operation of CPV if a geolocation technique is otherwise used. On the contrary, allowing the case of a client asserting a location addresses a sophisticated adversary that can evade a geolocation technique.[1]

The adversary could be unaware of the *target location* (the location where it is trying to appear to be at). This is commonly the case when the adversary executes a "trawling" online password-guessing attack (attacking multiple accounts at once) on a location-aware authentication server that restricts a user's login to location(s) associated with the user's account. We also consider an adversary that knows the target location—commonly the case with location-sensitive multimedia providers that announce their broadcasting regions (e.g. hulu.com). We assume that the adversary has full control over its own device, it can install/uninstall any software. We also consider within scope an adversary that uses generic proxies, Virtual Private Networks (VPNs) and/or anonymizers to hide its IP address or to hide any other identifying information that may reveal its true location, as well as an adversary capable of adding artificial delays to Internet Control Message Protocol (ICMP) packets, as outlined by Gill *et al.* [8].

IV. CPV: Client Presence Verification

CPV builds on the established correlation between Internet delays and geographic distances [28] (see §II). The adversarial environment that we consider introduces a new challenge while mapping delays to distances: filtration of noisy delay measurements (e.g. removal of delay outliers) could be exploited by an adversary for its advantage. As such, instead of mapping delays into distances, we escalate an important implication of their correlation: standard geometric theories apply to delays (as they do to distances) with some error margin. By the end of this section, we summarize how CPV manages the delay-sampling process to diminish this error margin without jeopardizing the integrity of the verification process.

In CPV, the client provides evidence, through delay measurements, of its presence inside a triangle determined by three *verifiers*.[2] The size of that triangle is the verification granularity. The verifiers are chosen according to the client's asserted location. The locations of the verifiers should be consistent with the LSP's *Permitted Geographic Regions* (PGRs), which are regions where clients are permitted to receive services/content or carryout location-specific tasks. We assume these verifiers are trustworthy, and that the adversary has absolutely no control over any of them. To successfully enforce the LSP's location-aware access policies, the verifiers must:

1) Be publicly reachable over the Internet.
2) Have a public-private key pair and each verifier must be aware of the public keys of all other verifiers in the set, possibly through a closed Public Key Infrastructure (PKI).

[1] Some geolocation techniques are harder to evade than others [8].

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

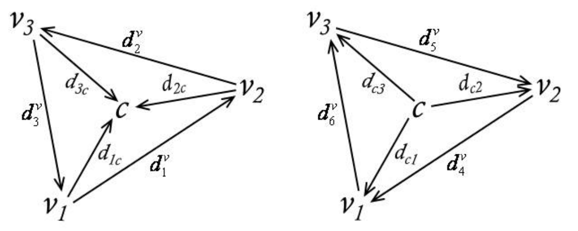


Fig. 1. Notations of OWDs between client c and verifiers v_1, v_2 and v_3.

3) Synchronize their clocks to the nearest ms to measure OWDs (One-Way Delays) among themselves [29].
4) Their convex hull encapsulates the LSP's PGR.

1) Notations and definitions: The union set of all geographic coordinates belonging to all of the LSP's PGRs is denoted $\mathbb{L}$. We refer to the set of verifiers available to the LSP by $\mathbb{V}$. The set of all triangles determined by the verifiers in $\mathbb{V}$ is $T_{\mathbb{V}}$, where $|T_{\mathbb{V}}| = \binom{|\mathbb{V}|}{3}$. For any triangle $\triangle \in T_{\mathbb{V}}$, $V_{\triangle} \subset \mathbb{V}$ is the set of the three verifiers determining $\triangle$. For any geographic location $l = \{x, y\}$, $E_l \subset T_{\mathbb{V}}$ is the set of triangles enclosing l, such that all $\triangle_l \in E_l$ are near equilateral in the network space (see §II), and no $\triangle_l \in E_l$ crosses the LSP's PGR border (PGR border crossing is discussed in §VI-3).

There are three edges joining a client with three verifiers, and three edges joining the three verifiers. Each of the six edges has two OWDs in opposite directions as shown in Figure 1. The set $\mathbf{D}^{\bullet}$ holds an approximation to the smaller OWD at each of the six edges, where the superscript $\bullet$ is the method at which this approximation was obtained. A client and three verifiers make four triangles. The function $valid(\mathbf{D})$ checks if the four triangles, whose side lengths are the six OWDs in $\mathbf{D}$, satisfy the *triangle inequality*. That is, the summation of each two sides is greater than the third for each of the four triangles. The *delay-based area* is the area of a triangle whose side lengths are the delays between its vertices. The function $ar\_v(\mathbf{D})$ calculates the delay-based area of the triangle determined by the three verifiers whose side lengths are the OWDs in $\mathbf{D}$ that belong to the edges between the verifiers. The function $ar\_c(\mathbf{D})$ calculates the summation of the delay-based areas of the three remaining triangles, the ones determined by each pair of verifiers and the client.

2) CPV description: CPV's verification process begins by having the client asserting its location. Assuming the client asserts to be at location $l = \{x, y\}$, 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[3] and the verification process begins. First, the verifiers calculate an approximate to the *smaller* OWD at each of the six edges linking the verifiers with each other and with the client. Second, they use these OWDs to verify the client's asserted location. These two steps are detailed in the following two sections respectively.

3) OWD estimation: If the verifiers settle for sampling the RTT and taking its half as an approximation to the OWD (which is the average of the actual two OWDs), they endure the negative effect of including a spike that occurs in one of the two directions. The verifiers can't ask the *potentially dishonest* client to synchronize its clock with theirs, and exchange timestamps with the client to measure the OWD [32]. So, while approximating the OWD, the verifiers must assume the client may:

- Drop/reject timestamp messages.
- Refrain from appropriately synchronizing its clock with the verifiers.
- Forge the calculated OWD.
- Falsify the timestamp before sending it to the verifiers.

To account for these threats, we devise what we refer to as the *minimum pairs* (mp) method for OWD calculation. To obtain $\mathbf{D}^{mp}$, the three verifiers take turns to create and send a *probing message* to the client. A probing message is a digitally signed timestamp with the most recent system time of the message creator (verifier). The client is required to echo back this message, and forward it to the other two verifiers once it receives the message.[4] This enables the verifiers to obtain nine delay values corresponding to $d_{ic} + d_{cj}$ for all $1 \leq i, j \leq 3$ (see Figure 1), which is possible because their clocks are synchronized. Note that the verifiers can't tell the values of d_{ic} and d_{cj} independently. Algorithm 1 details how the nine values are used to obtain $\mathbf{D}^{mp}$. The notation $S_a(m)$ means message m digitally signed by a; t_a refers to the most recent timestamp according to a's clock; the arrow $A \xrightarrow{m} B$ means A sends message m to B. As shown in lines 11 through 13, the verifiers also use the *average* method, to obtain $\mathbf{D}^{av}$, which will then be used (later in Algorithm 2) as a fallback if the mp method violates the triangle inequality [33]. During this process, the verifiers periodically calculate the OWDs between themselves [34] (line 10).

Because the mp approximation method excludes the larger delay-summation as outlined by lines 15 through 17 of Algorithm 1, it is better in excluding delay-spikes than the av method. For instance, assuming that the probability of occurrence of a spike is equal at the three edges linking the client with the verifiers in both directions, the mp method will surely exclude the occurrence of one spike. The av method will include it. As more spikes occur, the probabilities to include them by the mp method increase, but the probability the mp method includes all spikes never hits 1 except when six spikes occur, one in each direction of the three edges. The av method includes every spike that occurs with a probability of 1 regardless of the number of spikes that occurred.

4) Accept/Reject Decision: We now explain how $\mathbf{D}^{mp}$ and $\mathbf{D}^{av}$ are used to corroborate locations. The verifiers obtain new OWD samples iteratively to account for possible delay instability [35], and calculate $\mathbf{D}^{mp}$ and $\mathbf{D}^{av}$ at each iteration. Their confidence of the truthfulness of the asserted location is the proportion of iterations where the values of $ar\_c(\mathbf{D}^{mp})$ and $ar\_v(\mathbf{D}^{mp})$ almost match. If at some iteration, the delays in $\mathbf{D}^{mp}$ violate the triangle inequality for any of the four triangles determined by the verifiers and the client, the set $\mathbf{D}^{av}$ is used instead for that iteration. $\mathbf{D}^{mp}$ is checked first because

[3]The client may use websockets [30] to connect to the verifiers, as they are a stable means of delay measurement through the browser [31].

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

Algorithm 1: One-Way Delay approximation between a client asserting to be at l and the verifiers in $V_{\triangle_l}$.

```
Output: $\mathbf{D}^{mp}$ and $\mathbf{D}^{av}$
begin
    foreach $v$ in $V_{\triangle_l}$ do
        $v$ captures its current system time $k_1 = t_v$
        $v \xrightarrow{k_1, S_v(k_1)}$ client
        foreach $u$ in $V_{\triangle_l}$ do
            client $\xrightarrow{k_1, S_v(k_1)} u$
            $u$ captures its current system time $k_2 = t_u$
            $u$ validates $S_v(k_1)$
            $u$ calculates $e = k_2 - k_1$
            $u \xrightarrow{e} y \quad \forall y \in V_{\triangle_l} | y \neq u$
    The verifiers measure $d_i^v$ $\forall i = \{1..6\}$ (see Figure 1)
    $\gamma_i^c = (d_{ic} + d_{ci})/2 \;\; \forall i = \{1..3\}$
    $\gamma_i^v = (d_i^v + d_{i+3}^v)/2 \;\; \forall i = \{1..3\}$
    $\mathbf{D}^{av} = \{\gamma_i^c, \gamma_i^v\} \;\; \forall i = \{1..3\}$
    The verifiers solve simultaneously for $\beta_1, \beta_2, \beta_3$:
        $\beta_1 + \beta_2 = min(d_{1c} + d_{c2}, d_{2c} + d_{c1})$
        $\beta_2 + \beta_3 = min(d_{2c} + d_{c3}, d_{3c} + d_{c2})$
        $\beta_3 + \beta_1 = min(d_{3c} + d_{c1}, d_{1c} + d_{c3})$
    $\eta_i^c = min(\beta_i, \gamma_i^c) \;\; \forall i = \{1..3\}$
    $\eta_i^v = min(d_i^v, d_{i+3}^v) \;\; \forall i = \{1..3\}$
    $\mathbf{D}^{mp} = \{\eta_i^c, \eta_i^v\} \;\; \forall i = \{1..3\}$
    return $\mathbf{D}^{mp}$ and $\mathbf{D}^{av}$
```

it is more resilient to delay spikes, as discussed earlier (§IV-3). Algorithm 2 details this process.

It is important to introduce the error tolerance, ϵ (line 7), to account for route circuitousness [36], congested routes, or other factors that contribute to the imperfect delay-distance mapping over the Internet. However, such error tolerance may mislead the verifiers to falsely accept a distant adversary because if the adversary is far enough that the *inner* triangles (those having the client as one of their vertices) are obtuse, those triangles become flattened and their areas decrease. To mitigate this effect, we include the $acceptable(\mathbf{D})$ function (line 7), 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 condition is true for all three OWDs in $\mathbf{D}$ between the client and the verifiers. Finally, the confidence scale (cs) is calculated, and the assertion is rejected if the confidence scale is below some predefined threshold, $\tau_{\triangle_l}$.[5]

The input parameters of Algorithm 2 should be calibrated independently for each $\triangle \in T_{\mathbb{V}}$, where the number of iterations $n_\triangle$ as well as the threshold $\tau_\triangle$ should reflect the variance in delays sampled over a relatively long period of time in the region spanned by $\triangle$. Intuitively, if the regional delay is highly constant over time, no need to take numerous delay samples. The tolerance of the area inequality $\epsilon_\triangle$ should reflect the regional variance in the delay-to-distance ratio. We are

[5]The cs is calculated in every iteration in Algorithm 2 because we study its changing behavior in §V. In practice, it is sufficient to calculate it once at the end of all iterations.

Algorithm 2: Executed by the verifiers in $V_{\triangle_l}$ when a client asserting to be at l connects to them.

```
Input: $n_{\triangle_l}$, $\epsilon_{\triangle_l}$, $\tau_{\triangle_l}$
Output: Accept/Reject client's assertion
begin
    $pass = 0$
    for $i = 1$ to $n_{\triangle_l}$ do
        $\mathbf{D}_i$ = NIL
        Use Algorithm 1 to obtain new OWD samples for $\mathbf{D}^{mp}$ and $\mathbf{D}^{av}$.
        if valid($\mathbf{D}^{mp}$) then $\mathbf{D}_i = \mathbf{D}^{mp}$
        else if valid($\mathbf{D}^{av}$) then $\mathbf{D}_i = \mathbf{D}^{av}$

        if $\mathbf{D}_i \neq$ NIL and $ar\_c(\mathbf{D}_i) - ar\_v(\mathbf{D}_i) \leq \epsilon_{\triangle_l}$ and $acceptable(\mathbf{D}_i)$ then
            $pass = pass + 1$
        $cs_i = pass/i$
    if $cs_{n_{\triangle_l}} < \tau_{\triangle_l}$ then Reject client's assertion
    else Accept client's assertion
```

currently investigating the calibration process, and will present our outcomes in future extension to this work.

In summary, CPV manages potential errors while processing the measured delays to verify a client's location by utilizing the following precautions: (1) two methods are used to approximate the OWDs, with one (the mp) being more resilient to delay spikes; (2) active delay sampling is used with each client, which reflects the most recent delay status in the region [35]; (3) no static delay-to-distance mapping is required; (4) many delay samples are taken instead of one to better converge to the expected delay value [37]; and (5) verification occurs in localized geographic regions to reflect regional delay status [9], span few Autonomous Systems which reduces route circuitousness [28], suffer less triangle inequality violations (TIVs) [33], and exhibit higher positive correlation [19]. Next, we evaluate CPV by checking the existence of regional values for the input parameters of Algorithm 2 that will enable the verifiers to partition clients inside a triangle (legitimates) from outside ones (adversaries).

V. Empirical Evaluation

We use the rates of *false rejects* (FRs) and *false accepts* (FAs) as the evaluation metrics. Assuming a client asserting to be at l, a FR happens if the verifiers in $V_{\triangle_l}$ reject this client's assertion, and the client was geographically present somewhere inside $\triangle_l$. By contrast, a FA happens if the verifiers accept the assertion, and the client was geographically absent from $\triangle_l$.

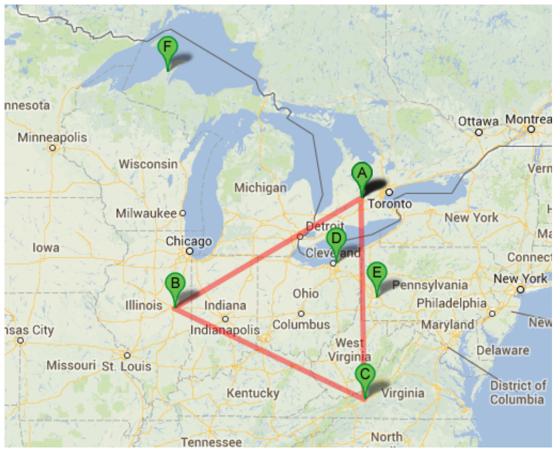
We used ∼80 PlanetLab [38] nodes located within USA and Canada, and identified 34 different sized triangles satisfying the requirements stated in §IV. The areas of these triangles ranged from ∼32,000km^2, close to the size of the state of Maryland, to ∼500,000km^2, close to the size of Spain. We assumed that each triangle is a PGR, i.e. the PGR is a triangular-shaped region that happens to perfectly coincide with the dimensions of the triangle. We considered one triangle at time. For each triangle, all nodes—except the three determining the triangle—acted as clients; all clients asserted to be inside that triangle regardless of whether they were actually inside or not. Combining clients of all triangles together, *legitimates* (clients actually inside) totalled 146 and *adversaries* (clients actually outside) totalled 2,301 giving a total of 2,447 experiment. The verifiers determining each triangle were verifying assertions of all clients concurrently. Knowing the ground truth of legitimates and adversaries with respect to each triangle, our objective is to identify the optimal $\epsilon_\triangle$ and $\tau_\triangle$ values for each of the 34 triangles, and quantify the FRs and FAs at these values.

Experiments were run over the course of a month (April 2013) and at different times of the day. The number of iterations (Algorithm 2) was fixed at $n_\triangle = 600$ for all $\triangle$ in our 34 triangle set to study the behavior of the factors affecting CPV over a relatively long period of time (a total of $\sim$13.3 million delay samples were taken between all nodes). We show in §V-C that fewer iterations are needed to judge a client.

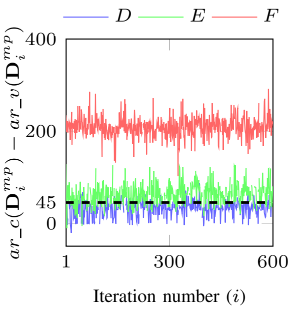
A. *Distinguishing Legitimates from Adversaries*

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 (i.e. inside), the other two were adversaries (i.e. outside). Figure 2(a) shows the location of that triangle and the three clients labelled D, E and F. To study the delay-based area difference (line 7 of Algorithm 2) when areas are calculated by the mp method, we plot the area difference only when $\mathbf{D_i} = \mathbf{D}^{mp}$, i.e. as if line 6 of Algorithm 2 was completely omitted. This is shown in Figure 2(b). There are some few iterations that has no corresponding values for the area difference (visible in high resolution). Those are the ones where the function $valid(\mathbf{D}^{mp})$ returned false. Because F is relatively faraway from the triangle, its average area difference is larger than those of D and E. The smallest recorded area difference for F is approximately 100. Therefore, setting $\epsilon_\triangle$ in the range $\epsilon_\triangle < 100$ makes $pass = 0$ (Algorithm 2) for all iterations, which will result in F's $cs_i = 0$ for all $1 \leq i \leq 600$. Consequently, at $\epsilon_\triangle < 100$, any value of $\tau_\triangle > 0$ will reject F's assertion. E was less than $50km$ away of 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 visible distinction between both nodes. That is, there existed a value for $\epsilon_\triangle$ that enabled the verifiers partition those two nodes despite being geographically collocated.

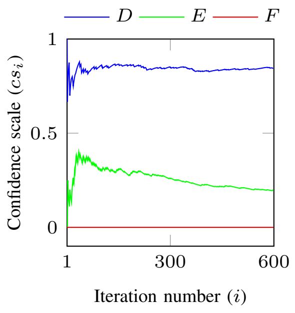
Figure 2(c) shows the progress of the confidence scale (cs) throughout the iterations for the three nodes at $\epsilon_\triangle = 45$. Despite the relatively close values in the area difference between D and E, their confidence scale greatly differs at this value of $\epsilon_\triangle$. The confidence scale at $i = 100$ is 0.86 and 0.3 for D and E respectively. Therefore, after 100 iterations, any value for $\tau_\triangle$ in the interval]0.3,0.86] enables the verifiers detect that D is a legitimate client and that E is an adversary. At $i = 600$, D and E's confidence scale becomes 0.84 and 0.2 respectively, showing no significant change from the 100^{th} iteration. The cs plotted in Figure 2(c) considers $\mathbf{D}^{av}$ at the iterations where the function $\mathbf{D}^{mp}$ returned false (i.e. same behavior as that described by Algorithm 2).



(a) This triangle's area is $\sim$230,000km^2. Clients E and F are outside, whereas D is inside. Map data: Google, INEGI.



(b) At $\epsilon = 45$ (the dashed line), D and E are almost partitioned.



(c) cs at $\epsilon = 45$.

Fig. 2. An example from our experiments showing a triangle and three clients (better viewed in colour).

B. *Proximity to Triangle's Sides*

We study the effect of the proximity of a legitimate client (i.e. inside the triangle) to the sides of its enclosing triangle. For any point $g = \{x, y\}$ inside $\triangle$, the function $awy(\triangle, g)$ (short for *away*) returns the ratio of the distance between g and the side $z^g_\triangle$ to the length of $z^g_\triangle$, where $z^g_\triangle$ is the closest side to g. If $awy(\triangle, g) = 0$, then g lies on one of the three side of $\triangle$. To study the effect of the client's proximity on CPV, we evaluate CPV's behavior when less legitimates with small $awy()$ values are included in the experiments.

Figure 3 shows the rates of FRs and FAs when the only legitimates experimented are those at g, such that $awy(\triangle, g) \geq \lambda$ for all $\triangle$ in our 34 triangles set. The number of remaining legitimates is shown on the same chart.[6] All adversaries are included in the plot regardless of their triangle proximity. As more legitimates get excluded, the effect of the remaining ones on the FRs increases. When remaining clients suffer higher network delays than the average, the FRs oscillate as shown in the plot. Of our chosen PlanetLab nodes, we noticed three suffering exceptionally high delays for unknown reason. Their proximity is in the range $0.002 \leq awy() \leq 0.28$. Those nodes contribute to the oscillation intensity occurring in Figure 3 as λ

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

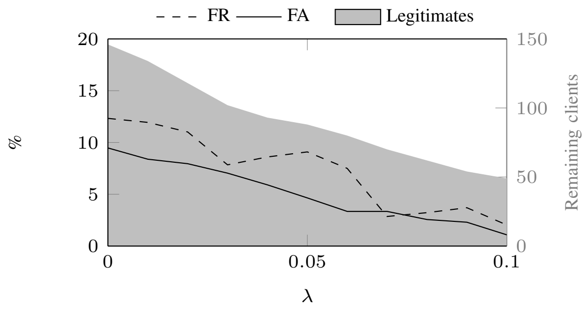


Fig. 3. FRs and FAs when legitimates at location $g = \{x, y\}$ are excluded from the experiments, such that $awy(\triangle, g) < \lambda$. The shaded region shows the number of remaining legitimates. None of the adversaries where excluded regardless of their triangles' proximity.

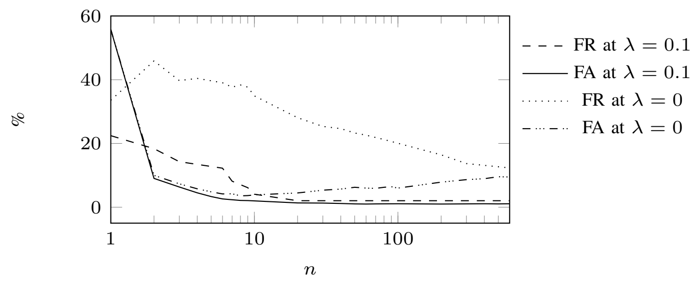


Fig. 4. FRs and FAs when only n iterations are performed.

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. For an assertion l, it is recommended that E_l is to be chosen such that $awy(\triangle_l, l) \geq 0.1$ for all $\triangle_l \in E_l$ (recall that E_l is the set of triangles that can verify clients at l). Despite the unchanged number of legitimates over the spectrum of λ (Figure 3), the FAs improve as λ increases, falling from 9% at $\lambda = 0$ to 1.1% at $\lambda = 0.1$. That is because of the ability to utilize smaller ϵ values.

C. Number of Iterations

The outcome of every iteration contributes equally to the client's confidence scale. However, the weight of this contribution changes whenever a new iteration is performed. Upon making iteration i, the weight of contribution of each iteration becomes $1/i$. Therefore, more iterations lessen the impact of delay outliers sampled during, for example, route congestion. Because the contribution's weight of each iteration is dependent on the number of performed iterations (n), the choice of n affects the final value of the confidence scale and hence, the verifiers' judgement.

Figure 4 shows the number of FRs and FAs at different values of n. For both, FRs and FAs, a generally decreasing behavior is visible as more iterations are performed at both values of $\lambda = 0.1$ and $\lambda = 0$ (see §V-B). The results for $\lambda = 0.1$ are quite sensible: FRs and FAs almost decrease monotonically when more iterations are performed. Performing 2 iterations (at $\lambda = 0.1$) dropped the FAs to $\sim$9% from over 50% when only 1 iteration was performed. Fewer than 10 iterations did not enable the verifiers to realize legitimates appropriately as the FRs were between 6-22%, i.e. no values for $\epsilon_\triangle$ and $\tau_\triangle$ existed to well-partition legitimates and adversaries. However, between 10 and 20 iterations, the FRs and FAs at $\lambda = 0.1$ almost levelled at $\sim$2% and $\sim$1% respectively. Note that our experiments involved a wait-period of 2 seconds between sending probing messages; an iteration took 6 seconds. Therefore, 10-20 iterations took 1-2 minutes. One way to reduce the overall verification time is to decrease this wait-period.

At $\lambda = 0$, the FAs dropped from $\sim$56% when 1 iteration was performed to $\sim$10% when 9 iterations were performed. It then oscillated between $\sim$10% and $\sim$6% when fewer than 100 iterations are performed, climbing steadily to $\sim$8% for the remaining values of n. 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 (i.e. for each $\triangle$) that partition legitimates from adversaries. To accommodate very close legitimates to the triangles' sides, all $\epsilon_\triangle$ were slightly *relaxed* (increased) for each $\triangle$, which resulted in falsely accepting more adversaries. This explains the rising FAs (at $\lambda = 0$) with increasing n. Over the entire range of n, the FRs at $\lambda = 0$ decreased from $\sim$34% at $n = 1$ to $\sim$12% at $n = 600$. Even when legitimates are highly adjacent to their enclosing triangles' sides, large number of iterations improves the ability of finding ϵ and τ values to better partition them from adversaries. This highlights the importance of *iterative* delay-sampling, specially when a triangle is used to corroborate highly-adjacent location assertions to its sides.

VI. Security Discussion

1) Classical Geolocation Attacks: Although submitting false information may mislead some geolocation processes [12], such simple attack does not defeat CPV because the verification process (Algorithm 2) is independent of any information submitted by the client. Experiments in §V show that CPV is able to catch false location assertions (Figure 5(a)) most of the time due to delay-based area mismatch or large delays between the client and the verifiers.

IP geolocation techniques can be circumvented if a client's IP address is concealed using generic *middleboxes* such as proxies, anonymizers, or VPNs [12]. Because probing messages in CPV are sent to the client's application layer, middleboxes that blindly relay application-layer traffic (Figure 5(b)) will relay the probing messages to the client [7]; they do not threaten the integrity of the verification process of CPV. Even a customized middlebox, one that inspects application-layer traffic searching for probing messages, could be mitigated by leveraging a proof-of-work mechanism [39].

Measurement-based geolocation is vulnerable to increased delays between the client and the measuring party, when the adversary delays the echo-reply messages [8]. Such delay-adding attacks can be attempted on CPV by an adversary inserting a delay before forwarding the probing messages as soon as it gets them. Assuming a probing message created by verifier i, not sending this message promptly to verifier j elongates the edges d_{ic} and d_{cj} together, i.e., increases the summation $d_{ic} + d_{cj}$ (see Figure 1 for notations). Note that

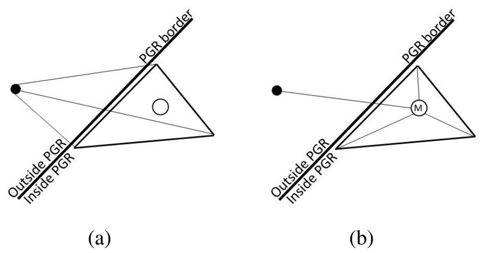


Fig. 5. An adversary asserting a false location (a) without using a middlebox, and (b) using a middlebox at the asserted location. ●=true location; ○=asserted location; M=middlebox; PGR=Permitted Geographic Region.

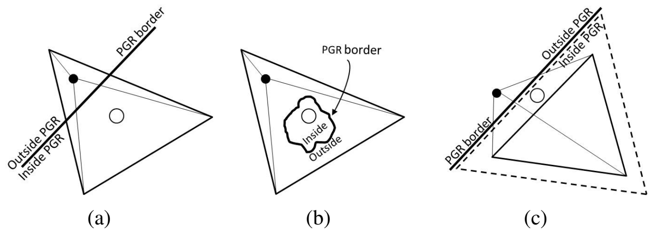


Fig. 6. (a) and (b) are examples of inappropriately-deployed verifiers; (c) is an example of insufficient deployment. ●=true location; ○=asserted location; PGR=Permitted Geographic Region.

because the verifiers, when using the mp method, approximate the smaller OWD at each edge by solving simultaneous equations, delaying the probing messages from a subset of the three verifiers can cause the calculated OWD of an edge to be *less* than the actual.[7] However, the adversary cannot reduce the *summation* of any two edges $d_{ic} + d_{cj}$, as this requires speeding up the traffic propagating in the paths between the adversary and the verifiers [8]. From a geometric perspective, increasing the *summation* of any pairs of edges does not help an adversary outside a triangle to forge its location to inside it. Formally, using the notation $\overline{AB}$ for the length of line segment AB, we have (see proof in the Appendix):

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 increasing the summation $\overline{XP}+\overline{PZ}$, $\overline{XP}+\overline{PY}$ or $\overline{YP}+\overline{PZ}$ without reducing at least one of the other pairs cannot place P inside $\triangle XYZ$.

Finally, as shown in Algorithm 2, CPV holds the number of TIVs (Triangle Inequality Violations) against the client (the condition $\mathbf{D}_i \neq$ NIL in line 7 means $\mathbf{D}_i$ must not violate the triangle inequality to increment $pass$). In conclusion, manipulating the delays does not help the adversary, it divulges the adversary's evasion attempts.

2) Attempts to Evade CPV: To study potential vulnerabilities in CPV, we review steps where the verifiers interact with the adversary.

Connecting to the verifiers. Assuming the adversary's target location (the location where it is trying to appear to be 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 $\triangle_l$.

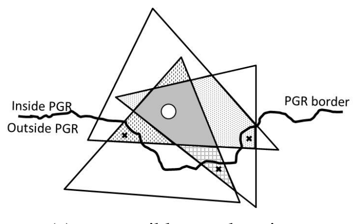
Forwarding the probing message. Attempts to tamper with the timestamp in the probing message will be caught by the receiving verifiers when they validate the signature (step 7 of Algorithm 1). The adversary cannot generate fake probing messages because it will not be possible to sign them using a verifier's private key. Delaying of probing messages is discussed in §VI-1.

3) Poor Verifier Deployment and PGR Border Proximity: Adversaries bordering the PGR (Permitted Geographic Region) may be able to exploit inappropriate or insufficient verifier deployment. Figures 6(a) and 6(b) show examples of inappropriately-deployed verifiers with respect to the PGR, where a triangle crosses the PGR border or encloses the PGR inside it. As shown, a close adversary could be outside the PGR but inside those triangles. Verifying the presence inside the triangle does not ensure the presence inside the PGR in those cases. Figure 6(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 Figure 6(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 Figure 6(c).

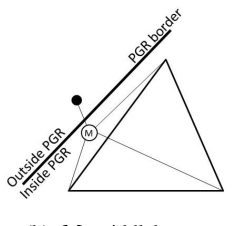
Despite potential vulnerabilities due to inappropriate/insufficient verifier deployment, there are possible countermeasures. To account for inappropriate deployment due to border crossing, additional overlapping triangles should be used to enclose the asserted location as long as a single triangle, or the intersection of multiple triangles, crosses the PGR border. The number of triangles must be such that their intersection region does not cross the PGR border and encloses the asserted location, as shown in Figure 7(a). The presence inside the PGR is then verified only if the verifiers of each triangle accept the assertion. For example, in Figure 7(a), if the adversary's true location was at any of the areas marked with $\times$, 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) must use a measurement-based IP geolocation technique instead of relying on client-dependent geolocation (such as GPS). A bordering adversary will have to evade this geolocation technique prior to bypassing CPV. It would then be challenging for the adversary to precisely target a location that is not covered by any triangle only through delay manipulation [8]. In such case, using a measurement-based IP geolocation technique motivates the adversary to use a middlebox inside the uncovered region of the PGR (Figure 7(b)). However, middleboxes tend to increase delays [7], which helps the verifiers detect the adversary's false assertion.

[7]As an example, solving $a+b=7$, $a+c=8$ and $b+c=9$ 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 $c=7$. *Increasing* $b+c$ resulted in a *smaller* value for a.



(a) ×=possible true locations.



(b) M=middlebox;

Fig. 7. Defences against a bordering adversary exploiting inappropriate or insufficient deployment of verifiers. ●=true location; ○=asserted location; PGR=Permitted Geographic Region.

VII. Related Work

We are not aware of any work in literature that leverages delays for location corroboration. However, delay measurements have been extensively used for location determination [13], [36], [4]. Such difference in objective dictates different evaluation criteria, and hence, it is unclear how our results can be assessed relative to the field of location determination.

Delay-based proximity verification has been well studied in contexts other than the Internet, such as single-hop wireless networks (e.g. Radio-Frequency Identifiers). Proposals included distance bounding protocols [40], ultra-sound based approaches [10], and region bounding through triangulation [11]. The nature of delays over the Internet are different of 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 tough ones (e.g. stochastic queueing delays due to traffic/route uncertainty [9]). Thus, location verification over the Internet is (almost) a different research problem.

Privacy-aware location-proof architectures were proposed in the literature; they enable users to obtain proofs of their presence in a certain location, where a *trusted* access point is available [41], [42], [43], [2]. A user would typically get an access point, trusted by the LSP, to assert the presence of their device within the access point's proximity. This is done by binding a secret and unique identifier of the user to the access point's location. These solutions assume that users are unwilling to disclose this identification credential. We do not make this assumption in our work and hence, these solutions target a different class of applications than the ones we address here. Such applications are when the user's motivation to preserve the confidentiality of their unique identification credentials exceeds their motivation to forge their location. If this is not the case, an adversary may—by the aid of a remote colluding party—send their identification credentials to get them bound to and endorsed by a remote access point, thus forging their location.

VIII. 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 sampled 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.

We presented CPV, 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. CPV mitigates delay spikes injected by the Internet as it iterates the delay-sampling process, and corroborates the client's location based on the *smaller* one-way delay. Accordingly, we devised a novel one-way delay calculation method, which takes into consideration adversarial clients. In CPV, delays are sampled between a client and three verifiers, which enclose the client's unverified location within their convex hull. The verifiers sample the delays over the client's application layer to overcome IP hiding tactics, typically carried out using *middleboxes*. CPV requires no client-side changes, no extra software is needed; the client's current browsing experience is retained as the verification process runs in the browser. Real-world experimentation of CPV results in a 2% false accept and 1% false reject rates. We plan to further inspect the calibration process prior to the operation of CPV, and explore possible venues for enhancing the performance of the location-verification process.

Acknowledgments. We thank the anonymous reviewers whose comments helped improve the paper. We also thank Pat Morin for help with the standard geometrical claims in the appendix. This work was supported by Canada's NSERC through the Canada Research Chairs program (third author) and Discovery Grants (second and third authors).

References

[1] J. Burnett, "Geographically Restricted Streaming Content and Evasion of Geolocation: the Applicability of the Copyright Anticircumvention Rules," *MTTLR*, vol. 19, no. 2, 2013.

[2] Polakis et al., "The Man Who Was There: Validating Check-ins in Location-based Services," in *ACM ACSAC*, 2013.

[3] M. Trimble, "The Future of Cybertravel: Legal Implications of the Evasion of Geolocation," *Fordham IPLJ*, vol. 22, 2011.

[4] Y. Wang, D. Burgener, M. Flores, A. Kuzmanovic, and C. Huang, "Towards street-level client-independent IP geolocation," in *NSDI*, 2011.

[5] D. Hu and C.-L. Wang, "GPS-Based Location Extraction and Presence Management for Mobile Instant Messenger," in *LNCS EUC*, 2007.

[6] Ziviani et al., "Improving the accuracy of measurement-based geographic location of Internet hosts," *Comp. Netw.*, vol. 47, no. 4, 2005.

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

[8] Gill et al., "Dude, where's that IP?: circumventing measurement-based IP geolocation," in *USENIX Security*, 2010.

[9] Dong et al., "Network measurement based modeling and optimization for IP geolocation," *Computer Networks*, vol. 56, no. 1, 2012.

[10] N. Sastry, U. Shankar, and D. Wagner, "Secure Verification of Location Claims," in *2nd ACM WiSe*, 2003.

[11] S. Capkun and J.-P. Hubaux, "Secure positioning of wireless devices with application to sensor networks," in *IEEE INFOCOM*, 2005.

[12] J. A. Muir and P. C. van Oorschot, "Internet geolocation: Evasion and counterevasion," *ACM Comput. Surv.*, vol. 42, no. 1, 2009.

[13] B. Gueye, A. Ziviani, M. Crovella, and S. Fdida, "Constraint-based geolocation of Internet hosts," *Trans. Netw.*, vol. 14, no. 6, 2006.

[14] B. Eriksson, P. Barford, J. Sommers, and R. Nowak, "A Learning-Based Approach for IP Geolocation," in *Springer PAM*, 2010.

[15] B. Gueye, S. Uhlig, A. Ziviani, and S. Fdida, "Leveraging Buffering Delay Estimation for Geolocation of Internet Hosts," in *IFIP-TC6*, 2006.

[16] Li et al., "IP-Geolocation Mapping for Moderately Connected Internet Regions," *IEEE TPDS*, vol. 24, no. 2, 2013.

[17] S. Laki, P. Mátray, P. Hága, T. Sebók, I. Csabai, and G. Vattay, "Spotter: A model based active geolocation service," in *IEEE INFOCOM*, 2011.

[18] Katz-Bassett et al., "Towards IP geolocation using delay and topology measurements," in *ACM IMC*, 2006.

[19] Landa et al., "Measuring the Relationships between Internet Geography and RTT," in *IEEE ICCCN*, 2013.

[20] M. Szymaniak, D. Presotto, G. Pierre, and M. van Steen, "Practical large-scale latency estimation," *Computer Networks*, 2008.

[21] T. Ng and H. Zhang, "Predicting Internet Network Distance with Coordinates-Based Approaches," in *IEEE INFOCOM*, 2002.

[22] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A decentralized network coordinate system," in *CCR*, vol. 34, no. 4. ACM, 2004.

[23] Wong et al., "Meridian: A lightweight network location service without virtual coordinates," in *CCR*, vol. 35, no. 4. ACM, 2005.

[24] B. Donnet, B. Gueye, and M. A. Kaafar, "A survey on network coordinates systems, design, and security," in *CST*, vol. 12, no. 4, 2010.

[25] Girlich et al., "Bounds for the Security of the Vivaldi Network Coordinate System," in *IEEE NetSys*, 2013.

[26] M. A. Kaafar, L. Mathy, T. Turletti, and W. Dabbous, "Real attacks on virtual networks: Vivaldi out of tune," in *ACM LSAD*, 2006.

[27] S.-H. Yook, H. Jeong, and A.-L. Barabási, "Modeling the Internet's large-scale topology," *NAS of the USA*, vol. 99, no. 21, 2002.

[28] L. Subramanian, V. N. Padmanabhan, and R. H. Katz, "Geographic properties of Internet routing," in *USENIX ATC*, 2002.

[29] MacGregor et al., "Precise measurement of one-way delays in an NTPv3 environment," in *Perf. Eva. of Comp. and Tel. Sys.* Citeseer, 2004.

[30] I. Fette and A. Melnikov, "The WebSocket Protocol," RFC 6455 (Proposed Standard), Internet Engineering Task Force, Dec. 2011.

[31] W. Li, R. K. Mok, R. K. Chang, and W. W. Fok, "Appraising the Delay Accuracy in Browser-based Network Measurement," in *IMC*, 2013.

[32] V. Smotlacha, "One-way delay measurement using NTP synchronization," *CESNET*, 2003.

[33] G. Wang, B. Zhang, and T. Ng, "Towards network triangle inequality violation aware distributed systems," in *ACM IMC*, 2007.

[34] S. Shalunov, B. Teitelbaum, A. Karp, J. Boote, and M. Zekauskas, "A One-way Active Measurement Protocol (OWAMP)," RFC 4656, 2006.

[35] Y. Zhang and N. Duffield, "On the Constancy of Internet Path Properties," in *ACM IMW*, 2001.

[36] B. Wong, I. Stoyanov, and E. G. Sirer, "Octant: a comprehensive framework for the geolocalization of Internet hosts," in *NSDI*, 2007.

[37] P. Hsu and H. Robbins, "Complete convergence and the law of large numbers," *NAS of the USA*, vol. 33, no. 2, 1947.

[38] "PlanetLab: An open platform for developing, deploying, and accessing planetary-scale services." Available: http://www.planet-lab.org/

[39] A. M. Abdou, A. Matrawy, and P. van Oorschot, "Taxing the Queue: Hindering Middleboxes from Unauthorized Large-Scale Traffic Relaying," *IEEE Commun. Lett. (to appear; accepted August 5, 2014).*

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

[41] Z. Zhu and G. Cao, "APPLAUS: A Privacy-Preserving Location Proof Updating System for Location-based Services," in *INFOCOM*, 2011.

[42] W. Luo and U. Hengartner, "VeriPlace: A Privacy-Aware Location Proof Architecture," in *18th ACM GIS SIGSPATIAL*, 2010.

[43] F. Olumofin, P. Tysowski, I. Goldberg, and U. Hengartner, "Achieving Efficient Query Privacy for Location Based Services," in *PET*, 2010.

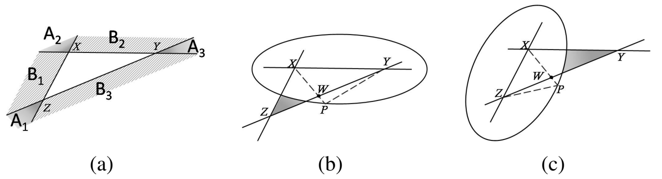


Fig. 8. (a) Regions A and B outside $\triangle XYZ$. (b) and (c) When $P \in B_3$, then $\triangle XYZ \subset \{\bigcirc XY(\overline{XP}+\overline{PY}) \cup \bigcirc XZ(\overline{XP}+\overline{PZ})\}$.

Appendix

We use the notation $\bigcirc XY(k)$ to refer 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 $\overleftrightarrow{XY}$ to refer to the straight line passing by the points X and Y. Consider $\triangle XYZ$ in Figure 8(a). 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 $B = B_1 \cup B_2 \cup B_3$. We split the proof of Claim 1 (§VI) to two parts: when $P \in A$ and when $P \in B$.

For the first part ($P \in A$), lets assume first 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 A_2 and A_3.

For the second part ($P \in B$), lets assume first that $P \in B_3$ as shown in Figures 8(b) and 8(c). If we prove that $\triangle XYZ \subset \{\bigcirc XY(\overline{XP}+\overline{PY}) \cup \bigcirc XZ(\overline{XP}+\overline{PZ})\}$, 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. We split $\triangle XYZ$ int two, $\triangle XYW$ and $\triangle XWZ$, where W is the intersection of line segments XP and YZ. We prove that $\triangle XYW \subset \bigcirc XY(\overline{XP}+\overline{PY})$ as follows (see Figure 8(b)):

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 $\triangle XYW \subset \bigcirc XY(\overline{XP}+\overline{PY})$. ■

For $\triangle XWZ$, the claim $\triangle XWZ \subset \bigcirc XZ(\overline{XP}+\overline{PZ})$ can be analogously proved (Figure 8(c)). 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 $\triangle XYZ \subset \{\bigcirc XY(\overline{XP}+\overline{PY}) \cup \bigcirc YZ(\overline{YP}+\overline{PZ}) \cup \bigcirc XZ(\overline{XP}+\overline{PZ})\}$.[8] This concludes our proof.

[8]We say that $\triangle \subset \bigcirc$ if $\forall p \in \triangle$, $p \in \bigcirc$.