Deployment Challenges in Log-Based PKI Enhancements

Stephanos Matsumoto
Carnegie Mellon University,
ETH Zurich
smatsumoto@cmu.edu

Pawel Szalachowski
ETH Zurich
psz@inf.ethz.ch

Adrian Perrig
ETH Zurich
perrig@inf.ethz.ch

ABSTRACT

Log-based PKI enhancements propose to improve the current TLS PKI by creating public logs to monitor CA operations, thus providing transparency and accountability. In this paper we take the first steps in studying the deployment process of log-based PKI enhancements in two ways. First, we model the influences that parties in the PKI have to incentivize one another to deploy a PKI enhancement, and determine that potential PKI enhancements should focus their initial efforts on convincing browser vendors to deploy. Second, as a promising vendor-based solution we propose *deployment status filters*, which use a Bloom filter to monitor deployment status and efficiently defend against downgrade attacks from the enhanced protocol to the current TLS PKI. Our results provide promising deployment strategies for log-based PKI enhancements and raise additional questions for further fruitful research.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and Protection*; K.6.0 [**Management of Computing and Information Systems**]: General—*Economics*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*Authentication*

Keywords

Public-key infrastructrures, deployment, Bloom filters

1. INTRODUCTION

Public-key infrastructures (PKIs) such as those used for the Transport Layer Security (TLS) protocol bind a web server's name to a public key that can then be used to set up an end-to-end encrypted connection between a client and the server. In the TLS PKI, a certificate authority (CA) signs a public-key certificate that binds a server name to a public key, and a client can verify this certificate during TLS connection setup. The public keys of trusted CAs are installed in the client's browser or operating system, allowing the client to authenticate the server by a chain of signed certificates from a trusted CA to the server.

EuroSec '15 April 21-24 2015, Bordeaux, France

ACM 978-1-4503-3479-2/15/04 ...$15.00.
http://dx.doi.org/10.1145/2751323.2751324.

However, the current TLS PKI is worryingly brittle. CAs are a single point of failure, and CA errors or compromises have resulted in unauthorized certificates being issued for sites such as Google, Facebook, Skype, Yahoo, Microsoft, and Mozilla [1, 10, 13, 18, 19, 26]. Such forged certificates are currently difficult to detect, and can be used for a variety of nefarious purposes, including theft of sensitive user data (such as credit card numbers) and government surveillance.

To combat these issues, *log-based proposals* such as Certificate Transparency (CT) [15] introduces public logs to monitor all certificates issued by CAs. These public logs provide transparency by ensuring that only publicly-logged certificates are accepted by clients, allowing any misbehavior by CAs, particularly the issuance of unauthorized certificates, to be detected by clients and servers alike. Other log-based schemes have proposed to further add resilience to compromise and features such as monitoring of revoked certificates [11, 22, 24].

However, with log-based proposals and with other proposed PKI enhancements, the deployment process requires careful study. In particular, we observe several shortcomings in our understanding of the deployment process:

- We lack insights about the incentives of parties in TLS and how they influence one another to enact changes to the PKI.
- The incentives for an individual entity to deploy public logs have not been enumerated.
- Proposals often do not consider the security benefits that they offer *during*, as opposed to after, the deployment process.

As a result, most log-based proposals have not gained widespread traction, and some suffer from *downgrade attacks*, in which an adversary makes a server falsely appear to have not yet deployed the PKI enhancement. This attack downgrades the security of the PKI to that of the legacy PKI, which is used instead of the enhancement.

In this paper, we take steps to address these shortcomings. We model incentives and mutual influences in today's TLS, studying how different parties in the PKI influence one another with respect to deployment, and the incentives that these parties have to deploy public logs. We then propose *deployment status filters (DSFs)*, a mechanism that uses Bloom filters to detect the deployment status of a PKI enhancement at a domain, thus protecting against downgrade attacks. DSFs amplify the incentives for domains to deploy a PKI enhancement and can be deployed by browser vendors, who are able to effectively incentivize other PKI parties to also adopt the enhancement.

While our work on tackling this important challenge is still preliminary, we make the following contributions in this paper:

- We formulate a model capturing how deployment decisions by various parties (users, domains, CAs, and browser vendors) affect other parties' deployment decisions.
- We make observations regarding the deployment of public

logs and effective deployment of log-based proposals.
- We propose DSFs, a browser vendor-driven mechanism for determining deployment status and incentivizing deployment while preventing downgrade attacks.

2. BACKGROUND

In this section we provide background information on log-based PKI proposals for readers. In particular, we describe the parties in these proposals and provide an overview of the general log-based approach. The terminology we introduce in this section will be used throughout the rest of the paper. We then conclude the section with a short description of recent log-based proposals and their attempts to mitigate downgrade attacks.

2.1 Parties

Figure 1 shows the parties in a log-based PKI proposal and interactions between them that we will refer to in the rest of the paper.

PKIs are primarily concerned with authenticating a *server* to a *client* during the TLS handshake. Often this authentication takes place in the client's browser, which is provided by a *browser vendor* (hereafter simply a *vendor*). As part of the handshake, the server sends the client a public-key certificate signed by a *certificate authority (CA)*. The CA is responsible for verifying the identity of the server before signing and issuing the server's public-key certificate [16]. For example, a CA issuing a certificate for yahoo.com is responsible for making sure that an authorized representative of yahoo.com is indeed requesting the certificate, and that only yahoo.com holds the private key corresponding to the public key contained in the certificate.

Log-based proposals introduce several new parties to the PKI. One of these parties is the *public log*, which maintains a publicly-visible database of CA operations. While the exact mechanism of how this database is structured varies among proposals, all of the proposals discussed below store their databases in a Merkle Hash Tree (MHT) [17]. Public logs must track CA operations and respond to client queries regarding CA behavior, such as the presence of an observed certificate in the log's database. Additionally, logs that detect CA misbehavior may be required in some proposals to disseminate the evidence of the CA's misbehavior.

Some proposals additionally introduce *log monitors*, which watch the public logs to ensure that they are correct (only logging valid certificates) and consistent (not retroactively modifying certificates). This functionality is split into several parties in some proposals, and have different names in each proposal. The terms *auditor*, *monitor*, and *validator* all refer to entities that fall under the category of log monitors and primarily have the responsibility of tracking log behavior and corroborating proofs for responses provided by the logs. Additionally, some log monitors watch logs for suspicious certificates that may be the result of a compromise, and randomly request and verify log proofs to ensure proper log behavior.

2.2 Overview of Log-Based Proposals

We now give a brief overview of certificate issuance and validation in log-based proposals. A layout of the general log-based architecture is illustrated in Figure 1.

Log-based proposals add the extra step of registering a TLS certificate at a public log. Usually the domain or CA sends a certificate to the log for registration, but any party can submit a certificate for logging. A registration request is sent to the log, who checks the request and the certificate to make sure they are valid and responds with a message indicating that the certificate has been or will be logged. This message is signed and usually takes the form of a timestamp to indicate when the certificate was registered or a log proof, either of which can be used immediately..

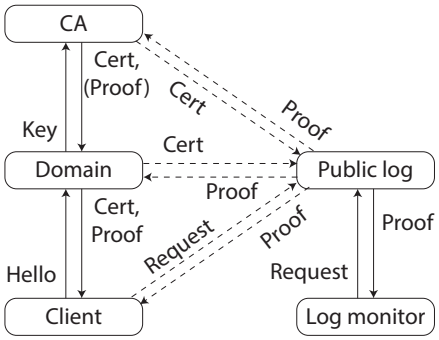

Figure 1: General architecture of the log-based PKI model. Solid lines indicate information sent during the certificate issuance and validation process, and dotted lines indicate the possible ways in which certificates are registered at the log, and how log proofs for a certificate can be obtained.

When a client initiates a TLS handshake with the domain, the client must also obtain a log proof for the domain's certificate or some other form of proof that the certificate was recorded in a public log. This proof is often provided by the domain itself, but may be retrieved directly from a public log. The client then verifies both the certificate and the proof to ensure that a valid chain from a root CA key to the domain's key exists and that the certificate has been checked and registered at a public log. Log proofs are also digitally signed by public logs and are thus non-repudiable, allowing log misbehavior to be proven to third parties.

Logged certificates are stored in MHTs, which can provide an append-only property [5] guaranteeing that no previously-observed certificates were retroactively removed or altered. The MHT structure also allows log proofs to be verified efficiently (requiring logarithmic time in the number of stored certificates). Log monitors can check these proofs over time to ensure that logs are maintaining the append-only property, and that logs are not adding any suspicious certificates to the MHT.

2.3 Specific Log-Based Proposals

We now briefly summarize several recent log-based proposals in the literature. Because we address downgrade attacks in this paper, we focus on the attempts of these proposals to mitigate these attacks by providing *proofs of absence*, in which a log can prove that a domain's certificate has not been logged.

Certificate Transparency (CT) [15] is a project by Google to add transparency to CA operations, monitoring all issued certificates to detect suspicious certificates. In CT, a certificate must be valid and publicly logged in order for a client to accept the certificate. A certificate can be submitted to a log by the CA, the domain, or by a client who observes the certificate. *Certificate Issuance and Revocation Transparency (CIRT)* [22] adds revocation functionality to CT through a new MHT that allows logs to provide efficiently-verifiable proofs of absence as well as presence.

The *Accountable Key Infrastructure (AKI)* [11] focuses on enhancing the log-based PKI with support for revocation and aims to handle events such as the loss or compromise of a private key. To recover from catastrophic events such as trusted CA compromise or domain key loss, AKI uses *cool-off periods*, during which a certificate is logged and publicly visible, but not valid for authentication purposes. These cool-off periods ensure that the domain has time

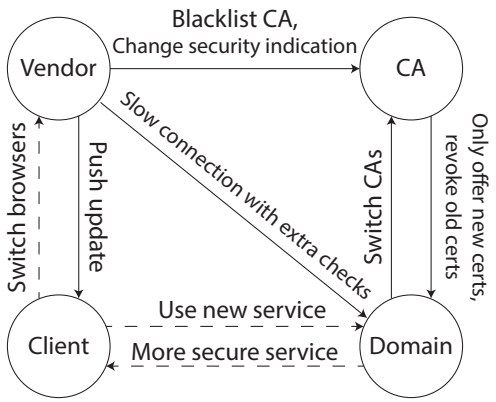

Figure 2: Mutual incentives. Solid arrows indicate a strong influence, while dotted arrows indicate a weak influence.

to detect a misissued certificate and contact the relevant CAs and logs to rectify the situation. In AKI, certificates contain policies that specify trusted CAs for the domain, as well as cool-off periods for various events (such as key loss). Logs' MHTs in AKI are organized differently from those in CT, allowing them to efficiently prove the absence of a certificate as well as its presence, defending against downgrade attacks. This organization is also used in PoliCert [24] and ARPKI [4], whose logs are based on those of AKI.

3. DEPLOYMENT MODEL

In this section we describe our model for deployment efforts in log-based PKIs. We describe mutual influences among parties in the PKI. We then discuss and evaluate incentives for each party to deploy a public log, and conclude the section with our observations on the deployment process.

3.1 Mutual Influences

We consider the influences of each of the four main parties on each other, which are summarized in Figure 2. We classify these influences into *strong influences*, which can effectively force a party to deploy a proposal, and *weak influences*, which may not always incentivize deployment.

Vendors and CAs: vendors have a strong influence on CAs, since browsers are many users' primary interface to the TLS PKI. Vendors can refuse to acknowledge the authority of a CA in their browsers by excluding the CA from the browser's root CA store [20]. However, this influence is limited for some CAs due to *too-big-to-fail* CAs: vendors cannot remove the largest CAs without cutting off access to many TLS-enabled sites, which runs counter to browsers' goal of enabling connectivity to websites [3, 16].

Since almost all modern web browsers are EV-aware, vendors can also influence CAs by refusing to show the EV indicator for EV certificates from a specific CA (usually for security reasons). Google has proposed this approach in Chrome as part of its deployment of CT [14], but it should be noted that the official guidelines for EV certification are set by the CA/Browser Forum [9].

Vendors and domains: vendors can strongly influence domains in several ways. As with CAs, vendors can change security indicators for specific non-deploying domains. Vendors may also carry out extra checks such as retrieving log proofs, since for example anyone in CT can submit a certificate to a log, and thus even a non-deploying domain may have a log proof corresponding to its certificate. However, a non-deploying domain would not include this proof in a TLS handshake, requiring an extra round trip for the browser to query for and obtain the proof, slowing a client's connection to the domain. Even if the domain is not concerned about enhanced security and thus not incentivized by the browser's security indicator, the domain may be incentivized by the possibility of slowing connections from all clients using the deploying browser.

Vendors and clients: Vendors have a strong influence on clients. Many modern browsers have the capability to receive push software updates and install these updates without client interaction (mainly for security purposes [12]), and thus a deploying browser vendor could easily reach its clients with new PKI functionality. While this capability does not incentivize clients to deploy a PKI proposal, it helps deployment by allowing a single browser vendor to quickly deploy and upgrade the proposal on its clients' machines.

Clients have a weak influence on vendors. While clients do not often do so, they can switch away from browsers that do not meet their desired level of usability, security, or connection speed. Even small differences in page loading times can affect whether or not a client visits a website [21], and thus browsers that differ significantly in how quickly they load pages can cause clients to switch browsers. Additionally, security issues such as TLS warnings can affect connectivity to HTTPS sites. Many clients click through these warnings [2, 7], and thus a browser making it difficult or impossible to click through TLS warnings may cause its clients to switch to a different browser.

CAs and domains: CAs and domains have arguably the closest business relationship of any two parties in the current PKI due to the fact that almost all HTTPS domains purchase certificates from CAs, and these purchases represent a majority of the CAs' business. CAs and domains have a strong influence on each other through this business relationship. This economic connection also manifests itself in deployment incentives for a new PKI proposal. For example, a deploying CA can easily incentivize non-deploying domains to adopt a new scheme by selling certificates of the new proposal at a better price than legacy certificates, or the CA could stop selling legacy certificates altogether.

On the other hand, as the CA's customers, domains can also influence the behavior of CAs, and domains could incentivize a CA to adopt a new proposal as well. Since domains tend to choose CAs based on the services they bundle with the certificate [3], domains could simply switch CAs. As long as the CA is approved by the browser, clients can establish TLS connections with a domain regardless of which CA signs its certificate.

Domains and clients: domains and clients may also share a close relationship for business or data privacy reasons. Domains that deploy a PKI proposal can incentivize clients to deploy the new scheme by offering a more secure service, particularly when the domain stores private user data. For example, the domain can detect the version of browser the clients are using and encourage them to switch to the latest version on the domain's webpage.

Clients can also influence domains by switching to a different service that does offer the security benefits provided by deploying the new PKI, particularly if the domain provides a security-sensitive service such as e-banking. However, if there is no similar service to switch to, then the clients cannot exert this influence, and thus we claim that clients have a weak influence on domains.

3.2 Log Deployment Incentives

Log-based proposals envision that anyone in the Internet can deploy a log, but in practice operating a reliable log requires more than just the desire to do so. In general, a party must continually maintain a public log (though some interactions can be automated), and thus proposals such as CT and AKI expect some minimum availability from logs, with Google requiring 99% [14]. We there-

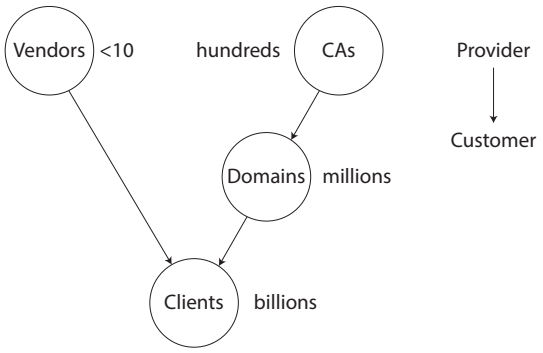

Figure 3: Provider-customer relationships in our model, with an approximate number of the size of each group.

fore consider the incentives that exist for various parties to deploy a public log and expend this continual effort.

Vendors: public logs operated by vendors can be useful for detecting misbehaving CAs. Vendors that detect misbehaving CAs in their logs can then block those CAs' public keys in their browser. For the purposes of protecting their users, vendors can also use their own logs to detect suspicious certificates rather than relying on logs operated by other parties, such as CAs, whose misbehavior may be correlated with those of the logs.

CAs: though many CAs today are not interested in deploying public logs,[1] providing a hurdle to global CA-driven deployment, logs operated by CAs have several advantages. CAs keep track of the certificates that they sign and issue, and can thus monitor logs for suspicious certificates, especially those that bear their signature. These suspicious certificates can be identified using systems such as Perspectives [25] or PoliCert [24]. CAs can also watch for suspicious certificates from other CAs, and expose this misbehavior in order to gain a competitive advantage in the CA market.

Domains: there is little incentive for a domain to operate a public log, since from a security perspective it can only help in protecting the domain itself along with a few high-profile sites. In particular, a domain can recognize its own TLS public-key certificates and detect misissued certificates for its own name, and the domain may also be able to do the same for several high-profile sites whose public keys are well-known. However, the operating costs in availability and bandwidth is likely to be higher than the benefit a log provides to the domain. In particular, the domain can simply monitor other logs to detect forged certificates for its name.

Clients: clients have almost no incentive to operate a public log, since like domains, they are not familiar with most sites' public keys and can simply query other logs to determine whether or not a certificate is valid and logged. However, unlike domains, they generally have no TLS public keys of their own and thus cannot even watch for suspicious certificates for even a single domain without knowing the domain's authentic key *a priori*.

3.3 Observations

We now present several observations that we have made from examining our model.

Provider-customer relationships closely map to strong influences. We observe the existence of several provider-customer relationships among the parties as shown in Figure 3. Users are customers of browser vendors and of domains, while domains are customers of CAs. These relationships map closely to strong influences, with providers exerting strong influences over their customers. Deploying at providers is likely to spur further adoption at their customers through their influences, and in particular, providers who are not customers (vendors and CAs in our model) are ideal choices to influence adoption in the rest of the PKI.

Without concerted action and adequate choice, customers generally have a weak influence on their providers. We observe that the influence of customers on their providers is limited to switching their business to a different provider. However, if a provider has little or no viable competition, such as a company that controls a large majority of its market share, then customers do not even have the influence of switching providers. Therefore, deploying a PKI proposal at a customer is unlikely to incentivize adoption at its provider unless the proposal is wildly successful.

On the other hand, providers have a much more direct influence on their customers. The customers are already interested in the provider's product or service, and increasing security by deploying a PKI proposal is unlikely to lose customers unless the change causes the existing product or service to suffer. For example, recall that browser vendors can deploy a PKI proposal for all of its users by simply integrating the proposal into the browser. Unless the change causes slowdown or reduced connectivity in the browser through excessive warnings or failures, users are unlikely to switch.

Deployment must reach domains and clients to be effective. PKI proposals ultimately aim to enhance the security of client-domain communication, and therefore deployment must take place at both domains and clients to be effective. In particular, domains must indicate in their certificates or during the handshake that they are using the PKI proposal. Clients or their browsers must deploy the proposal to gain security benefits, since all of the log-based proposals discussed in section 2.3 require the client browser to verify both the domain certificate and an SCT or log proof.

Deployment is best driven by vendors and CAs. Regarding deployment of public logs, both vendors and CAs have the resources to maintain a high-availability log server. They can leverage their logs to provide greater security to their own customers and to detect misbehavior by potential competitors. Indeed, the existing known logs in the ongoing deployment of CTs are operated by Google (a vendor) and several CAs.[2]

For deployment of the overall proposal, however, we argue that vendor-driven deployment is the most effective. While neither vendors nor CAs are customers of other entities, vendors have a strong influence over every other party. A vendor can add browser features that effectively deploy a proposal at all of its customers, and few vendors need to deploy a proposal to spur adoption among a majority of Internet users. Therefore, vendors should exert their strong influences over other parties to most effectively deploy a proposed PKI enhancement.

4. DEPLOYMENT STATUS FILTERS

In this section, we introduce DSFs, which allow client browsers to determine whether or not a domain has deployed a PKI enhancement. Full adoption of a PKI enhancement requires significant time and effort, and during the adoption process, knowing the deployment status of a domain is necessary to prevent downgrade attacks. We first provide an overview of DSFs and their operation, and describe how they fit into log-based PKIs and how clients utilize them to detect deployment status. We discuss the benefits of using DSFs and justify its approach as a viable strategy according to our mutual influences model from section 3. We then enumerate several challenges that remain in order to bring DSFs to fruition.

[1] http://www.certificate-transparency.org/feb-2014-survey-responses

[2] http://www.certificate-transparency.org/known-logs

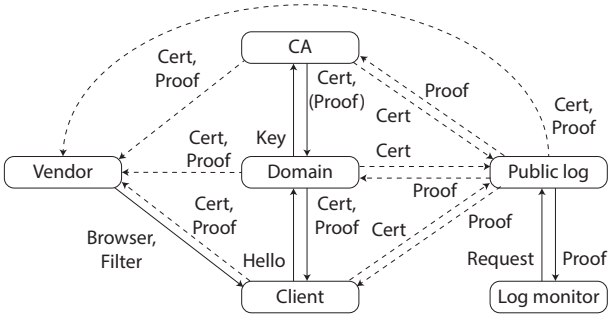

Figure 4: Overview of PKI with DSFs. Solid lines indicate information sent during normal log-based PKI operation, and dotted lines indicate possible ways for the certificate and proof to flow to the vendor.

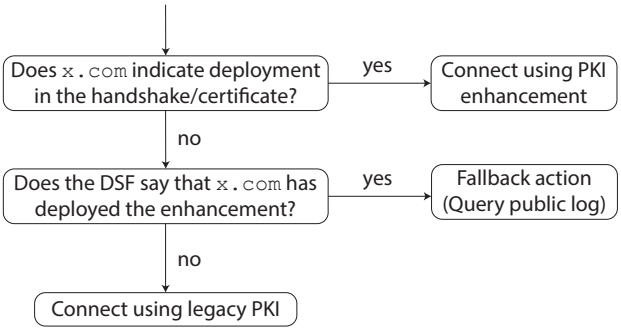

Figure 5: Flow chart for authentication of `x.com` using DSFs.

4.1 Operation

DSFs determine whether or not a domain has deployed a given PKI enhancement in order to prevent downgrade attacks. They are intended to be used during the early deployment process and may not be necessary after a majority of domains have deployed the enhancement and indicate this during the handshake (at which point log checking can be performed for each non-deploying domain). Vendors implement and maintain DSFs in client browsers, allowing clients to check the deployment status of a domain without contacting additional parties. DSFs are independent from absence proofs in proposals such as AKI and CIRT; while both allow clients to know whether or not a domain has deployed, DSFs can be checked by the client without querying any additional parties because they are stored in the browser rather than at an externally-located log.

Figure 4 illustrates the way in which DSFs fit into the log-based PKI ecosystem. The browser vendor maintains a Bloom filter and inserts the domain names of deploying domains into the filter. The vendor can obtain these names by receiving a certificate and corresponding log proof from the CA, domain, log, or client. Domains can query logs directly to obtain this information if the logs store the certificates along with the corresponding proofs. We envision that a majority of domains will be registered in a DSF by a CA or domain, since clients are not expected to track certificates and proofs themselves. Rather, vendors periodically push the filter in its current state to client browsers (such as with each new release of the browser or on a daily/weekly basis), so that all clients have up-to-date information on which domains have deployed.

During the process of deploying a PKI enhancement, a deploying client utilizing DSFs follows the decision process shown in Figure 5. The client checks for some indication of deployment from a domain, such as the inclusion of log proofs with the domain's TLS certificate. If the domain does not indicate that it has deployed the enhancement, clients who have DSFs in their browsers can check the domain's deployment status by querying the DSF's Bloom filter. If the domain has not deployed the enhancement, the client can simply continue the connection setup using the legacy PKI.

If on the other hand the DSF indicates that the domain has deployed the enhancement, there are several possibilities: (a) the domain has multiple certificates, only some of which use the enhancement, and is sending a certificate that does not use the enhancement; (b) the domain has not deployed the enhancement and is thus a false positive in the DSF; or (c) the domain is under a downgrade attack. In any of these cases, the client will take a *fallback action*, most commonly to query a log with the domain's name for certificates that may be logged.

The fallback action must check domain's deployment status carefully before proceeding with the connection, as it may be an attempted downgrade attack. By querying a log, the client can determine whether or not a logged certificate for the domain exists. If DSFs are used to monitor the deployment of a proposal such as AKI that provides proofs of absence, the client can obtain this proof from the log to ensure that the domain has not yet deployed the enhancement. If the domain has a logged certificate but did not send it to the client, the client should abort and retry the handshake to obtain the logged certificate.

Because public logs usually store certificates rather than domain names, logs must also offer a service to efficiently find certificates or their corresponding log proofs given a domain name. Logs can achieve this property by adopting an approach similar to CIRT [22], which combines MHTs with a binary search tree, to enable efficient domain-name-based lookups in public logs.

False positives in the DSF cause a fallback action such as querying a public log for a non-deploying domain, but the domain may have no corresponding proof in the log, effectively resulting in a wasted round trip to query the log. While these false positives amplify the incentives for domains to deploy the enhancement and indicate this fact in their certificates or TLS handshakes, vendors must change the hash functions regularly in their DSFs to avoid false positives for the same domains over subsequent time periods.

False negatives are also possible, since the browser may not immediately add newly-deploying domains to the DSF and push them to clients. However, we can assume that browsers will update their filters relatively frequently during the deployment process in order to minimize false negatives. Vendors can also provide advanced users with a feature to manually add certificates to the DSF in order to eliminate false negatives for individual users.

4.2 Benefits

We now describe several important benefits of DSFs.

Near-complete prevention of downgrade attacks. False negatives are only possible for domains that have deployed but have not yet been added to the DSF. Therefore, downgrade attacks are only possible for these domains, limiting the window of opportunity for adversaries aiming to carry out downgrade attacks. Otherwise, false negatives are not possible because DSFs make use of Bloom filters, and thus downgrade attacks, which rely on the DSF reporting that a deploying domain has not deployed the enhancement, are impossible for a registered domain.

Vendor-driven deployment. The vendor is an ideal party at which to implement DSFs. As seen in Figure 2, vendors have strong influence on all other parties, particularly on their browser clients, since vendors can simply push the filters to all browsers to

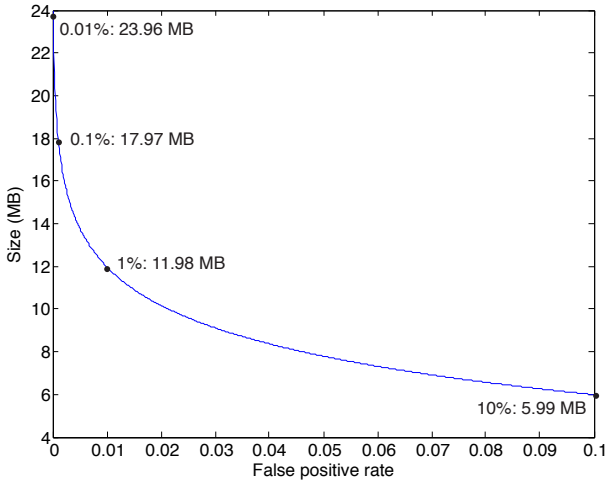

Figure 6: Optimal size of a DSF with 10M domain names for different false positive rates.

deploy the PKI enhancement throughout the Internet. DSFs also make vendors a bigger part of the PKI ecosystem. Though vendors take part in the current PKI by designing the user interface to TLS and by collaborating on important TLS standards such as EV certificates [8], their part in certificate verification is quite minor considering that the client carries out the verification process.

Incentivization of domains. The delay caused by a fallback action in the case of a non-deploying domain amplifies the incentives for domains to deploy a PKI enhancement. With DSFs, non-deploying domains will not have certificates in the filter and a client's fallback action may involve contacting more logs to retrieve log proofs, slowing connections to the non-deploying domains. False positives in the DSF will result in performing these extra checks for some non-deploying domains, but the false positive rate of the DSF's underlying Bloom filter can be tuned to a desired value, and the hash functions of the DSF's Bloom filter can easily be changed or randomized with each version of the DSF in order to ensure that a single domain is not always a false positive.

Size and ease of implementation. We expect that DSFs will be quite small by modern standards. Given a false positive rate p with n items in a Bloom filter, the number of bits m in a Bloom filter and the optimal number k of hash functions is given by

$$m = -\frac{n \ln p}{(\ln 2)^2},\ k = \frac{m}{n} \ln 2. \quad (1)$$

We assume a rough upper bound of 10 million certificates [6] each representing a unique domain name, though this number will likely be much less since DSFs are only intended to be used during the deployment process. Then for a 1% false positive rate the size of a DSF should be at most 13 MB, which is well below the storage limit of modern client machines. Figure 6 shows the storage requirements for a range of false positive rates, which even for a 0.01% false positive rate does not exceed 24 MB. Given that larger smartphone apps larger, we find this space requirement reasonable even for space-limited devices such as smartphones.

Browsers can easily change their hash functions with each iteration of the DSF. For example, if we let h be a hash function, h_i be the ith hash function of the DSF out of k functions as described in Equation 1, and j be the version of the DSF (with each update from the browser resulting in a new version number), then the browser can implement $h_i(d)$ for a domain name d as $h_i(d) = h(i \| j \| d) \bmod m$ where $\|$ denotes concatenation.

DSFs can also be easily implemented and integrated into the existing infrastructure. Browser-based mechanisms to enhance TLS are not new; for example, Google Chrome pushes CRLSets [12] to clients to ensure that clients do not accept revoked certificates. CRLSets use the existing Chrome update mechanism, and we envision that DSFs can similarly leverage the existing infrastructure for browser updates. Additionally, like CRLs, Bloom filters are a relatively simple data structure and thus can be quickly and easily implemented, reducing the deployment cost for vendors.

Efficiency and privacy. In light of proposals such as Let's Encrypt[3] that aim to vastly increase the amount of TLS certificates and traffic in the Internet, we envision that efficiency will be critical in checking deployment status to avoid high latency. An OCSP lookup, for example, takes almost 500 ms on average [23], and we envision that a log query will take a similar amount of time. Thus DSFs provide more efficient checking by only performing queries for sites that have likely deployed the enhancement.

DSFs also provide several advantages over purely log-based approaches such as proofs of absence. Without DSFs, clients who want to confirm that a domain has deployed an enhancement must query a log server to obtain a proof of absence. Not only does this extra query require an additional round trip, increasing page loading times, but this query also results in a loss of privacy for the client, since the log can see which domain's proof the client is requesting. Additionally, there may also be multiple logs that must be checked for consistency among logs, incurring an even greater latency cost than just one round trip. Thus DSFs provide a more efficient and privacy-oriented solution than proofs of absence.

5. REMAINING CHALLENGES

We now list several important challenges that remain to better understand the deployment efforts for log-based PKI enhancements.

Measuring influence. We hope to extend our model with metrics that can be quantitatively analyzed and use real-world technical and economic data to perform a detailed analysis of the extent to which the parties in the PKI ecosystem influence one another. In doing so, we hope to quantify the strength of mutual influences rather than simply classifying an influence as strong or weak.

Vendor/CA relations. The relationship between vendors and CAs is difficult to characterize, since they are not customers of each other, yet exert influence over each other and collaborate on important standards such as EV certificate guidelines. We hope to distill deeper insights from our data and gain a clearer understanding of mutual influences and incentives.

Tuning the false positive rate. False positives add unnecessary latency to a client's connection setup with a non-deploying domain. This extra latency can incentivize deployment and thus a higher false positive rate spurs rapid adoption of the enhancement; however, a false positive rate that is too high will result in an unpleasant user experience for the client, whose connections to a greater number of sites suffer from added latency. Thus we hope to explore this tradeoff in more detail and determine an optimal false positive rate.

Fallback actions. While we propose obtaining log proofs as the fallback action for the filter, there are many other possibilities for this action. A client could even specify policies that govern different fallback actions for various circumstances, similarly to how PoliCert allows domains to specify how TLS errors should be handled by the browser [24]. We hope to explore other fallback actions and their effects on deployment incentives.

[3] https://letsencrypt.org/

6. CONCLUSIONS

In this paper we have formulated a simple model to examine mutual influences in the PKI ecosystem, and analyzed this model to argue that browser vendors and CAs are ideal starting points for initial deployment efforts, and that CAs have the greatest incentive to deploy public logs. We have also proposed DSFs as a vendor-based solution to check the deployment status of domains and protect clients from downgrade attacks to regular TLS. Though there are still details to be worked out, our work has addressed an important problem in deploying enhancements to the current PKI, and identified future research topics that can further incentivize deployment that makes these much-needed PKI enhancements successful.

7. ACKNOWLEDGMENTS

We would like to thank Raphael M. Reischuk for his feedback and discussions of this paper. We also gratefully acknowledge support from ETH Zurich, the National Science Foundation under grant DGE-1252522, and a gift from Google.

8. REFERENCES

[1] Comodo fraud incident 2011-03-23. `https://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html`, March 2011.

[2] Devdatta Akhawe and Adrienne Porter Felt. Alice in Warningland: A large-scale field study of browser security warning effectiveness. In *Usenix Security*, pages 257–272, 2013.

[3] Hadi Asghari, Michel J. G. van Eeten, Axel M. Arnbak, and Nico A. N. M. van Eijk. Security economics in the HTTPS value chain. In *Twelfth Workshop on the Economics of Information Security (WEIS 2013)*, November 2013.

[4] David Basin, Cas Cremers, Tiffany Hyun-Jin Kim, Adrian Perrig, Ralf Sasse, and Pawel Szalachowski. ARPKI: Attack Resilient Public-key Infrastructure. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, November 2014.

[5] Scott A. Crosby and Dan S. Wallach. Efficient data structures for tamper-evident logging. In *USENIX Security Symposium*, pages 317–334, August 2009.

[6] Antoine Delignat-Lavaud, Martín Abadi, Andrew Birrell, Ilya Mironov, Ted Wobber, and Yinglian Xie. Web PKI: Closing the gap between guidelines and practices. In *Proceedings of the 21st Annual Network and Distributed System Security Symposium*, 2014.

[7] Adrienne Porter Felt, Robert W Reeder, Hazim Almuhimedi, and Sunny Consolvo. Experimenting at scale with Google Chrome's SSL warning. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pages 2667–2670. ACM, April 2014.

[8] CA/Browser Forum. Guidelines for the issuance and management of extended validation certificates (v. 1.0). `https://cabforum.org/wp-content/uploads/EV_Certificate_Guidelines.pdf`, June 2007.

[9] CA/Browser Forum. Guidelines for the issuance and management of extended validation certificates (v. 1.5.2). `https://cabforum.org/wp-content/uploads/EV-V1_5_2Libre.pdf`, October 2014.

[10] Hans Hoogstraaten, Ronald Prins, Daniël Niggebrugge, Danny Heppener, Frank Groenewegen, Janna Wettink, Kevin Strooy, Pascal Arends, Paul Pols, Robbert Kouprie, Steffen Moorrees, Xander van Pelt, and Yun Zheng Hu. Black Tulip: Report of the investigation into the DigiNotar certificate authority breach. `www.rijksoverheid.nl/bestanden/documenten-en-publicaties/rapporten/2012/08/13/black-tulip-update/black-tulip-update.pdf`, August 2012.

[11] Tiffany Hyun-Jin Kim, Lin-Shung Huang, Adrian Perrig, Collin Jackson, and Virgil Gligor. Accountable Key Infrastructure (AKI): A Proposal for a Public-Key Validation Infrastructure. In *Proceedings of the International World Wide Web Conference (WWW)*, May 2013.

[12] Adam Langley. Revocation checking and Chrome's CRL. `https://www.imperialviolet.org/2012/02/05/crlsets.html`, February 2012.

[13] Adam Langley. Enhancing digital certificate security. `http://googleonlinesecurity.blogspot.ch/2013/01/enhancing-digital-certificate-security.html`, January 2013.

[14] Ben Laurie. Improving the security of EV certificates, December 2014.

[15] Ben Laurie, Adam Langley, and Emilia Kasper. Certificate transparency. `https://tools.ietf.org/html/rfc6962`, June 2013.

[16] Stephanos Matsumoto and Raphael M. Reischuk. Certificates-as-an-Insurance: Incentivizing accountability in SSL/TLS. *Proceedings of the NDSS Workshop on Security of Emerging Network Technologies (SENT '15)*, February 2015.

[17] Ralph C. Merkle. A digital signature based on a conventional encryption function. In Carl Pomerance, editor, *Advances in Cryptology – CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pages 369–378. Springer Berlin Heidelberg, 1988.

[18] Microsoft. Erroneous verisign-issued digital certificates pose spoofing hazard. `https://technet.microsoft.com/library/security/ms01-017`, March 2001.

[19] Elinor Mills and Declan McCullagh. Google, Yahoo, Skype targeted in attack linked to Iran. `http://www.cnet.com/news/google-yahoo-skype-targeted-in-attack-linked-to-iran/`, March 2011.

[20] Jonathan Nightingale. DigiNotar removal follow up. `https://blog.mozilla.org/security/2011/09/02/diginotar-removal-follow-up/`, September 2011.

[21] Forrester Research. eCommerce web site performance today, August 2009.

[22] Mark D Ryan. Enhanced certificate transparency and end-to-end encrypted mail. *Network and Distributed System Security Symposium (NDSS)*, February 2014.

[23] Emily Stark, Lin-Shung Huang, Dinesh Israni, Collin Jackson, and Dan Boneh. The case for prefetching and prevalidating TLS server certificates. In *NDSS*, 2012.

[24] Pawel Szalachowski, Stephanos Matsumoto, and Adrian Perrig. Policert: Secure and flexible TLS certificate management. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 406–417. ACM, 2014.

[25] Dan Wendlandt, David G. Andersen, and Adrian Perrig. Perspectives: Improving SSH-style host authentication with multi-path probing. In *USENIX Annual Technical Conference*, June 2008.

[26] Michael Zusman and Alexander Sotirov. Sub-prime PKI: Attacking extended validation SSL. *Black Hat Security Briefings, Las Vegas, USA*, 2009.