Server Location Verification (SLV) and Server Location Pinning: Augmenting TLS Authentication

AbdelRahman Abdou*, Paul C. van Oorschot†
School of Computer Science, Carleton University
Ottawa, ON, Canada
**abdou@scs.carleton.ca*, †*paulv@scs.carleton.ca*

***Abstract*—We introduce the first known mechanism providing realtime server location verification. Its uses include enhancing server authentication by enabling browsers to automatically interpret server location information. We describe the design of this new measurement-based technique, Server Location Verification (SLV), and evaluate it using PlanetLab. We explain how SLV is compatible with the increasing trends of geographically distributed content dissemination over the Internet, without causing any new interoperability conflicts. Additionally, we introduce the notion of (verifiable) *server location pinning* (conceptually similar to certificate pinning) to support SLV, and evaluate their combined impact using a server-authentication evaluation framework. The results affirm the addition of new security benefits to the existing TLS-based authentication mechanisms. We implement SLV through a location verification service, the simplest version of which requires no server-side changes. We also implement a simple browser extension that interacts seamlessly with the verification infrastructure to obtain realtime server location-verification results.**

1. Introduction

Knowledge of a webserver's verified geographic location can provide greater assurance of the webserver's authenticity, and helps establish the legal jurisdiction under which the server resides, *e.g.,* in case of disputes. The street address of a domain owner/operator is typically different than the location of the physical server hosting its content. If a server's geographic location is verified in realtime, a user-agent (*browser* henceforth) may, *e.g.,* by virtue of a pre-established privacy policy, refrain from proceeding with a connection knowing that the website is hosted from a suspicious location, or a jurisdiction lacking solid privacy laws.

Server authentication on the web is primarily achieved using HTTP over TLS (or HTTPS) and a distributed PKI, albeit with questionable trust semantics. Many known problems in that architecture have been identified [1], [2], raising open-ended questions about the security of the status-quo [3], [4]. This paper reinforces server authentication on the web, by weaving the server's physical location information into current authentication mechanisms. This helps mitigate server impersonation attacks such as phishing [5], pharming [6], and the use of rogue certificates [7] possibly after a certification authority (CA) compromise.

To achieve such location-based webserver authentication, we successfully address several challenges. For example, because of the increasing physical distribution of web content (*e.g.,* cloud computing environments, content distribution networks or CDNs, distributed web-caching and proxy servers, load balancers, and P2P networks), the traditional server-client model where an HTTP session is established entirely between a browser and a single physical server, and content is downloaded only from that server, is becoming less common. Web content is often fetched from several physical/virtual servers, possibly not geographically collocated. How can useful location information be extracted from that context to provide assurance of the domain's authenticity?

Another challenge is the lack of a practical mechanism for realtime server location verification. IP-based location determination is susceptible to location spoofing attacks [8], making it unsuitable for authentication. Offline location verification, *e.g.,* a CA verifying the server's location at the certificate-issuing time and binding the issued certificate to the server's location [9], does not provide location assurance at time of later interaction with the server. Additionally, such a solution would require the domain owner to obtain a certificate for each group of physically collocated webservers, which is impractical in both cost and complexity for large providers that may have thousands of servers around the world (*e.g.,* Akamai [10]). On the other hand, common delay-based IP geolocation schemes work in realtime, but are susceptible to delay manipulation attacks [11], [12]. Prior to the work herein, no known realtime server location verification mechanism that accounts for common adversarial location-forging tactics existed.

To tackle the aforementioned challenges, we introduce Server Location Verification (SLV)—a measurement-based realtime server location verification mechanism. Using a

This paper will appear in ACM Transactions on Privacy and Security (TOPS). *Version: 6th September, 2017.*

network of distributed verifiers[1] over the Internet, the goal of SLV is to verify the geographic location of the first webserver with which the client has a TCP connection. We explain the design of SLV, and implement a simple version that requires no server-side changes, which is thus readily deployable through a browser extension.

We test SLV's efficacy from a location verification standpoint, and analytically evaluate its usage as an additional webserver authentication mechanism. The granularity of location tested in our experiments, using verifier pairs 700 to 1,000km apart, translates to a resolution granularity of circles of radius 350 to 500km, or somewhat finer than the size of an average country (which is about 760,000km^2, or a circle of radius $\sim$500km). We believe this would be a useful choice in practice; finer granularity may well be possible (though unexplored in our present work), but may in some cases be expected to increase SLV's inaccuracies (*e.g.,* false reject rates).

The strong assurance SLV provides to the geographic location of a server adds a new, beneficial dimension to the current notion of webserver authentication. Comparing a server's location to its public key (or certificate), realtime location verification can be seen as analogous to browser certificate validation. As introduced herein, *server location pinning* (*e.g.,* in the browser) can also model key pinning [3] to further enhance server authentication. Browsers can cross-check a server's verified location in a fashion similar to Multipath Probing [13]. A list of physical locations where a server is hosting its content from can be made publicly available for realtime consultation (*cf.* list of active certificates [1]). Existing certificate revocation primitives can be extended to *revoke a location*, *e.g.,* if a data centre was relocated or if content is no longer distributed from a previous mirror.

A domain may legitimately have multiple public keys; primitives such as key pinning and certificate revocation are useful as they attempt to specify which of the validated public keys (or properties related to public key certificates) are authentic, and can accommodate multiple public keys simultaneously. This is comparable to legitimately distributing a domain's content from multiple geographic locations or multiple data centers; adopting analogous primitives can likewise apply to server location, *e.g.,* by pinning all such locations, or actively revoking obsolete locations.

We make the following contributions to enhance server authentication on the web:

- conceptualizing the notion of incorporating physical (geographic) location of a webserver as an additional dimension to strengthen server authentication, in a manner compatible with but independent of current server authentication standards;
- designing and implementing SLV, a new measurement-based algorithm for server location verification, which in its simplest form requires no server-side changes nor human-user interactions, and evaluating its efficacy through pilot experiments using PlanetLab [14];
- augmenting this new mechanism with *browser-based server-location pinning*—a primitive to enable browsers to establish location-based trust semantics over time.

In addition to location-based server authentication, verified server location may provide evidence for services like cloud providers that their servers are in a particular country [15], *e.g.,* with more favourable data privacy laws than others, thus gaining a competitive advantage. Likewise, e-commerce service providers may benefit from assuring their clients that payments are processed in a country they expect or are comfortable with.

The rest of this paper is organized as follows. Section 2 reviews traffic hijacking tactics, characteristics of Internet delay measurements, and the distributed nature of fetching web content. Section 3 defines the threat model. SLV is explained in Section 4, followed by server-location pinning in Section 5. Section 6 empirically tests a prototype implementation of SLV, and evaluates the presented location verification primitives using a server-authentication evaluation framework. Further discussion is given in Section 7. We review related work in Section 8 and conclude in Section 9.

2. Background

This section reviews Internet traffic hijacking mechanisms, the role of timing measurements in location inference, and mechanisms of content distribution over the Internet. Readers familiar with this background can proceed to Section 3.

2.1. Traffic Hijacking: A Network Perspective

Web traffic hijacking is an attack whereby the adversary impersonates the authentic domain, directing users' requests to a machine under the adversary's control rather than one under the control of the domain owner.

Hijacking at different levels. Starting by the user initiating a connection to a domain over the Internet, and moving down the TCP/IP protocol stack, traffic hijacking could be mounted at every point where a new network addressing scheme identifying the intended destination is introduced. Such identifiers include the domain name, IP address, MAC address, and the switch port that a machine is physically connected to. Note that at higher layers of the protocol stack, the notion of an *Internet domain* is abstracted, and can be viewed as a single entity. At lower layers, that entity can become more distributed across multiple physical or virtual machines. References to an authentic/intended webserver or machine in what follows denote any such physical or virtual machines designated by the domain owner to store and offer the domain's services/content over the network.

1. Verifiers could be regular servers deployed using VMs, on cloud infrastructure, or physical servers. Third-parties could provide this in practice, including existing commercial CDN-providers, or as a non-profit organization.

Misleading the user to visit a different (visually similar or disguised) domain name than the intended one is phishing [5]. Because the identifier here differs at the highest addressing scheme, subsequent identifiers, namely the IP address, MAC and port connecting the fraudulent machine to the network, are expected to be different from that of the authentic webserver. Similarly, a pharming attack [16] occurs by misleading the browser-consulted name resolver, which could be at any level in the DNS hierarchical lookup procedure, to resolve the domain name to an IP address assigned to the adversary's machine. The requested domain name is thus equal to the intended one, but the IP address and the remaining identifiers are different from that of the intended machine(s).

ARP spoofing [17] and BGP spoofing [18] are examples of traffic hijacking, where an adversary misleads switches or routers respectively about the network location of the authentic webserver. Both the domain name and IP address of the fraudulent machine are the same as that of the webserver, but the MAC address is different.

Finally, after the switch knows the MAC address of the intended destination, it looks up its MAC table for the physical port number where that machine is plugged. Poisoning the switch's MAC table [17] causes the adversary to deceive the switch into forwarding the data to the physical port where the adversary's machine is connected, thus hijacking traffic intended to the authentic machine. In such a case, the domain name, IP and MAC addresses of the fraudulent machine match those of the authentic one, but the switch port number is different.

This background is used later in the threat model, as summarized in Table 1. Other on-route hijackings are also possible with other addressing schemes, such as in the Spanning Tree Protocol (STP) [19], where switches are assigned `BridgeIDs`, or by other injection mechanisms [20].

Hijacking versus MitM. Once traffic is hijacked, the adversary may itself open another back-end connection to the authentic domain as a regular user, to present the actual user with seemingly authentic responses and thus avoid exposure. The adversary thus becomes a Man-in-the-Middle (MitM) [21], relaying traffic between the user and the intended domain. Our work herein addresses traffic hijacking in general, whether it is a *hijack-and-host* or *hijack-and-relay* (MitM).

The role of TLS. Regardless of where in the network traffic hijacking occurs, HTTPS using TLS with a browser-trusted certification authority (CA) is intended to give assurances about the identity of an authentic domain, aiming to prevent the adversary from *successfully impersonating* the authentic domain/webserver. Such successful impersonation requires not only traffic hijacking, but also defeating TLS protection mechanisms.

To successfully impersonate an HTTPS-enabled domain, the adversary either needs to hijack traffic at the highest addressing level—phishing—or at lower levels, which would also require other actions such as compromising a browser-trusted CA to bind the domain name to the adversary's private key, compromising the authentic domain's private key, or downgrading from HTTPS to HTTP during the connection establishment time, *i.e.,* TLS stripping [22].

While phishing should technically be the easiest to detect since all addresses identifying the adversary's machine differ from the authentic one, it remains effective as it relies on social engineering rather than technical manipulations.

For hijacking traffic at lower levels (as noted above), the Internet's open PKI system is subject to a single point of failure; a single CA compromise could jeopardize the security of the entire system [1]. As such, the system is at most as secure as the weakest CA. Various enhancing primitives have been proposed, such as certificate pinning [3] and Multipath Probing [13], but these aim to strengthen the current PKI system. In contrast, server location verification operates orthogonally as an independent webserver authentication dimension.

Other than a CA compromise, previous literature reports domain operators sharing their private keys among other constituents [23], which corrupts the system's key mechanism of identity assurance. An adversary with access to the domain's private key need not compromise any CA to mount a successful impersonation attack; this is undetectable by primitives such as key pinning.

Next, we review characteristics of Internet delays, and their relationship to geographic locations over the Internet.

2.2. Timing-based Measurements

Literature over the past decade confirms a strong correlation between Internet delays and geographic distances [24], [25], [26], [27]. Although network routes are subject to many conditions that may impede such a correlation, like route circuitousness [28] and delay spikes due to possible network congestion, the strong correlation remains [29]. This is usually attributed to constantly improving network connectivity and bandwidth availability [30].

Many networking applications have leveraged this correlation to achieve accurate IP geolocation over the Internet [31], [32], [33]. A common approach is to derive functions that map delays to distances based on observing various network characteristics (topology, latency, *etc*). The function is then used to map delays measured between multiple vantage points (with known locations) and the target IP address to geographic distances, thus constraining the region where the machine assigned that IP address is physically present. Measurement-based location techniques can achieve high accuracy (*e.g.,* a few hundred meters [25]), for inferring geographic information from network measurements. CPV [34] (see Section 8) was the first known measurement-based technique to verify Internet *client* location assertions, addressing an adversarial client aiming to evade [8] geolocation or manipulate those techniques to its favour [11], [12].

2.3. Fetching Web Content

We review common methods used for dissemination and delivery of web content.

Content Distribution Networks (CDN). A CDN is a network of caching servers used to distribute web content efficiently. CDNs, which have become quite popular, aim to offload the effort of managing and distributing content at large scale from the content owner. Different techniques are used for managing content replication and redirecting browsers to the appropriate CDN surrogate server.

Liang *et al.* [23] note two common practices for browser redirection. The first rewrites the URLs of objects (scripts, images, *etc*) to point to their location on the appropriate CDN server, *e.g.,* using the `src` HTML attribute. For example, to instruct the browser to fetch `image.gif` from the CDN server, the webserver uses:

```
<img src="http://www.cdn-server.com/image.gif">
```

instead of

```
<img src="some-local-directory/image.gif">
```

The second practice resolves the website's domain name to the IP address of the respective CDN server, achieved either by a DNS server under the CDN's administration configured as the authoritative name server for the original website, or by the website's DNS server itself.

In the first practice, the browser establishes HTTP(S)/TCP connections with the original server first, and then with the CDN surrogate server; in the second, the browser only contacts the CDN server without the need to contact the original server.

Caching and proxy servers. A caching server, sitting in the middle of the connection between the browser and the original webserver, terminates the TCP connection intended between client and webserver, and re-initiates another one with the webserver. When the caching server receives an HTTP GET request to a cached object, it sends a conditional GET request to the original server that includes the header line `If-modified-since` specifying the date/version of the cached object. The server either responds with `304 Not Modified`, or with the requested object if the cached version is stale.

Caching servers can be set up at any point along the communication path between client and webserver. For example, the network administrator could set up a caching server, and route network traffic to it to reduce external network usage. An ISP could set-up caching servers to manage network congestion, and the website operator could also set up caching servers to reduce load on the main server.

A non-caching proxy is sometimes also used, *e.g.,* for privacy purposes; the TCP termination hides the client's IP address from the webserver. If the client configures its local machine to use a remote proxy, outbound packets have the proxy's IP address as their destination.

Other schemes. Other content distribution schemes and legitimate browser (re)directs/pointers also exist, such as browser-based ads, collocated load balancers, fast-flux servers (see Section 8), authentication servers and P2P networks. These operate largely similar to the methods reviewed above; we omit further discussion for space reasons.

TABLE 1. LEVELS OF TRAFFIC HIJACKING, AND WHETHER EACH CAN AFFECT A LOCAL AND/OR A GLOBAL SET OF CLIENTS. A CHECK-MARK (✓) MEANS THE RESPECTIVE TRAFFIC HIJACKING IS INCLUDED IN THE THREAT MODEL.

On-route hijacking?	Level	Example hijacking	Local hijacking considered?	Global hijacking considered?	Identifier: Domain Name	Identifier: IP Address	Identifier: MAC Address	Identifier: Switch port
No	*(Human)*	Phishing	✓	✓	□	□	□	□
	Application	Pharming	✓	✓	■	□	□	□
	AS	BGP spoofing		✓	■	■	□	□
Yes	Network	ARP spoofing		✓	★	★	□	□
	Link	MAC table poisoning		✓	★	★	★	□
—	Physical	*no network hijacking*	-	-	■	■	■	■

The *Identifier* assigned to the hijacker's machine (or, if applicable, seen by the client) is either different from (□), or same as (■) that of the intended destination or the next hop machine along the route (★). If the on-route hijacking occurs at the final destination network, then ★ are replaced with ■

3. Threat Model and Assumptions

Adversary's objective. The threat model assumes an adversary aiming to impersonate a webserver by hijacking its traffic. The adversary's typical goals are eavesdropping, or stealing a user's authentication credentials.

Summary of hijacking mechanisms. To define the scope of traffic hijacking mechanisms (see Section 2.1) included in the threat model, Table 1 classifies them by the subset of adversary's machine identifiers that would be equal to that of the authentic machine in each mechanism. Identifiers include the *Domain Name*, *IP Address*, *MAC Address*, and *Switch port number* the machine is connected to (note that Table 1 is also used in Section 6). The table indicates whether the *Identifier* assigned to the hijacker's machine (or, if applicable, seen by the client) is either different from (□), or same as (■) that of the intended destination or the next hop machine along the route (★).

The hijacking level dictates whether an adversary needs to be on-route between the user and the intended destination. On-route hijacking, *e.g.,* ARP spoofing and MAC table poisoning, need not necessarily be mounted at the destination network where the intended machine is connected; it may occur at any intermediate network along the route. If hijacking is mounted at an intermediate network, the destination IP or MAC addresses of the fraudulent machine would be equal to that of the router or switch respectively of the next hop along the route. (Note that the *levels* in Table 1 differ from the five TCP/IP *layers* of the protocol stack.)

On-route network hijacking requires the adversary to locally place itself in one of the intermediate networks, possibly by compromising a host or switch already part of that network. Rows 1-3 in Table 1 do not have that requirement.

Local versus global effect. Note that each hijacking mechanisms in Table 1 may affect either a global or a

local set of clients. For example, an adversary compromising the intended domain's authoritative DNS server itself, can mount pharming attacks on essentially all clients visiting that domain; on the other hand, spoofing local DNS resolutions affects only a local subset of clients. If spoofed BGP prefix announcements propagate to large portions of the Internet, they result in a global effect. Otherwise, their effect is local to the set of affected networks. For on-route hijackings, the closer they are to the network of the intended destination, the more global their effect. For example, ARP spoofing and MAC table poisoning mounted within the local network of the intended destination will affect almost all visiting clients.

Adversarial capabilities assumed. The adversary is assumed the ability of hijacking traffic such that its fraudulent machine's IP address differs from that of the authentic webserver, *i.e.,* Table 1's first two levels. This covers a set of local hijacking, including local phishing and pharming attacks.

For attacks where the fraudulent machine's IP address is equal to that of the authentic destination (*i.e.,* on the AS, Network and Link levels of Table 1), the threat model also includes global hijacking attacks. This includes the 2008 Pakistani Telecom incident [35] and that of China Telecom in 2010 [36]. The threat model also encompasses low-level network hijacks, like ARP spoofing and MAC table poisoning, mounted within the local network of the intended destination as it will affect almost all visiting clients—see Table 1. In summary, SLV is designed to address all MitM attacks conducted by hijacking traffic on any such layer marked by a check-mark (✓) in Table 1.

The threat model excludes hijacking mechanisms that satisfy the following two conditions together (the three cells without ✓ under "*Local hijacking considered?*" in Table 1): (1) they affect only a local subset of clients and (2) they are conducted on a level where the IP address of the fraudulent machine is equal to that of the intended one. An adversary mounting this class of hijacking can bypass location verification because the selected verifiers (those used to verify a location, as explained in Section 4) may not be affected by that locally-impactful traffic hijacking. Thus, they might end up verifying the location of the authentic machine, as identified by the client-submitted IP address, versus the fraudulent one.

The mechanisms presented herein can provide further assurance to a webserver's identity in the absence of HTTPS. The threat model thus assumes an adversary that may or may not compromise the domain's TLS server private key, or issue a fraudulent certificate possibly by compromising a browser-trusted CA. The model also addresses the case of an adversary that can mount TLS stripping attacks [22] to downgrade a connection, and generally other TLS-related attacks. Finally, it is assumed that the verifiers are trusted to carry out and report delay measurements honestly. However, contrary to the PKI trust model where a single CA compromise threatens the entire system, a single verifier compromise does not. It might rather allow an attacker to forge its (malicious) server's location only to a location in the proximity of the compromised verifier, thus potentially allowing the attacker to impersonate servers only in that region. Finally, the entity managing the verifiers (the SLV *Manager*—see Section 4 below) is assumed trusted, so are the user's browser and the webserver itself. Otherwise, if the attacker is already in control of the webserver, the attacker would no longer need to impersonate it by, *e.g.,* breaking authentication, compromising a CA, breaking TLS, or forging geographic locations.

4. Server Location Verification

This section introduces the measurement-based Server Location Verification (SLV) technique. SLV leverages generic delay measurement guidelines from previous literature to infer location information from timing analysis [24], [30], [31]. It is custom-designed to *verify* (not determine) an assertion about a webserver's geographic location.

Location Assertion. There exists no standard mechanism to enable a webserver to assert its geographic location to a browser, *e.g.,* no standard HTTP headers convey that information. For deployability benefits, we rely on IP geolocation databases [37] for location assertions (these assertions will be verified by SLV), thus requiring no server-side changes nor any additional server involvement. These databases may however occasionally have outdated or coarse-grained IP address location information. Webserver provision of an accurate location assertion, *e.g.,* through HTTP headers, would thus be beneficial but comes at the cost of server-side changes. SLV is flexible with respect to the sources where a location assertion is obtained.

Location Verification. SLV is designed to verify the geographic location of the first machine the browser establishes a TCP connection with. This is the machine assigned the IP address resulting from the domain name resolution, thus the verification process may commence in parallel to loading the page (*e.g.,* in a fashion similar to pre-fetching web content). This enables SLV to address all pharming attacks (see Section 3), regardless of where in the hierarchical lookup procedure an adversary may spoof the name resolution; SLV itself does not contact any DNS systems for name resolutions. If the browser receives a spoofed IP address via DNS due to a pharming attack, that IP address is the one passed to SLV for verification. Accordingly, a fraudulent IP address from a local pharming attack would be presented to SLV for location verification.

4.1. Architecture and Algorithm

The system's architecture is shown in Fig. 1. The SLV *Manager* is an independent server acting as an interface between a browser and the *verifiers*. A verifier is a machine, *e.g.,* a virtual private server or a cloud-based server, used to measure Internet delays to a webserver as instructed by the *Manager*. Location verification can occur only in regions where verifiers are deployed; in practice, the entity owning/controlling these verifiers would be similar to, *e.g.,* CDN or cloud providers, and would need to handle location

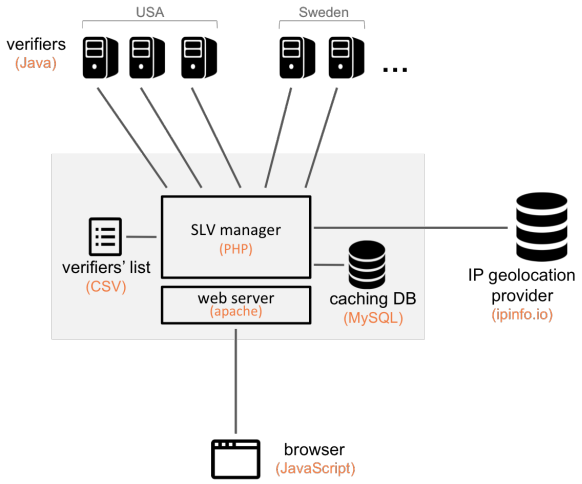

Figure 1. System architecture. Parenthesized terms correspond to implemented prototype (Section 4.2)

assertions in regions with no verifier deployment as an exception. The *Manager* itself runs on top of a webserver, and has access to a list of distributed verifiers and their geographic locations. For efficiency, the *Manager* caches location verification results; entries for a given IP address are cached for a configurable period (*e.g.,* a few hours), before expiring. The *Manager* would typically be deployed on CDNs to bring it closer to clients and verifiers worldwide, thus becomes physically distributed but logically centralized.

Algorithm 1 details the location verification process (see Table 2 for data structures used). When a user visits a website, the browser sends the resolved IP address (the input $adrs$ in Algorithm 1) of the website to the *Manager*. By the *locate()* function in line 4, the *Manager* obtains the best available assertion of the server location (the simplest case may involve using the server's IP address, *e.g.,* using IP-to-location mapping databases [38]). The returned result serves as an unverified assertion of the server's location. The *Manager* then checks for a cached verification result of that address (see the `result` structure in Table 2), and returns it to the browser if the location corresponding to the IP address has not changed since caching time (*e.g.,* the IP address was not assigned to another machine somewhere else).

If no entry was cached, or the cached location is not equal to the newly asserted one, the *Manager* begins location verification by selecting any set of three suitable verifiers (in any order), from all available such sets encompassing the asserted location (line 12). The thee verifiers are selected locally, *e.g.,* with distance ranging 700-1,000km in our experiments (Section 6.1). The order of selecting verifier subsets does not affect the verification result since all encompassing subsets can be chosen until the location is positively verified or no more subsets remain (in which case the asserted location is judged as a false assertion—see below). Each verifier measures the round-trip time (RTT) to the target webserver and to the other two verifiers (line 15). RTTs are not measured using standard ICMP-based tools; this avoids QoS and routing policies at intermediate ASes from artificially delaying or dropping probing messages, and other known problems of such techniques [39]. Instead, the verifiers measure RTTs over the application layer by initiating a TCP connection to the target server like a regular web client, and calculating the RTT from the `SYN–SYNACK` handshake. The verifiers conduct several RTT measurements, and send the smallest back to the *Manager*, which helps exclude delay outliers due to temporary network congestions or routing instabilities. Location verification occurs locally, *i.e.,* using nearby verifiers, so that the measured RTTs exhibit stronger delay-distance correlation [27].

Due to last-mile delays [28], the delay-to-distance ratio is inflated near the edge networks of two communicating parties over the Internet. Such inflation occurs four times while measuring the RTTs between a pair of verifiers and the webserver (*i.e.,* twice between each verifier and the webserver). On the other hand, RTTs are inflated only twice when measured directly between a pair of verifiers. As such, we filter out the inflation factor by subtracting a value, λ, from the three RTT measurements between each verifier and the webserver. In practice, the value of λ may be calibrated in realtime, *e.g.,* as the average RTT between all verifiers in the region and their network gateway. Extensive delay analysis in previous literature found the network edge causes a delay inflation equivalent to ∼5ms [25]. We use this value in our prototype implementation (see Section 4.2).

After subtracting λ, the *Manager* stores the delay information in a two-dimensional array D, in line 15, such that $D[v][i]$ is verifier v's measured RTT between itself and entity i; i is either another verifier or the target webserver.

Geometric Verification. By Thales' theorem [40], an inscribed triangle with one side being the diameter of the prescribing circle is a right-angled triangle, with the diameter its diagonal (Fig. 2). SLV verifies location assertions using this, where times are treated as distances consistently for a given pair of verifiers at a given instant in time. Thus no static delay-to-distance mapping function is required, avoiding any potential inaccuracies introduced by such functions [31]. The asserted location is positively verified if, for any of the three pairs of the selected verifiers, the sum of the squared RTTs between each verifier and the target webserver does not exceed the square of the average RTT between the pair, *i.e.,*

$$(D[v1][ip])^2 + (D[v2][ip])^2 \leq \left(\frac{D[v1][v2] + D[v2][v1]}{2}\right)^2 \quad (1)$$

Figure 3 shows an example webserver encapsulated by a triangle determined by three verifiers. The webserver is inside the two circles whose diameters are delimited by verifiers $[A, B]$ and $[A, C]$. If the measured RTTs of one of those pairs of verifiers support the webserver's presence inside the respective circle, *i.e.,* inequality (1) above holds, this circle becomes the verification granularity. The *Manager* then signs and sends the verification response to the browser,

TABLE 2. STRUCTURE OF EXCHANGED MESSAGES

Struct	Attribute	Data Type	Description
`result`	*ip*	`IP_info`	See `IP_info` below
	veri_passed	`boolean`	Verification result
	region	`circle`	Verification granularity
	when_veri	`timestamp`	Date and time of last verification
`IP_info`	*value*	`string`	IP Address *e.g.,* "1.2.3.4"
	loc	`location`	See `location` below
`circle`	*centre*	`location`	Centre of a circle (See `location` below)
	radius	`double`	Radius of the circle (*e.g.,* in km)
`location`	*lat*	`double`	Latitude
	lon	`double`	Longitude

Algorithm 1: Location verification run by the *Manager*. See Table 2 for data types, and inline for explanation.

Inputs
$\mathbb{C}[.]$: An array of `result` cached at the *Manager*
$adrs$: A `string` of the server's IP address
Output:
res: A structure of verification `result`
begin

```
1  Declare ip of type IP_info
2  Declare res of type result
3  ip.value := adrs
4  ip.loc := locate(adrs)
5  if ip.value exists in ℂ then
6      res := ℂ[ip.value]
7      if res.ip.loc = ip.loc then
8          return res
9  res.ip := ip
10 res.when_veri := local time at the Manager
11 Declare circ of type circle
12 T := the set of triangles geographically
   encompassing ip.loc
13 foreach t in T do
14     V := the three verifiers determining t
15     D := RTT measurements between the
       verifiers in V and ip
16     foreach verifier pair [v1, v2] in V do
17         circ.centre := mid_point(v1, v2)
18         circ.radius := distance(v1, v2)/2
19         if RTTs in D indicate ip inside circ then
20             res.veri_passed := true
21             res.region := circ
22             ℂ := ℂ ∪ res
23             return res
24 res.veri_passed := false
25 res.region := null
26 ℂ := ℂ ∪ res
27 return res
```

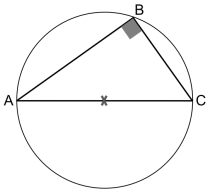

Figure 2. Thales' theorem [40] is used herein for server location verification. It states that for an inscribed triangle (as shown), the angle $\angle ABC = 90°$

along with the centre and radius of the circle (lines 17 and 18), and caches the result for that IP address. If the verification result is negative for all three circles, a new encompassing triangle determined by three different verifiers is selected. That process is repeated (line 13) until (a) the location is positively verified, or (b) the verifiers of all triangles (or a sufficient subset thereof) encompassing the asserted location are exhausted. In our experiments (see Section 6), typically about four triangles suffice for this test. In (b), a negative verification result is returned, and the granularity field ($region$) is set to `null` (line 25). The justification for a positive verification from a single triangle being deemed sufficient to pass location verification is that RTT delays have a lower bound restricted by the spanned geographic distance (data flows in fibre at two-thirds the speed of light [41]).

Caching. For efficiency, the system employs two layers of caching: one at the browser and another at the *Manager*. The former is per browser instance, and is cleared when the browser process terminates. Browser caching is useful as it helps when the user switches between tabs or refreshes a page. A cached entry is as simple as a tuple of <IP address, verification decision>. Note that results are cached by IP address, not domain name. Thus, a page refresh that results in a different non-cached IP address upon resolving its domain causes the browser to resend the new IP address for verification. This makes the browser check the webserver's location if the domain's resolved IP address changes.

The caching at the *Manager* is more persistent. A cached entry is also addressable by IP address, but formatted differently as: <IP address, asserted location, verification date and time, verification result, centre, radius>. The last two entries are the centre and radius of the circle delineating the verification granularity, if the location was positively veri-

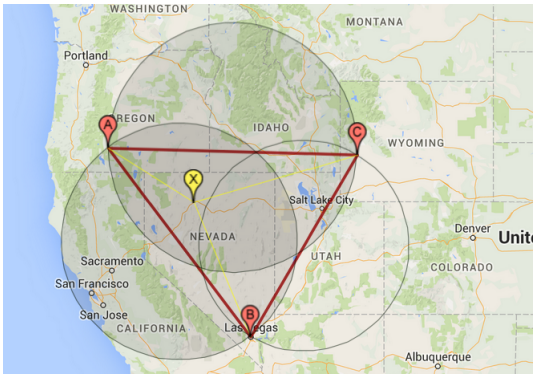

Figure 3. Example using Thale's theorem (map data: Google, INEGI). Each pair of verifiers determines a unique circle whose centre is the midpoint between both verifiers, and radius is half the distance between them. Because server X is geographically inside the circle determined by the pair $[A, B]$, it follows from Thales' theorem (see Fig. 2) that $AX^2 + XB^2 \leq AB^2$. Similarly for the circle determined by $[A, C]$.

fied, and `null` otherwise. Because the *Manager*'s caching is centralized, *i.e.,* relative to the browsers' local caching, a browser is likely to get an instant response from the *Manager* as more verification requests to the *Manager* are made by an increasing number of relying clients.

4.2. Implementation

In our prototype implementation (see Fig. 1), we used Apache as our webserver, PHP for the *Manager*, MySQL for the *Manager*'s caching, and Java for the verifiers. The communication between *Manager* and verifiers uses standard TCP sockets. The *Manager* learns about the m available verifiers using a simple csv-formatted file with m lines and 3 entries per line: the verifier's IP address, its geographic latitude, and longitude. We used $m = 20$ PlanetLab nodes as verifiers in our testing (see Section 6), which were situated in the US. For IP address location lookups, we used the ipinfo[2] DB.

Browser extension. On the browser side, we implemented a Mozilla Firefox extension to submit the website's IP address to the *Manager*, and process a response. We used jQuery to receive verification results from the *Manager* asynchronously. Verification responses are cached locally by the browser, independent of the caching layer on the *Manager*, to avoid re-consulting the *Manager* on page refreshes and tab-switches. The extension also implements server location pinning, as explained below.

5. Pinning of Server Location

We introduce the idea of (verifiable) *server location pinning*, following the idea of certificate pinning [3]; as such, we first quickly review certificate pinning. Certificate pinning is one approach[3] introduced in the server authentication ecosystem to reduce the user's required interpretation of cues and decision making, shifting that responsibility to the browser. A website's certificate is *pinned* (or saved) in the browser, such that future certificates presented by the website are cross-checked against the saved one, with non-matches typically raising suspicion. A domain's certificate can be pinned in the browser or in DNS records [42]. A browser can pin certificates (1) automatically as the user browses the web, (2) when instructed by the server, *e.g.,* through HPKP [43], or (3) when preloaded with pinned certificates. Note that non-DNS based methods (1) and (3) do not require server-side changes, and can thus be immediately deployed through browser extensions.

We put forward the principle of server location pinning. A set of expected server locations (*e.g.,* geo-coordinates) is saved locally (by one of several means explained below) by the user's browser for future cross checking. When a website is then visited and its location is verified (see Section 4), the browser checks if the verified location falls within any of the pre-pinned regions. The action upon a failed check can then be handled automatically by, *e.g.,* a pre-specified policy. Such a policy might handle three possible pinning-validation outcomes, as we explain below: *Critical*, *Suspicious* and *Unsuspicious*. The policy mechanism is outside of the scope herein. However, a simple intuitive policy could instruct the browser to refrain from any connection with login forms or financial transactions in case of *Critical* or *Suspicious* outcomes; in the absence of login forms and financial transactions, the browser drops the connection only in the case of a *Critical* outcome.

Since the geographic locations where a website is hosted from could change frequently for some websites (*e.g.,* due to different content distribution architectures as explained in Section 2.3), server-side cooperation can provide the benefit of dictating which geographic locations should be pinned. This could be, for example, in the form of (1) a publicly queryable set of websites and their locations, which can also provide the benefit of quick location updates; (2) realtime location pinning instructions possibly in the form of HTTP headers created by the webserver itself; and (3) incorporating server location updates into DNS.[4] These examples respectively can be viewed as conceptually analogous to trusted directories or Online Certificate Status Protocol (OCSP) [1], HTTP Public Key Pinning (HPKP) [43], and DNS-Based Authentication of Named Entities (DANE) [42] in the current server authentication standards.

Location Pinning Algorithm. Locations are pinned as an array $\mathbb{P}$ of the data structure shown in Table 3. The array is referenced by the domain name ($name$). The attribute ips is an array of IP addresses that $name$ (*i.e.,* domain name) has previously resolved to. $regs$ is an array of geographic regions, each described as a centre and radius of a circle, where the domain name was verified to be hosted from. $rmax$ is the upper limit on the number of allowed server locations (*e.g.,* dictated by the domain operator).

2. http://ipinfo.io/
3. Other items could be pinned, such as the public key value.
4. DNS location records (`LOC`) were initially proposed as a means of disseminating IP-address location information [44].

TABLE 3. STRUCTURE OF PINNED LOCATIONS FOR A DOMAIN (SEE TABLE 2)

Attribute	Data Type	Description
$name$	`string`	The domain name
$ips[.]$	`IP_info`	An array of the domain's saved IP addresses
$ver\_regs[.]$	`circle`	An array of the domain's verified regions
$rmax$	`Integer`	Upper limit on the number of allowed server locations
$when\_veri$	`timestamp`	Date and time of last verification
$when\_pin$	`timestamp`	Date and time of pinning

Algorithm 2 details the server location pinning mechanism. When a location verification response is received from the SLV *Manager* (see Section 4), the browser first searches $\mathbb{P}$ for a previously pinned location entry for the corresponding domain name. If none is found (line 23), the browser either pins the domain's location if it was verified, or reacts to a *Suspicious* outcome as specified by the policy if location verification fails. If a pinned location is found but location verification has failed, it is a *Critical* outcome.

Assuming the browser had previously pinned server locations for that domain, and that the domain's IP address is verified (line 7), the browser checks if the domain's asserted location falls within any of the pinned regions for that domain. If it does, the browser either updates the IP address's corresponding stored geographic locations, or if the IP address was seen for the first time for that domain, adds it to the array of IP addresses corresponding to the domain name. If the asserted location does not fall within any of the pinned locations (but was positively verified), the browser adds it to the pinned domain as a new region only if more regions are allowed for that domain (line 20). Otherwise, the new asserted location, despite being successfully verified, is classified as a *Critical* outcome.

Note this algorithm does not place any restrictions on the number of IP addresses allowed per domain. The restriction is only on the number of different geographic regions ($rmax$) where content is initially provided. In practice, the value $rmax$ might be set and announced by each domain operator.

Caching versus Pinning. The principle of location pinning introduced in this section is different from caching the results for efficiency (see Section 4 for caching). A verification result is always cached whether it is positive or negative, whereas a domain's location is only pinned when positively verified. Additionally, a disagreement between a previously cached result and a new one is not acted upon by the browser, but a negatively verified location that was previously pinned must be interpreted by the browser, and is indicative of a potentially critical browser connection. Also a cached entry automatically expires after a certain predefined time window, but a pinned location is typically revoked. Since caching is performed for efficiency, it occurs in two levels in the system, browser and *Manager*, whereas location pining is a browser-only capability to enable automatic interpretation of changes in domain name location verification results. As explained by Algorithm 2, location pins are region oriented, whereas a cached location verification result is not. Finally, cached entries are indexed by the *IP address*, whereas a pinned server location is indexed by the server's domain name, indicating that the *domain* resolved to a machine whose location was positively verified. In summary, pinning is performed for security, where the browser processes results and takes actions to protect the user when necessary (*e.g.,* a verification failure for a previously pinned location), whereas caching is a *dummy* operation involving no browser interpretations of any results, and is only performed for efficiency.

6. Evaluation

We evaluate SLV in two stages. First, we establish the conceptual validity of the measurement-based location verification technique itself from a networking perspective, by attempting to verify websites with known locations using a prototype implementation. Second, we evaluate the benefits of combining this with server location pinning to augment server authentication mechanisms.

6.1. Evaluating Measurement-based SLV

Our pilot testing uses PlanetLab [14], employing as verifiers 20 testbed nodes distributed in North America. We measure the false reject (FR) and false accept (FA) rates when using the described SLV approach to verify server location assertions. As such, we test SLV by verifying locations of servers in which we have available ground truth about their geographic locations. We followed the assumption that university websites are hosted on-campus [25], thus we can use their posted street addresses as our approximation for their webserver locations. See below on verifying this assumption. For each location assertion (*i.e.,* university website), triangles were selected such that the distance between each pair of verifiers is 700-1,000km, giving a granularity equivalent to a circle of radius 350-500km.[5]

Note that regardless of the content-distribution scheme employed in practice by a website (*cf.* Section 2.3), a browser always downloads content from one or more physical or virtual server(s). We focus here on SLV's feasibility to provide measurement-based location assurance to the server currently being contacted (as explained in Section 4),

5. Accounting for 195 countries in the world, the average country size is $\sim$760,000km^2 (approximately the size of Turkey). The tested granularity is equivalent to 400,000-790,000km^2 giving approximately a country-size average.

Algorithm 2: Server location pinning in the browser.

Inputs
$\mathbb{P}[.]$: An array of pinned domain locations
d: Domain in question
r: Verification `result` of Algorithm 1

Goal
To enable the browser to establish location-based trust semantics over time.

begin

```
1  outcome := Unsuspicious
2  if d exists in ℙ then
3      pin := ℙ[d]
4      if r.veri_passed = false then
5          outcome := Critical
6      else
7          found := false
8          foreach region in pin.regs do
9              l := dist(r.ip.loc, region.centre)
10             if l ≤ region.radius then
11                 found := true
12                 if r.ip.value exists in pin.ips
                   then
13                     pin.ips[r.ip.value].loc :=
                       r.ip.loc
14                 else
15                     pin.ips := pin.ips ∪ r.ip
16                 pin.when_veri := r.when_veri
17                 break
18         if found = false then
19             if size of pins.regs < pin.rmax
               then
20                 pin.ver_regs :=
                     pin.ver_regs ∪ r.ver_regs
21             else
22                 outcome := Critical
23 else if r.veri_passed = true then
24     Declare x as a pinning struct (see Table 3)
25     x.name := d
26     x.ips := pin.ips ∪ r.ip
27     x.ver_regs := pin.ver_regs ∪ r.ver_regs
28     x.when_veri := r.ver
29     x.when_pin :=local time at the browser
30     ℙ := ℙ ∪ x
31 else
32     outcome := Suspicious
33 return outcome
```

TABLE 4. SUMMARY OF PILOT EVALUATION RESULTS.

	Total	Accepted	Rejected	FR/FA
True assertions	83	81	2	2.4% FR*
False assertions	100	0	100	0% FA

*See inline for an explanation of the results

whether that server is standalone or part of a larger distribution network (*e.g.,* a CDN). University servers with known ground-truth locations thus suffice for our evaluation purpose.

6.1.1. False Rejects. We randomly selected 94 university/-college websites for testing, and excluded 11 of these that simple filtering found to be hosted by a cloud or a CDN, lacking ground truth knowledge of their true geographic locations. The filtering involved looking up from public registries, the AS from which the domain is reachable. As Table 4 shows, two of the 83 remaining domains were falsely rejected; these both fell in a region deficient in verifiers. In exploring this, we found one involved verifier (PlanetLab node) which contributed to both FRs was extremely slow, including in responsiveness to running commands/processes, thus presumably suffering technical problems. From this initial study, we know how to reduce the FR rate below 2.4% (*i.e.,* by testing the reliability of verifier nodes in advance), but reporting this initial result highlights the importance of a responsive and sufficient verification infrastructure.

6.1.2. False Accepts. To evaluate FAs, the SLV *Manager* was manually configured to select triangles not encapsulating the asserted locations, and far enough away (not within the same country) to have the asserted location outside the three circles determined by each pair of verifiers, as explained in Section 4. The expectation is that the false location assertion will be correctly *rejected*. That is, an attacker hosting a malicious server (*e.g.,* a phishing website, pharming, or a machine hijacking traffic through BGP) in La Paz, Bolivia for example, and falsely asserting (possibly through delay manipulation, or the use of proxy servers) that the server's location is in Cincinnati, Ohio, won't be falsely accepted by SLV's verifiers encapsulating that fraudulently asserted location.

Four triangles were randomly chosen for each tested domain. The verifiers determining each of four triangles must reject presence inside the respective circles for a reject decision to result; the parameter four was empirically determined, and is subject to adjustment. Again, domains were chosen randomly from among those for which we had ground truth knowledge of webserver location.

One hundred domains were chosen in the following manner: 40 in Europe, 20 in eastern Asia, 20 in Latin America, and 20 in Oceania. As summarized in Table 4, none of the tested domains was falsely accepted. The false accept rate of 0% is not intended to claim perfection, but rather is an artifact of limited preliminary testing.

6.2. Evaluation with Server Location Pinning

Table 5 uses a webserver-authentication evaluation framework almost identical (in columns) to that developed by Clark *et al.* [1]. The column headers show the evaluation criteria, and the rows are the enhancement primitives.

Baseline HTTPS. To identify new benefits relative to the standard HTTPS defense mechanism, we first evaluate HTTPS itself as a baseline for comparison. Row 0 was not required in the work of Clark *et al.* [1], which specifically evaluated SSL/TLS-enhancements. In our row 0, all comparative evaluation criteria, such as *No New Trusted Entity* and *No Extra Third Party*, are relative to regular HTTP (non-HTTPS); the other rows are rated relative to row 0.

HTTPS provides the first three security properties in Table 5 but only partially, in light of the recent community awareness of critical HTTPS weaknesses and real-world attacks [7], [45], [46]. The attack surface includes CA compromise, TLS stripping, implementation vulnerabilities, misconfiguration, and reliance on users to make security decisions.

Basic HTTPS relies on trusting CAs and signed certificates for server authentication, and thus lacks bullets at *No New Trusted Entity* and *No New Auth'n Tokens*. While not introducing new traceability avenues, it does not *reduce* traceability because revocation methods, such as OCSP responders, are still required for revocation. It requires servers to obtain certificates, thus lacks *No Server-Side Changes*. Finally, HTTPS lacks all three usability properties relative to HTTP.

SLV. For the security properties, SLV (row 1) provides both the benefit of detecting global MitM attacks, regardless of how the adversary hijacks traffic (recall Section 2.1), and of detecting a subset of local hijacks (column 2 in Table 5), including local pharming attacks. As noted in Section 3, if a local ARP spoofing or local BGP prefix hijacking occurs, the selected verifiers will not be affected and will thus attempt to verify the location of a machine that is different from the fraudulent one communicating with the client.

Leaking client credentials (column 3) and TLS stripping (column 5) require the adversary to conduct traffic hijacking first, and SLV provides partial protection if that hijacking was locally conducted (column 2). Thus, SLV offers a partial benefit (○) in both situations. The *Affirms POST-to-HTTPS* benefit prevents submitting POST requests over HTTP; SLV does not provide that benefit.

In terms of the impact to HTTPS, no new authentication tokens are introduced by SLV since the verification results are sent to requesting clients automatically and in realtime. For deployability, SLV with assertions based on IP-address to location lookup tables requires no server-side changes, and can be deployed without DNSSEC. It provides the benefit of *Internet Scalable* because the location verification process is fully automatable (unlike, *e.g.,* certificate preloading, where requests are manually reviewed [3]). However, the benefit is graded as only partial because the required verification infrastructure, such as the verifiers, grows as the need for location verification increases. Finally, SLV provides the benefit of signalling the status completely because all browsed servers' locations are sent to the *Manager* (see Section 4) for verification, *i.e.,* server participation is not optional.

Note that the nature of communication between a browser and the SLV *Manager* is similar to that of Multipath Probing [13], and thus their beneficial properties (row 6) are similar (*cf.* [1]).

Location Pinning Alternatives. Server location pinning is conceptually similar to key/certificate pinning, and thus have similar advantages and weaknesses (*e.g.,* scalability). Any of the four pining methodologies (see rows 2-5 in Table 5) could be adopted for the server's location. For example, just as public keys could be pinned with DNS records, servers' geographic locations can be likewise. In fact, the DNSLOC records were proposed experimentally in 1996 [44] for non-adversarial location assertion purposes. In conclusion, evaluation outcomes of location pinning primitives are similar to those of key pinning (*cf.* [3]).

Note that the benefits *No New Entity*, *No New Traceability*, *No New Authentication Tokens*, and *No Extra Third Party* are provided by (some of) the location pinning primitives, but not SLV (row 1). For example, for *No New Traceability*, the pinning process itself, including checking verification results against already pinned locations, does not introduce new traceability (*e.g.,* if the location verification results were already cached). An analogous argument applies to the remaining three benefits.

List of Expected Locations. Domain owners can maintain a publicly accessible list of geographic locations where a client should expect the server offering their content. This is analogous to maintaining a *list of active certificates* [1] (*e.g.,* Certificate Transparency [4]) to facilitate revocation simply by removing the revoked certificate from the list, and may thus aid in location revocation.

7. Discussion

When TLS is not available for a domain, *e.g.,* if not supported or because of a TLS stripping attack, the location verification mechanism presented herein offers an independent means for detecting fraudulent server authentication. Nonetheless, verified location information is best combined with TLS, to provide an additional authentication dimension. This becomes especially useful in cases where certificate validation is suspicious, *e.g.,* when the browser is presented a self-signed certificate, an insecure/outdated cipher suite, or an HTTPS page with mixed content [23].

The tree diagram in Fig. 4 shows how location-based server authentication can complement TLS to reduce the likelihood of successful attacks. The dashed lines indicate parts contributed by the presented primitives; they highlight scenarios where traffic hijacking and/or MitM may go undetected without SLV, but would instead be mitigated if SLV is used.

From a user's perspective, we believe that, even without requiring any new user actions, it can be useful for some

TABLE 5. EVALUATION OF LOCATION-BASED PRIMITIVES TO AUGMENT WEBSERVER AUTHENTICATION. ● DENOTES THE PRIMITIVE PROVIDES THE CORRESPONDING BENEFIT (COLUMN); ○ DENOTES PARTIAL BENEFIT; AN EMPTY CELL DENOTES ABSENCE OF BENEFIT. THE SHADED ROW (INDEXED 0), WHICH IS ITSELF RATED COMPARED TO REGULAR HTTP, SERVES AS THE BASELINE FOR COMPARATIVELY ASSESSING IMPROVEMENTS OR RETROGRESSION OF THE NEW PRIMITIVES DETAILED HEREIN (ROWS 1-6).

		Security Properties Offered							*Beneficial Properties*								
		A				B	C		*Security/Priv*			*Deployability*				*Usability*		
	Primitive	*Detects MitM*	*Detects Local MitM*	*Protects Client Credential*	*Updatable Pins*	*Detects TLS Stripping*	*Affirms POST-to-HTTPS*	*Responsive Revocation*	*Intermediate CAs Visible*	*No New Trusted Entity*	*No New Traceability*	*Reduces Traceability*	*No New Auth'n Tokens*	*No Server-Side Changes*	*Deployable without DNSSEC*	*No Extra Third Party*	*Internet Scalable*	*No False-Rejects*	*Status Signalled Completely*	*No New User Decisions*
0	*Standard HTTPS*	○	○	○						●				●	●	●			
1	SLV without location pinning	●	○	○		○							●	●		○		●	
2	Server loc. pinning (Client History)	○	○	○					●	●		●	●	●	●	●			
3	Server loc. pinning (Server)	○	○	○					●	●			●	●	●		●		●
4	Server loc. pinning (Preloaded)	●	●	●	●				○	●		●	○	●	●		●	○	●
5	Server loc. pinning (DNS)	●	●	●	●				○	●	●		○		●	●	●	○	●
6	List of expected locations						●				●		●	●		●	●	●	●

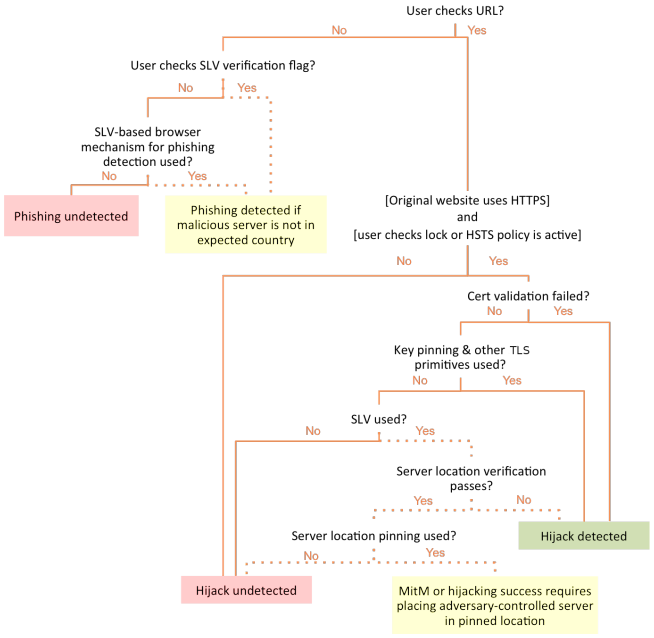

Figure 4. Decision tree for detection of traffic hijacking attacks. As explained in Section 3, from the server's perspective, phishing is a class of traffic hijacking. Dashed lines indicate attacks detected only by the new mechanisms presented herein.

users to see in which country a server is located [47]—whether this information is verified by SLV or just asserted by any browser plugin, *e.g.,* flagfox (see Section 8). In a phishing attack, if the adversary obtains a valid certificate for a spoofed domain, standard visual browser cues will show green locks, and positively assuring symbols [48]. A country's flag or a displayed world map will however differ from expectations (*i.e.,* when the adversary's fraudulent machine is hosted remotely). There may be higher potential to attract user notice when such a location indicator conveys intuitively meaningful information (*e.g.,* the country flag or city where the server is) rather than cryptic symbols—a green lock, or an exclamation mark on a grayed out triangle, *etc*. This case is similar to that where a browser-trusted CA is compromised and no certificates are pinned for the victim domain. On the other hand, location verification provides that missing benefit of signaling such an adversarial situation using an intuitively meaningful visual cue; *e.g.,* the browser will either display an unexpected flag if the asserted country is different from that of the authentic server, or depending on implementation choices, show a struck out flag in case the location fails verification.

Several useful features can be built on top of server location verification. For instance, SLV can benefit from a policy-based mechanism [13], [49] that customizes how a browser automatically handles various transactions based on their location [50]. An instruction could be of the form *allow credit card transactions only at this set of countries* or *deny email logins in that set of locations*. This may also help control fraud and deter phishing attacks, ideally requiring no new user actions or decisions whatsoever.

Finally, SLV is compatible with known mechanisms (*cf.* [51]) that help systems scale, and increase efficiency to address time-sensitive applications. For example, the verifiers (acting as regular clients) could proactively measure delays periodically (*e.g.,* to high-runner websites) to reduce verification time. The centralized caching layer at the *Manager* implies that for most visited websites (*e.g.,* Alexa's top 1000), it is highly likely that clients will receive spontaneous responses any time. Note that SLV enforces repeating the location verification process when requested whenever the caching window expires.

8. Related Work

GeoPKI. GeoPKI [9] is a location-aware PKI system that associates certificates to geographic *spaces*, *e.g.,* land or property boundaries. A certificate contains a high granularity definition of the space to which it is associated. This could be in the form of GPS coordinates along with lateral and longitudinal distances that accurately delineate the space boundaries. To claim a space, the owner submits their space-defined certificate (self-signed or CA-signed) to a public log (*e.g.,* similar to certificate transparency [4]), and monitors the log to detect any other entity claiming ownership of their space. To validate a space ownership, GeoPKI relies on CA-issued Extended Validation (EV) certificates, associated to a real world street address. An attacker would thus need to either compromise a CA to issue an EV certificate to tie its public key to the fraudulently-claimed space, or forge legal documents proving such ownership.

The goals and threat model of GeoPKI differ from those we address herein. GeoPKI is designed to provide assurance that no other (malicious) entity issues a certificate claiming a geographical space that is already claimed by the (legitimate) owner, but does not indicate or assure the location of the actual server a client is connected to.[6] In contrast to GeoPKI, SLV verifies server locations in realtime thus compromising a CA alone is insufficient; the attacker must also evade SLV to succeed in a MitM attack.

Client Presence Verification. CPV [34] verifies geographic location claims of Internet *clients*. Client locations are corroborated based on triangular *areas* derived from delay measurements. While this enables CPV to verify client locations with high granularity, its verification process suffers occasional Triangular Inequality Violations (TIVs) [52]. In contrast, while SLV selects verifiers forming triangles, it does not verify server presence within them, nor use triangular areas; its use of Thales's theorem avoids TIVs entirely, and reduces the number of delay measurements required between verifiers and the webserver. SLV also provides assurance to clients about *server* locations (note that servers and clients differ fundamentally in many factors, including that a third party location verification service provider can easily get clients to run code, *e.g.,* using JavaScript as CPV does; this is not applicable when verifying geographic locations of webservers).

Geolocating fast-flux servers. Delay-based geolocation of fast-flux hidden webservers has been proposed [53]; hidden behind proxies, their IP addresses are not known to the client. When geolocating a webserver, the geolocation service provider can first detect that the webserver is hidden behind a proxy by noticing a large difference between the RTTs measured on the network layer (*e.g.,* using `ping`) and the application layer (*e.g.,* using an HTTP GET). To estimate the hidden server's location, a group of landmarks measure application layer RTTs to the server, which are then used to obtain rough estimates to the direct RTTs between the landmarks and the hidden server (excluding the proxy). The RTTs are then mapped to distances to constrain the region of the hidden server relative to the landmarks [30].

This geolocation mechanism aims at disclosing an inconsistency between the geographic location of the sever terminating the TCP connection and the one processing HTTP requests. SLV does not attempt to determine webserver locations, but rather verifies the plausibility of the webserver within an asserted region. While attempts to evade SLV may include hiding the attacker's IP address behind a proxy, SLV handles that attack differently—it reports that the asserted location (that of the IP address seen by the client) is not verified.

Using network characteristics to detect phishing and pharming attacks. Previous literature [54] proposed to use routing information, including mean RTTs and standard deviation of network delays, to detect phishing and pharming attacks. The authors fed the network characteristics of many legitimate and malicious websites to different classifiers as the training data set, then used the classifiers to detect other malicious sites. Their evaluation showed an accuracy above 99%, further affirming the plausibility of relying on such network characteristics in exhausting server impersonation attacks.

Flagfox extension. Flagfox is an example[7] Firefox extension that looks up the countries of webserver IP addresses as a user browses the Internet, and displays the country flag in the URL bar. The flag is based on the tabulated location of the IP address, not the country TLD in the domain name. Flagfox uses Maxmind's IP database for geolocation,[8] and does not employ any location verification mechanism. Since locations obtained by tabulation-based techniques are falsifiable [8], *e.g.,* by the IP address owner, they are unreliable in adversarial environments. For instance, an adversary aiming to impersonate the University of Tennessee's website (*e.g.,* through phishing, pharming or a MitM attack) could register the IP address assigned to its malicious webserver to be in Knoxville, Tennessee. Indeed, a previous study [31] found that most of Google's IP addresses are reported by the American Registry for Internet Numbers (ARIN)[9] to be physically located in Mountain View, California; such clearly incorrect assertions have been proven wrong [31].

Network Coordination Systems (NCSs). NCSs are conceptually different from measurement-based geolocation. In the former, each network node is located based on its network position in the *delay space*, whereas the latter determines the physical positions of nodes based on their geographic distances. NCSs are useful because when the coordinates of all nodes are calculated, the delay between any pair of nodes could be predicted without actively measuring it, thus eliminating the need of flooding the network with delay-measuring probes to find all pair-wise delays. Measurement-based geolocation can be seen as a subse-

6. A client browser could be connected to a webserver in China, legally owned and operated by an entity in the US. The GeoPKI EV certificate then validates the US location, not the physical server's location.

7. Other extensions exist with similar objectives.

8. https://www.maxmind.com

9. http://whois.arin.net

quent step, which takes-in network delays (which may be, *e.g.,* predicted, calculated, actively measured, or passively estimated), and approximates the geographic location accordingly. NCSs are thus considered a different problem, with its own threats and defenses, efficiency parameters, and evaluation metrics.

SALVE. Another scheme whereby a server's physical location is used as a server authentication factor is SALVE [55].[10] In SALVE, location verification is achieved using the Location Service architecture [56]—a set of techniques developed and used by telecommunication operators to geolocate SIM-identified devices in the network.

9. Concluding Remarks

The server location verification mechanism detailed herein does not conflict with the web's growing trend of distributed content dissemination and geographically diverse replicated caching. The initial front-end server to which a client connects is the port of entry to the distribution infrastructure, if one is being used; paying more attention to that server, *e.g.,* by verifying its physical presence in a known/expected geographic location as explained herein, provides information often relevant to the target domain's authenticity. Thus, SLV works regardless of the distribution and architecture of such infrastructure. Additionally, depending on a client's location, a finite set of n such "ports of entry" are typically expected for any single domain, and that set is often stable. Pinning several server locations (see Section 5) is thus beneficial for new and verified locations.

Despite efforts from the security community to address shortcomings in the current server authentication ecosystem, PKI compromise still admits MitM attacks, *e.g.,* due to slow or non-adoption of primitives like key pinning, or user inability to reliably react to visual browser cues. The proposals herein constitute a new and parallel server authentication dimension (*e.g.,* comparable to client multi-factor authentication), relying not on the standard *something you have* principle (namely, the server private key), but in addition *where you are*. While the general notion of location-based authentication is known, the novelty herein is the measurement-based mechanism itself which verifies server locations in realtime, in a manner compatible with the current server authentication standards, and without requiring human-user involvement in decision making.

To mount a successful MitM attack when SLV is used, the adversary must, in addition to compromising the TLS infrastructure, co-locate its malicious (possibly virtual) machine in the geographic vicinity of the authentic one. In addition to being less scalable, this places a heavy set of burdens including an attack customized to the location of each target server. A mechanism like SLV thus compels the adversary to make a true assertion about the location of its fraudulent servers, both divulging the fraudulent servers' true geographic location, and forcing the adversary to operate in the geographic vicinity of the authentic webserver—often a region in a more familiar country, or with more favourable laws and accountability measures.

SLV leverages established networking principles that location information can be inferred from timing measurements, and existing methodological guidelines for use of timing measurements to achieve server location verification. While large-scale evaluation of SLV's verification process is not the main focus of this paper, preliminary experiments highlight the algorithm's efficacy in verifying webservers' geographic locations, by means immediately deployable through a browser extension without requiring webserver modifications.

10. We became aware of SALVE subsequent to our submission.

Acknowledgments

The second author acknowledges funding from the Natural Sciences and Engineering Research Council of Canada (NSERC) for both his Canada Research Chair in Authentication and Computer Security, and a Discovery Grant.

References

[1] J. Clark and P. C. van Oorschot, "SoK: SSL and HTTPS: Revisiting past challenges and evaluating certificate trust model enhancements," in *IEEE Symposium on Security & Privacy*, 2013, pp. 511–525.

[2] R. Holz, L. Braun, N. Kammenhuber, and G. Carle, "The SSL landscape: a thorough analysis of the X. 509 PKI using active and passive measurements," in *ACM IMC*, 2011, pp. 427–444.

[3] M. Kranch and J. Bonneau, "Upgrading HTTPS in mid-air: An empirical study of strict transport security and key pinning," in *NDSS*. Internet Society, 2015.

[4] B. Laurie, "Certificate Transparency," *Communications of the ACM*, vol. 57, no. 10, pp. 40–46, 2014.

[5] R. Dhamija, J. D. Tygar, and M. Hearst, "Why phishing works," in *ACM CHI*, 2006, pp. 581–590.

[6] Y. Li, S. Chu, and R. Xiao, "A pharming attack hybrid detection model based on IP addresses and web content," *Optik-International Journal for Light and Electron Optics*, vol. 126, no. 2, pp. 234–239, 2015.

[7] N. Vratonjic, J. Freudiger, V. Bindschaedler, and J.-P. Hubaux, "The inconvenient truth about web certificates," in *Economics of info sec and priv III*. Springer, 2013, pp. 79–117.

[8] J. A. Muir and P. C. van Oorschot, "Internet geolocation: Evasion and counterevasion," *ACM Comput. Surv.*, vol. 42, pp. 4:1–4:23, 2009.

[9] T. H.-J. Kim, V. Gligor, and A. Perrig, "GeoPKI: Converting Spatial Trust into Certificate Trust," in *Springer EuroPKI*, 2013, pp. 128–144.

[10] Akamai, "Facts & Figures," https://www.akamai.com/us/en/about/facts-figures.jsp, 2015.

[11] A. M. Abdou, A. Matrawy, and P. C. van Oorschot, "Accurate Manipulation of Delay-based Internet Geolocation," in *ACM AsiaCCS*, 2017, pp. 887–898.

[12] P. Gill, Y. Ganjali, B. Wong, and D. Lie, "Dude, where's that IP? Circumventing measurement-based IP geolocation," in *USENIX Security*, 2010, pp. 241–256.

[13] D. Wendlandt, D. G. Andersen, and A. Perrig, "Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing," in *USENIX ATC*, 2008.

[14] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "PlanetLab: An Overlay Testbed for Broad-coverage Services," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 33, pp. 3–12, 2003.

[15] Z. N. J. Peterson, M. Gondree, and R. Beverly, "A position paper on data sovereignty: The importance of geolocating data in the cloud," in *USENIX HotCloud*, 2011.

[16] C. Karlof, U. Shankar, J. D. Tygar, and D. Wagner, "Dynamic pharming attacks and locked same-origin policies for web browsers," in *ACM CCS*, 2007, pp. 58–71.

[17] T. Kiravuo, M. Sarela, and J. Manner, "A survey of Ethernet LAN security," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1477–1491, 2013.

[18] S. Goldberg, "Why is it taking so long to secure Internet routing?" *Communications of the ACM*, vol. 57, no. 10, pp. 56–63, 2014.

[19] I. C. Society, "IEEE Std. 802.1D. Media access control (MAC) Bridges," 2004. [Online]. Available: http://standards.ieee.org/getieee802/download/802.1D-2004.pdf

[20] Wired, "How to Detect Sneaky NSA "Quantum Insert" Attacks," https://www.wired.com/2015/04/researchers-uncover-method-detect-nsa-quantum-insert-hacks/, 2015, 2005.

[21] A. Ornaghi and M. Valleri, "Man in the middle attacks," in *Blackhat Conference Europe*, 2003.

[22] M. Marlinspike, "More tricks for defeating SSL in practice," Black Hat USA, 2009.

[23] J. Liang, J. Jiang, H. Duan, K. Li, T. Wan, and J. Wu, "When HTTPS meets CDN: A case of authentication in delegated service," in *IEEE Symposium on Security & Privacy*, 2014, pp. 67–82.

[24] V. N. Padmanabhan and L. Subramanian, "An investigation of geographic mapping techniques for Internet hosts," in *ACM SIGCOMM*, 2001, pp. 173–185.

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

[26] Z. Dong, R. D. Perera, R. Chandramouli, and K. Subbalakshmi, "Network measurement based modeling and optimization for IP geolocation," *Elsevier Computer Networks*, vol. 56, pp. 85–98, 2012.

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

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

[29] R. Landa, J. T. Araújo, R. G. Clegg, E. Mykoniati, D. Griffin, and M. Rio, "The large-scale geography of Internet round trip times," in *IFIP Networking*, 2013, pp. 1–9.

[30] B. Gueye, A. Ziviani, M. Crovella, and S. Fdida, "Constraint-based geolocation of Internet hosts," *IEEE/ACM Trans. Netw.*, vol. 14, pp. 1219–1232, 2006.

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

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

[33] M. Arif, S. Karunasekera, and S. Kulkarni, "GeoWeight: Internet Host Geolocation Based on a Probability Model for Latency Measurements," in *Australian Computer Society ACSC*, 2010, pp. 89–98.

[34] A. M. Abdou, A. Matrawy, and P. C. van Oorschot, "CPV: Delay-based Location Verification for the Internet," *IEEE Trans. Dependable and Secure Computing, TDSC*, vol. 14, no. 2, pp. 130–144, 2017.

[35] D. McCullagh, "How Pakistan knocked YouTube offline," http://www.cnet.com/news/how-pakistan-knocked-youtube- offline-and-how-to-make-sure-it-never-happens-again/, 2008.

[36] R. Hiran, N. Carlsson, and P. Gill, "Characterizing large-scale routing anomalies: A case study of the China telecom incident," in *Springer PAM*, 2013, pp. 229–238.

[37] B. Huffaker, M. Fomenkov, and K. Claffy, "Geocompare: a comparison of public and commercial geolocation databases," CAIDA, Tech. Rep., 2011.

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

[39] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira, "Avoiding Traceroute Anomalies with Paris Traceroute," in *ACM IMC*, 2006, pp. 153–158.

[40] I. Agricola and T. Friedrich, *Elementary Geometry*, 1st ed. American Mathematical Society, 2008, vol. 43.

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

[42] V. Dukhovni and W. Hardaker, "The DNS-Based Authentication of Named Entities (DANE) Protocol: Updates and Operational Guidance," RFC 7671 (Proposed Standard), Internet Engineering Task Force, 2015.

[43] C. Evans, C. Palmer, and R. Sleevi, "Public Key Pinning Extension for HTTP," RFC 7469 (Proposed Standard), Internet Engineering Task Force, 2015.

[44] C. Davis, I. Dickinson, T. Goodwin, and P. Vixie, "A Means for Expressing Location Information in the Domain Name System," RFC 1876 (Experimental), 1996.

[45] N. Heninger, Z. Durumeric, E. Wustrow, and J. A. Halderman, "Mining your Ps and Qs: Detection of widespread weak keys in network devices," in *USENIX Security*, 2012, pp. 205–220.

[46] S. Fahl *et al.*, "Why Eve and Mallory love Android: An analysis of Android SSL (in)security," in *ACM CCS*, 2012, pp. 50–61.

[47] D.-Y. Yu, E. Stobert, D. Basin, and S. Capkun, "Exploring website location as a security indicator," *arXiv preprint arXiv:1610.03647*, 2016.

[48] A. Adelsbach, S. Gajek, and J. Schwenk, "Visual spoofing of SSL protected web sites and effective countermeasures," *LNCS Information Security Practice and Experience*, vol. 3439, p. 204, 2005.

[49] N. Santos, R. Rodrigues, K. P. Gummadi, and S. Saroiu, "Policy-sealed data: A new abstraction for building trusted cloud services," in *USENIX Security*, 2012, pp. 175–188.

[50] M. van Polen, G. Moura, and A. Pras, "Finding and Analyzing Evil Cities on the Internet," in *Springer Autonomous Infrastructure, Management, and Security*, 2011, pp. 38–48.

[51] A. Bates, J. Pletcher, T. Nichols, B. Hollembaek, and K. R. Butler, "Forced perspectives: Evaluating an ssl trust enhancement at scale," in *ACM IMC*, 2014, pp. 503–510.

[52] Y. Zhang and H. Zhang, "Triangulation Inequality Violation in Internet Delay Space," in *Adv. in Comp. Sci. and Info. Eng.* Springer, 2012, vol. 169, pp. 331–337.

[53] C. Castelluccia, M. A. Kaafar, P. Manils, and D. Perito, "Geolocalization of proxied services and its application to fast-flux hidden servers," in *ACM IMC*, 2009, pp. 184–189.

[54] H. Kim and J. Huh, "Detecting dns-poisoning-based phishing attacks from their network performance characteristics," *Electronics Letters*, vol. 47, no. 11, pp. 656–658, 2011.

[55] D.-Y. Yu, A. Ranganathan, R. J. Masti, C. Soriente, and S. Capkun, "SALVE: Server Authentication with Location Verification," in *ACM MobiCom*, Oct 2016, pp. 401–414.

[56] 3GPP. TS 23.271, "Functional stage 2 description of Location Services (LCS)," http://www.3gpp.org/dynareport/23271.htm, 2015.

Appendix

The following explains the column headers of the evaluation framework in Table 5 (Section 6). It is reproduced essentially verbatim from Clark and van Oorschot [1].

A.1. Security Properties Offered by Primitives

Detecting Certificate Substitution (Table 5–column A).

- *Detects MITM.* Provided if a primitive detects a MITM attack involving a substituted certificate/location. If a primitive requires risk or "blind" trust on first use (TOFU) to detect these attacks, we use ○ to denote partial fulfillment.
- *Detects Local MITM.* We say a MITM attack is *local* if the adversary is able to insert itself into connections to the server from only a subset of clients (through, *e.g.,* poisoned local DNS cache or on-path interception near the client).
- *Protects Client Credential.* An HTTPS connection is often used to transmit a client authentication credential (*e.g.,* a password or secure cookie) to the host. If a primitive focuses on protecting against credential theft during an HTTPS MITM attack, it provides this benefit. Blind TOFU primitives partially fulfill.
- *Updatable Pins.* Some primitives that use pinning make false-reject errors if a server updates its public key, switches issuing CAs, or uses multiple certificates for the same host. Primitives that resolve such false-reject errors provide this benefit.

Detecting TLS Stripping (Table 5–column B).

- *Detects TLS Stripping.* Since many enhancements to HTTPS do not take into account security-relevant details of a connection until there is an HTTPS request from the client, TLS stripping bypasses them. Primitives that can detect stripping attacks fulfil this benefit, and partially fulfil it if they rely on blind TOFU.
- *Affirms POST-to-HTTPS.* Primitives that deter (through enforcement or a security indicator) POST requests from being submitted over HTTP fulfil this benefit.

PKI Improvements (Table 5–column C).

- *Responsive Revocation.* We assume in evaluating the primitives that CRLs or OCSP responses are not available and examine their ability to otherwise detect a revoked certificate; primitives which do, fulfill this benefit.
- *Intermediate CAs Visible.* A primitive fulfils this benefit if every intermediate CA is visible to the user at any time.

A.2. Evaluation Criteria for Impact on HTTPS

Security & Privacy.

- *No New Trusted Entity.* A primitive not introducing any new trusted parties fulfills this property, with partial fulfillment if the responsibilities of an already trusted party are expanded.
- *No New Traceability.* A primitive that does not introduce any new parties that will become aware of all (or a fraction of) sites a user visits over HTTPS fulfills this property.
- *Reduces Traceability.* A primitive fulfills this prop if it eliminates such a class of entities mentioned in the previous point.
- *No New Auth'n Tokens.* Many primitives effectively introduce new server authentication tokens, like pins or signed OCSP responses, that are transmitted to the client. Generally procedures for issuing, updating, and revoking these new tokens must be established, as well as integrity protection. This property is fulfilled by a primitive that does not introduce new tokens.

Deployability.

- *No Server-side Changes.* Primitives that do not change how web servers implement TLS and HTTPS have the greatest potential for deployment. Primitives that do not require any server involvement or code changes fulfill this property, while primitives that only require servers to participate in a way that does not involve changing any server code partially fulfills it.
- *Deployable without DNSSEC.* Some primitives rely on DNSSEC which has not been fully deployed; if they don't, they fulfill this property.
- *No Extra Communications.* This is fulfilled by primitives that do not introduce an extra communication round that blocks completion of the connection.
- *Internet Scalable.* This is fulfilled by systems that could foreseeably support enrolment from all current HTTPS servers and potentially beyond.

Usability.

- *No False-Rejects.* A primitive fulfills this property if it does not reject legitimate server certificates. Otherwise, it requires the user (*e.g.,* through a warning dialogue) to distinguish false-rejects from an actual attack. This is often not fulfilled when user are frequently being presented with warning messages.
- *Status Signalled Completely.* If users cannot readily determine the reason for trust, the primitive lacks this property. A partial fulfillment is awarded if the basis of trust is not clear because server enrolment is optional, and thus a fallback trust mechanism may be also necessary.
- *No New User Decisions.* This is fulfilled when primitives are automated and do not require users to respond correctly to new security cues or dialogues. If a primitive introduces a new security cue, it will most likely fail to fulfill this benefit.