Server Notaries: A Complementary Approach to the Web PKI Trust Model

Emre Yüce[1] and Ali Aydın Selçuk[2]

[1] Dept. of Cryptography, Middle East Technical University, Ankara, Turkey
e132740@metu.edu.tr
[2] Dept. of Computer Eng., TOBB Univ. of Economics and Tech., Ankara, Turkey
aselcuk@etu.edu.tr

Abstract. SSL/TLS is the de facto protocol for providing secure communication over the Internet. It relies on the Web PKI model for authentication and secure key exchange. Despite its relatively successful past, the number of Web PKI incidents observed have increased recently. These incidents revealed the risks of forged certificates issued by certificate authorities without the consent of the domain owners. Several solutions have been proposed to solve this problem, but no solution has yet received widespread adaption due to complexity and deployability issues. In this paper, we propose a practical mechanism that enables servers to get their certificate views across the Internet, making detection of a certificate substitution attack possible. The origin of the certificate substitution attack can also be located by this mechanism. We have conducted simulation experiments and evaluated our proposal using publicly available, real-world BGP data. We have obtained promising results on the AS-level Internet topology.

Keywords: Web PKI, SSL/TLS, man-in-the-middle attack, notary

1 Introduction

Today the Internet is massively used for e-government, e-commerce, and e-banking applications unlike its early days with static web pages. These applications require exchange of sensitive data including financial or personal information. It is crucial to provide a secure connection for this communication which is achieved using different network protocols. Secure Socket Layer (SSL) [14] and its successor Transport Layer Security (TLS) [9] are protocols designed to provide confidentiality, authenticity, and integrity over the Internet. SSL[3] relies on the *Web PKI* trust model [7] for authentication and secure key exchange. In this model, Certificate Authorities (CAs) issue X.509 digital certificates that bind the SSL server identity to a public key. SSL clients receive the digital certificate when they request to establish a secure connection to the server. They verify it using the embedded public keys of CAs in their browser or operating system certificate trust stores.

[3] Hereafter, we use SSL to mean both SSL and TLS.

There exist serious concerns regarding the reliability of the Web PKI trust model. The model employs a list of CAs that are trusted by default. There are hundreds of fully trusted root CAs from more than 50 countries [10]. They are able to delegate their authority to subordinate CAs (sub-CAs) as well. For any domain name both root CAs and sub-CAs are able to issue valid certificates, trusted by most of the browsers, without the consent or knowledge of the domain owner. One of the most recent incidents has happened in March 2015 [24]. Google has detected forged certificates for several Google domains. A sub-CA certificate, signed by National Informatics Centre of China (CNNIC), has been used in the incident. Browser and operating system vendors revoked the certificates after the discovery of the attack. This attack is an example of misuse of sub-CA certificates. Other examples are IndiaNIC case in July 2014 [23], ANSSI case in December 2013 [22], and TurkTrust case in January 2013 [21]. Yet in other incidents, CAs were compromised resulting in the fraudulent issue of forged certificates [41], [6]. Governmental and private organizations may also use forged certificates for their surveillance activities [27], [34], [37].

In response to these vulnerabilities of the Web PKI, several protocols have been proposed as an enhancement or an alternative to the current model. These proposals include Public Key Pinning [19], Perspectives [42], Convergence [29], DANE [33], Sovereign Keys [11], and Certificate Transparency [25]. Although some of these proposals are used, there is no commonly accepted and widely deployed solution yet. The security threats and design constraints to be addressed are still being discussed [5], [26]. The solution should be applicable for any participant, should comply with the current model, and should propose a practical method which does not introduce complex components, and does not depend on end user decisions.

In this work, we focus on the fact that the SSL servers, in the current trust model, are not able to obtain information on how their certificates are observed at different locations on the network. We propose a complementary solution, the *server notaries* method, which enables servers to get their certificate views across the Internet. In this way servers will be able to check whether their certificates are observed as expected. Thus detecting a certificate substitution will be possible. Moreover a server may locate the origin of the attack by analyzing certificate views from different vantage points. In order to see how our method performs on the Internet, we have conducted simulation experiments and evaluated our proposal at AS-level Internet topology using publicly available BGP data. We can summarize our primary contributions as follows:

- We propose the *server notaries* method, a practical and efficient mechanism that enables servers to observe their certificates from different points on the Internet. Our proposal makes *detecting* and *locating* a certificate substitution attack possible.
- We present results of simulation experiments conducted using real-life AS-level Internet topology data and evaluate how effective server notaries method can be at detecting a certificate substitution.

– We present a qualitative assessment of advantages and disadvantages of the server notaries method.

2 Server Notaries

The idea of observing the server certificate from different network vantage points has been used in several proposals to improve the Web PKI trust model. This idea was introduced in Perspectives [42], where Wendlandt et al. defined *notaries* as publicly available semi-trusted hosts deployed at various locations on the network. The main idea is that after a client obtains the server certificate in the usual way, it may compare received certificate with the server certificate obtained from a notary's network point of view. A difference between the certificates may indicate a certificate substitution. Different variants of notaries have been used in several different protocols. Similar proposals such as Convergence [29], DoubleCheck [1], and CrossBear [16] followed a similar method to enhance the Web PKI trust model.

In this work, we propose a complementary way of using notaries for detecting fake certificates and MITM attacks over the network. In our method, notaries are used by SSL servers rather than clients, hence the name is *server notaries*.

2.1 Scenario and Threat Model

Our scenario consists of an SSL server, a number of notaries and an adversary. The server in the scenario may be any kind of generic or special purpose server. It announces a certificate publicly to any client wishing to establish a secure channel. Notaries are pre-deployed publicly accessible semi-trusted hosts located at various network points and they are managed by different entities. We assume that the server has already obtained the current list of active notaries and their public keys, as we will explain later.

Our threat model considers an adversary who is able to modify the network traffic flowing over itself. Aim of the adversary is to eavesdrop and tamper with this traffic by executing non-selective MITM attacks against the server. In order to perform such an attack, the adversary may use one of the following methods:

– Obtaining a forged certificate for the servers domain name that is signed by a trusted CA or sub-CA.
– Using a revoked certificate before CRL update occurs and by interrupting OCSP queries.
– Launching an HTTPS downgrade attack.
– Using a certificate, untrusted by root stores (e.g. self-signed).

If the MITM attack is local, i.e. the adversary is located in the vicinity of the client, probably the adversary and the client are at the same subnetwork, the same ISP, or the same country. The adversary may be a governmental entity or the ISP itself. In this scenario, the server observes a fake certificate from the notaries deployed within the attack region and a genuine certificate from the

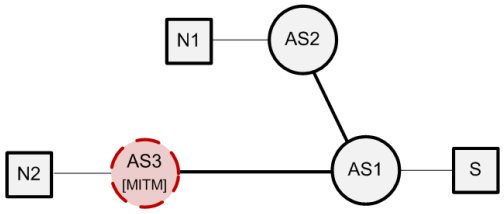

Fig. 1. A local MITM attack scenario showing AS level network paths between S (*server*), $N1$ (*notary*) , $N2$ (*notary*). Adversary is located at $AS3$. $N1$ observes the genuine certificate, $N2$ observes a fake certificate. Thus S infers that there exists a misbehaving node between S and $N2$.

remaining notaries. This scenario makes locating the adversary possible. Such an attack scenario is represented in Figure 1.

If the adversary is located at a network point close to the server, almost all network paths between the server and the notaries include the adversary. Hence the server will mostly observe a fake certificate from the notaries. The server should check its local network or inform its ISP about the issue.

Our threat model does not consider attacks exploiting implementation or configuration errors. Also we assume that the server is not compromised and is a trusted participant. The notaries are semi-trusted participants. We assume that the adversary is not able to break cryptographic primitives; i.e. the adversary cannot tamper with the data that provides authentication, encryption, or integrity.

2.2 Protocol Details

Server notaries method is based on the exchange of *observation request-response* messages between the server and the notary. The message transaction is given below and demonstrated in Figure 2.

1. Server selects a set of notaries from its notary list and initiates the protocol by sending an *observation request* to these notaries over a secure channel.
2. After receiving the observation request, a notary establishes a connection to the server as any SSL client would do.
3. The notary receives the server's certificate. If there exists an active adversary through the network path between the server and the notary, the notary will receive a fake certificate.
4. Notary sends the signed *observation response* to the server over the previously established secure channel. The observation response includes the observed certificate.

Server notaries method enables servers to *detect* and *locate* the certificate substitution. If the server receives an unexpected certificate, this is a sign of a certificate substitution between the server and the notary. Hence the server is

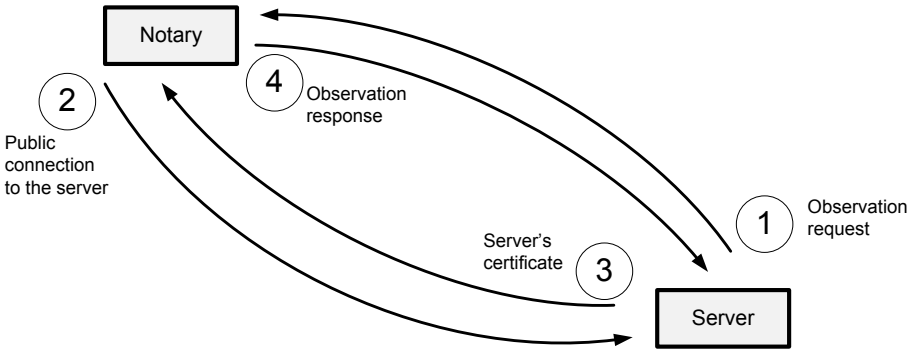

Fig. 2. Server notaries method overview: (*1*) Server sends an observation request to the notary over secure channel. (*2*) Notary connects to the server over public channel. (*3*) Server sends its certificate. (*4*) Notary sends observation response including the received certificate to the server.

able to detect a possible MITM attempt or a misissued certificate. Moreover the server is able to locate the network point where the certificate substitution occurs. Spotting the possibly misbehaving nodes through the network may be achieved by comparing the network paths between the server and multiple notaries.

Our proposal does not increase the complexity of the current system. Servers are expected to make periodical probes through the notaries. This can be implemented by minor changes on the server side. Clients are not a part of this method and will remain unmodified.

Similar to other notary-based solutions [42],[16], the server side implementation will include the contact information of a bootstrapping node which will be used to obtain an active list of notaries and their public keys so that the communication between the server and the notaries are secured.

As a final remark, we would like to note that although we have focused on detecting MITM attacks targeting the Web PKI, server notaries can be used in order to track the view of any certificate or public key served by other processes, such as SSH, as well.

3 Simulations

We have conducted a server notaries simulation on an AS-level Internet topology using publicly available BGP data. In this section we present the simulation details. First we present how we have collected and analyzed the data. Then we share our simulation methodology and conclude the section by commenting on the simulation results.

3.1 Data Collection and Analysis

Throughout this experiment we used the BGP data provided by the University of Oregon Route Views Project [40]. This project aims publishing data about the global view of the Internet using routing information. This project gives real time access to the routing data publicly. Routeviews data have been used in several projects. An already completed one is the NLANR [31] project which had used the data for AS path visualization and IPv4 address space utilization. In a more recent study, CAIDA [4] has been using Routeviews data to generate geographical location of hosts in conjunction with the NetGeo [30] database. CAIDA AS Relationships [3] project is another example. This project investigates business agreements between ASes based on customer/provider/peer relations.

There are collectors deployed worldwide which gather the routing data. They have established BGP connections with several BGP peers. By August 2015, there are 437 peering to 188 distinct ASes using 19 collectors in total [39]. It is observed that some of the collectors are deployed within Tier-1 networks. Collectors' main purpose is to observe advertised AS paths through the Internet. Although it is not feasible to deploy a collector at every AS for observation, it is shown that the public BGP information is enough to capture relatively complete AS level Internet topology [13].

We have downloaded and parsed the data set (MRT-formatted full-table RIBs Routing Information Base, i.e., BGP dumps.) for 9 August 2015 (08:00) for the vantage points: Oregon IX, Equinix Ashburn, ISC/PAIX, KIXP, LINX, DIXIE/WIDE, RouteViews-4, Sydney, and São Paulo. The data includes BGP tables collected from 188 distinct ASes world wide. The raw data includes misleading information such as repetition of AS paths or loops inside AS paths. We have discarded data sets that are truncated or having limited IP space. We have removed invalid paths like loops or repetitive ASes and duplicate paths. After these steps we have obtained the *AS path dataset* including more than 11 million AS paths from 124 distinct ASes destined to almost all ASes observed worldwide.

3.2 Server Notaries Simulation

Methodology Server notaries method has two types of components namely the *servers* and the *notaries*. We consider the AS-level Internet topology where BGP policies determine the AS paths available between two ASes.

As for the *servers*, we used the *collectors* of the AS path dataset described in Section 3.1. Recall that we have obtained AS paths sourcing from 124 distinct ASes to almost all ASes observed in the Internet. Hence, we have decided to use the 124 distinct source ASes as our servers in the simulation.

An important question regarding the deployment of the server notaries method is how to distribute the notaries over the Internet for an effective utilization. An intuitive idea for deployment is to put the notaries at the highly-connected ASes. To choose the notary ASes, we sorted all ASes in descending order with respect to the following five AS features and took a given number of highest ranking ones.

Last three items are related to the business agreements between ASes which are typically confidential but may be inferred from BGP data [28], [15].

- **Degree**: The number of ASes directly connected to an AS.
- **Prefix**: The number of prefixes an AS announces.
- **Provider:** The number of providers an AS has.[4]
- **Customer:** The number of customers an AS has.
- **Peer:** The number of peers an AS has.[5]

We used RouteViews BGP data to calculate number of announced prefixes per AS. We used CAIDA AS Relationship dataset [3], which presents the AS relations as provider-to-customer or peer-to-peer, to calculate the remaining AS features.

We say that ASes observed between the server AS and the notary AS are *covered* by the notary for the server. Covered ASes are critical at detecting adversaries. Assume an adversary is located at one of the covered ASes and substitutes the server certificate by a forged one. Then the server would detect the adversary by querying the respective notary's view since the notary observes the forged certificate.

A simple scenario is presented in Figure 3. The server S is located at AS7 and the notaries *N1* and *N2* are located at AS1, AS4 respectively. AS1, AS2, AS3, and AS7 are covered by *N1*. AS4, AS5, AS6, and AS7 are covered by *N2*. Server detects the adversary, located at AS6, by querying *N2*.

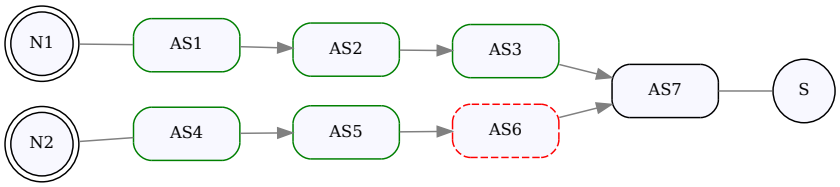

Fig. 3. Sample set of AS paths including the server (S) and the notaries (*N1*, *N2*). An adversary is located at *AS6*. *N1* observes the genuine certificate. *N2* is effected by the adversary on its path to S and observes the fake certificate.

Performance Metrics We define the following performance metrics over the AS path dataset generated in Section 3.1. Hereafter s denotes an SSL web server AS, n_i denotes a notary AS, and N denotes the set of all notary ASes.

[4] A *provider* is an AS that enables its customers to reach other ASes by carrying customers' transit traffic over itself.

[5] A *peering* is defined as the exchange of traffic between the respective customers of each peer free of charge. This kind of connection may be observed between ISPs who cannot afford additional Internet services for better connection or between administrative domains who wish to deploy a backup connectivity.

***CAS*(s, N) :** "Covered AS" (CAS) is the number of distinct ASes observed through the AS paths between s and all notaries in N.

***TAS* :** "Total AS" (TAS) is the number of distinct ASes observed in the AS path dataset.

In order to calculate $CAS(s, N)$ value for one server s, we scanned the AS path dataset for paths having s and n_i as the first and last ASes, $\forall n_i \in N$. We counted the number of distinct ASes observed on these paths and found the $CAS(s, N)$ value. After calculating the $CAS(s, N)$ values for all servers, we calculated their mean value CAS. Using CAS and TAS values, we calculated CAS *Ratio* as follows:

$$CAS\ Ratio = \frac{CAS}{TAS} \tag{1}$$

This value gives the ratio of covered distinct ASes using the set of notary ASes N.

***CASH*(s, N) :** "Covered AS Hit" (CASH) is the total number of occurrences (including multiple counts) of covered ASes in the AS path dataset.

***TASH* :** "Total AS Hit" (TASH) is the total number of occurrences (including multiple counts) of all ASes in the AS path dataset.

We found covered ASes by n_i for s, $\forall n_i \in N$. Then we counted the occurrences of these ASes in the AS path dataset and found $CASH(s, N)$ value. After calculating $CASH(s, N)$ values for all servers, we calculated their mean value $CASH$. Using $CASH$ and $TASH$ values, we calculated $CASH$ *Ratio* as follows:

$$CASH\ Ratio = \frac{CASH}{TASH} \tag{2}$$

$CASH$ *Ratio* value represents how frequent the covered ASes are observed over the AS path dataset. This is also the probability that a random AS path includes a covered AS. If an adversary, launching a MITM attack by certificate substitution, is located at one of the covered ASes, it will be detected using our method. Hence, we interpret $CASH$ *Ratio* as the *probability of detecting an adversary* at AS-level.

Results The contribution of this simulation is twofold. Firstly, we evaluate how successful server notaries method is at detecting certificate substitution attacks. Secondly, we analyze the effect of several AS features on AS selection for notary deployment.

CAS *Ratio* values are given in Figure 4. This figure shows that top n ASes with the highest number of providers will cover a larger portion of the network than other alternatives, for a given number n. For instance, top 200 ASes from the "provider" list cover approximately 1.5% of all ASes where top 200 ASes from the other lists cover less than 1% of all ASes.

$CASH$ *Ratio* values, which measure the probability of detecting an adversary, are presented in Figure 5. The results are very promising. By deploying notaries at top 200 ASes from the "degree" list, probability of detecting an adversary at the AS level is more than 50%. The simulation results show that it is

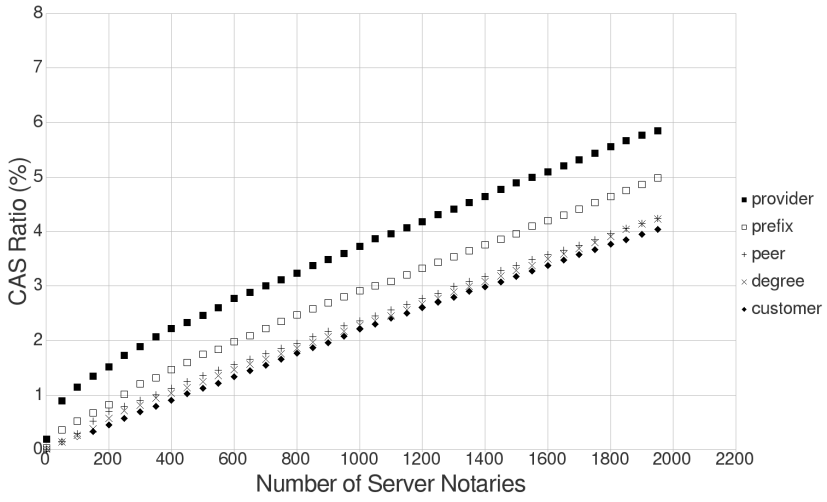

Fig. 4. Percentage of covered ASes (*y-axis*) with respect to the number of notaries (*x-axis*), selected according to the ASes features given in the legend.

better to deploy notaries at ASes with higher degrees in order to have a higher probability of detecting adversaries. By deploying notaries at the top 2000 ASes from the degree list, the *CASH Ratio* becomes 70%.

4 Related Work

There exist several proposals suggesting improvements to the current Web PKI trust model. Some of them try to replace the CA infrastructure completely, while others try to fit in and enhance the current model.

Pinning methods try to detect certificate substitutions at the client side [19]. Pinning is the process of associating a host with a certificate (or a public key). HPKP creates pins by the user's browsing history [35]. TACK uses server-pushed pins with the TOFU method [38]. Google deploys preloaded pins for various domain names in Chrome [20]. These methods are successful at detecting certificate changes which are possible MITM attacks. They however have some issues about revocation and certificate updates.

Another proposal is binding SSL keys to DNS entries using DNSSEC namely DANE [33]. This proposal may be seen as pinning keys to the DNS entries. In order for the DANE solution to be used, the vast majority of DNS servers should be configured to use DNSSEC. Also revocation is again problematic in DANE since all DNS records, including caches, worldwide should be updated in case of a public key update. This depends on the TTL value of the records.

Perspectives [42] is the first notary-based solution which utilizes notaries in order to observe server certificates from different network vantage points. Convergence [29] improves the Perspectives proposal by using bounce notaries

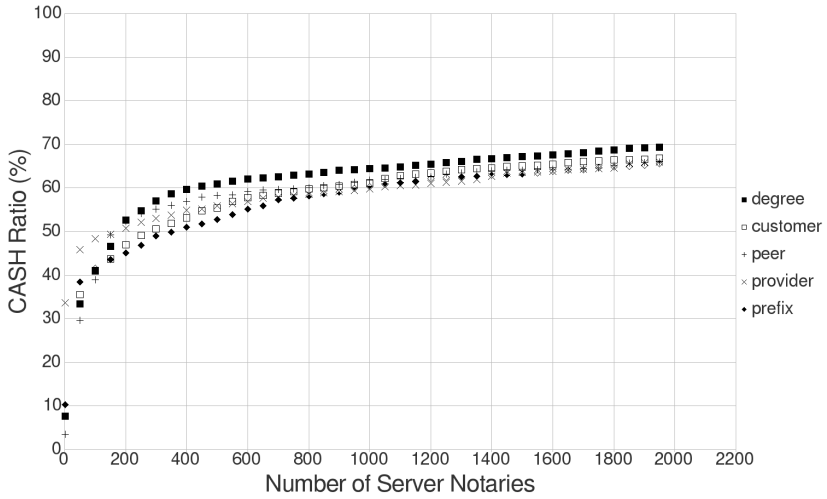

Fig. 5. Percentage of covered ASes hit (*y-axis*) with respect to the number of notaries (*x-axis*), selected according to the ASes features given in the legend.

to prevent privacy issues, enabling other methods (DANE, CAs, etc.) to be used for authentication, and solving the notary lag problem. Doublecheck [1] proposes using the TOR network instead of notaries. DetecTor [8] is a similar solution extending the usage of TOR idea to any protocol. An interesting idea for both detecting and locating the adversaries using notaries, originally called hunters, has arisen in the CrossBear proposal [16]. Notary-based solutions are generally criticized for certificate update issues and ineffectiveness in the case when adversaries are close to the server [5].

The ICSI Certificate Notary [18] and the EFF SSL Observatory [12] projects collect SSL certificates and publish statistical information about them. The ICSI Certificate Notary also provides a public DNS interface to query its database. These projects collect the certificates by actively probing the websites. As another approach, Huang et al. [17] have used client-side applets implemented in the Facebook website in order to analyze the certificates observed by the client. They have analyzed more than 3 million SSL connections and shared the properties of the observed certificates.

Sovereign Keys method [11] is a combination of server pinning and logging based methods. Server specifies a public key and logs it at a publicly available append-only log. Losing the private key may end up in losing the domain. Another example is Certificate Transparency method [25] proposed by Google. Every issued certificate is logged at a publicly available append-only and read-only log with a signed certificate timestamp (SCT). Thus certificates are transparent and verifiable. It is claimed that a MITM attack may be launched by redirecting a client to a specific log or by using a rogue CA [36]. Also revocation seems problematic in logging-based methods since the logs are append and read only.

In fact, Certificate Transparency does not claim to prevent MITM attacks but to detect them as fast as possible.

There exist proposals focusing on the current binary trust model of the Web PKI with trust computation enhancements [32], [2].

5 Discussion

The current Web PKI model is heavily used by billions of users everyday. It is not possible to interrupt the model and to change it by setting a "Flag Day". Hence a viable solution should propose a smooth, gradual transition. It would better include a transition period that interoperates with the current model at least for a while. Server notaries method proposes a quick fix for the vulnerabilities observed in the Web PKI trust model; our proposal would aid servers to mitigate certificate substitution attacks until a final consensus is reached.

The number of participating entities on the Internet is increasing every day. A potential solution should scale as the Internet grows and any participant should be able to use it. For instance, embedding public keys into browsers (preloaded pins) aided researchers in detecting several incidents [21], [23], [24]. However it is not feasible to embed each and every SSL public key in the world into the browsers. On the other hand, the solution should not require every one in the world to participate in order to work properly. For instance, Certificate Transparency enables detecting forged certificates for the participating CAs. It is not applicable, however, to non-participating CAs. Similarly, DANE requires DNSSEC to be deployed at every DNS server worldwide. Thus it can be stated that these solutions are limited by the degree of deployment. It is not the case for server notaries method as any server is able to use it and observe its certificate throughout the Internet. Also it does not require every entity to participate.

Complexity is the enemy of security. The more components a solution has, the harder it is to make it secure. The solution should propose a practical method which does not introduce complex components. Also, it should require as few changes as possible at the server and client sides. Servers, using the server notaries method, will make periodical probes to the notaries. This can be implemented by minor changes on the server side. Notaries can be deployed worldwide using cloud infrastructures. Clients will remain unmodified.

Another issue at the client side is the privacy. In the current model, whenever a client visits a website over SSL, the client's browser queries the CA's OCSP responders to verify that the server certificate is not revoked. Hence, the browsers already leak information about the client's SSL browsing history. Similarly some notary-based solutions suffer from privacy issues. The proposed solution should not introduce additional privacy issues. As clients are not a part of the server notaries method; it does not introduce any privacy issues.

Some of the notary-based solutions solve the privacy issues by anonymizing the communication over the TOR network [8], [1], which causes extra latency for every newly observed certificate at the client side. A usable solution should

not add extra latency. The server notaries method will just create extra network traffic on the server side which will not constitute a latency problem.

Notary-based solutions and pinning methods may produce false positive warnings for server farms with multiple different certificates or for websites updating their certificates frequently [5]. Users are expected to make a final decision in such cases. There are also MITM attack detection methods proposed to be used by tech savvy users [16]. A solution may give feedback to the user in case of a suspicious case. However it should not fully depend on end user decisions. Our proposal expects a decision from the server. As the server has the genuine certificate, it can make a final decision for the observed certificate easily.

The deployment of the notary nodes across the Internet is a major issue of our protocol. As noted in [42], independent nodes run by volunteers, like TOR relays, would make an excellent notary infrastructure. Bootstrapping servers can also be implemented à la TOR.

6 Conclusion

Recent incidents have demonstrated the vulnerabilities in the Web PKI trust model. As most of these vulnerabilities remain unsolved, number of MITM attacks are expected to increase over time. Unfortunately, it may be thought that there will not be a final, elegant solution in the near future by looking at the complexity and deployability issues of the proposed solutions. We have proposed a practical mechanism which enables servers to observe their own certificates using public notaries. This will bring the server administrators into the game as they will try to detect attacks against their servers. Simulations, conducted using real-life Internet topology data, have shown promising results for the effectiveness of the proposed solution.

Acknowledgments. We thank Onur Bektaş and Uğur Yılmaz from TÜBİTAK ULAKBİM for their comments and feedback through this work.

References

1. Alicherry, M., Keromytis, A.D.: Doublecheck: Multi-path verification against man-in-the-middle attacks. In: Computers and Communications, 2009. ISCC 2009. IEEE Symposium on. pp. 557–563. IEEE (2009)
2. Braun, J., Volk, F., Buchmann, J., Mühlhäuser, M.: Trust views for the web pki. In: Public Key Infrastructures, Services and Applications, pp. 134–151. Springer (2014)
3. CAIDA: AS Relationships (2015), `http://www.caida.org/data/as-relationships/`
4. CAIDA: Center for applied Internet data analysis (2015), `http://www.caida.org`
5. Clark, J., van Oorschot, P.C.: SSL and HTTPS: Revisiting past challenges and evaluating certificate trust model enhancements. In: Security and Privacy (SP), 2013 IEEE Symposium on. pp. 511–525. IEEE (2013)

6. Comodo: Comodo SSL affiliate the recent RA compromise (March 2011), https://blog.comodo.com/other/the-recent-ra-compromise/
7. Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., Polk, W.: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard) (May 2008), http://www.ietf.org/rfc/rfc5280.txt, updated by RFC 6818
8. DetecTor, http://www.detector.io
9. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard) (Aug 2008), http://www.ietf.org/rfc/rfc5246.txt, updated by RFCs 5746, 5878, 6176, 7465, 7507, 7568, 7627
10. Eckersley, P., Burns, J.: The (decentralized) SSL observatory. In: Invited talk at 20th USENIX Security Symposium (2011)
11. EFF: The sovereign keys project, https://www.eff.org/sovereign-keys
12. EFF: The EFF SSL observatory (2015), https://www.eff.org/observatory
13. Faloutsos, M., Faloutsos, P., Faloutsos, C.: On power-law relationships of the Internet topology. SIGCOMM Comput. Commun. Rev. 29(4), 251–262 (Aug 1999), http://doi.acm.org/10.1145/316194.316229
14. Freier, A., Karlton, P., Kocher, P.: The Secure Sockets Layer (SSL) Protocol Version 3.0. RFC 6101 (Historic) (Aug 2011), http://www.ietf.org/rfc/rfc6101.txt
15. Gao, L.: On inferring autonomous system relationships in the Internet. IEEE/ACM Trans. Netw. 9(6), 733–745 (Dec 2001), http://dx.doi.org/10.1109/90.974527
16. Holz, R., Riedmaier, T., Kammenhuber, N., Carle, G.: X. 509 forensics: Detecting and localising the SSL/TLS men-in-the-middle. In: Computer Security–ESORICS 2012, pp. 217–234. Springer (2012)
17. Huang, L.S., Rice, A., Ellingsen, E., Jackson, C.: Analyzing forged SSL certificates in the wild. In: Security and Privacy (SP), 2014 IEEE Symposium on. pp. 83–97. IEEE (2014)
18. The ICSI certificate notary (2015), https://notary.icsi.berkeley.edu/
19. Kranch, M., Bonneau, J.: Upgrading HTTPS in mid-air: An empirical study of strict transport security and key pinning. NDSS (2015)
20. Langley, A.: Public key pinning (2011), https://www.imperialviolet.org/2011/05/04/pinning.html
21. Langley, A.: Enhancing digital certificate security. Google Online Security Blog (January 2013), http://googleonlinesecurity.blogspot.com/2013/01/enhancing-digital-certificate-security.html
22. Langley, A.: Further improving digital certificate security. Google Online Security Blog (December 2013), http://googleonlinesecurity.blogspot.com/2013/12/further-improving-digital-certificate.html
23. Langley, A.: Maintaining digital certificate security. Google Online Security Blog (2014), http://googleonlinesecurity.blogspot.com/2014/07/maintaining-digital-certificate-security.html
24. Langley, A.: Maintaining digital certificate security. Google Online Security Blog (March 2015), http://googleonlinesecurity.blogspot.com/2015/03/maintaining-digital-certificate-security.html
25. Langley, A., Kasper, E., Laurie, B.: Certificate Transparency. RFC 6962 (Experimental) (2013), https://tools.ietf.org/html/rfc6962
26. Laurie, B.: Certificate transparency public, verifiable, append-only logs (2014), http://queue.acm.org/detail.cfm?id=2668154

27. Leyden, J.: Trustwave admits crafting SSL snooping certificate: Allowing bosses to spy on staff was wrong, says security biz. The Register (2012), http://www.theregister.co.uk/2012/02/09/tustwave_disavows_mitm_digital_cert/
28. Luckie, M., Huffaker, B., Dhamdhere, A., Giotsas, V., et al.: AS relationships, customer cones, and validation. In: Proceedings of the 2013 conference on Internet measurement conference. pp. 243–256. ACM (2013)
29. Marlinspike, M.: Convergence (2012), http://conergence.io
30. NetGeo: The Internet geographic database (2015), http://www.caida.org/tools/utilities/netgeo/
31. NLANR: The national laboratory for advanced network research (2006), http://www.caida.org/projects/nlanr/
32. Ries, S., Habib, S.M., Mühlhäuser, M., Varadharajan, V.: Certainlogic: A logic for modeling trust and uncertainty. In: Trust and Trustworthy Computing, pp. 254–261. Springer (2011)
33. Schlyter, J., Hoffman, P.: The DNS-based authentication of named entities (DANE) transport layer security (TLS) protocol: TLSA (2012)
34. Singel, R.: Law enforcement appliance subverts SSL. Wired News (2010), http://www.wired.com/2010/03/packet-forensics/
35. Sleevi, R., Evans, C., Palmer, C.: Public key pinning extension for HTTP (2015)
36. Slepak, G.: The trouble with certificate transparency (September 2014), https://blog.okturtles.com/2014/09/the-trouble-with-certificate-transparency/
37. Soghoian, C., Stamm, S.: Certified lies: Detecting and defeating government interception attacks against SSL (short paper). In: Financial Cryptography and Data Security, pp. 250–259. Springer (2011)
38. TACK: Trust assertions for certificate keys, http://tack.io
39. Routeviews peering status report. Tech. rep. (July 2015), http://www.routeviews.org/peers/peering-status-by-as.html
40. University of oregon route views project (2015), http://www.routeviews.org/
41. VASCO: Diginotar reports security incident (August 2011), https://www.vasco.com/company/about_vasco/press_room/news_archive/2011/news_diginotar_reports_security_incident.aspx
42. Wendlandt, D., Andersen, D.G., Perrig, A.: Perspectives: Improving SSH-style host authentication with multi-path probing. In: USENIX Annual Technical Conference. pp. 321–334 (2008)