Measurement and Analysis of Private Key Sharing in the HTTPS Ecosystem

Frank Cangialosi* Taejoong Chung† David Choffnes† Dave Levin*
Bruce M. Maggs‡ Alan Mislove† Christo Wilson†

*University of Maryland †Northeastern University ‡Duke University and Akamai Technologies

ABSTRACT

The semantics of online authentication in the web are rather straightforward: if Alice has a certificate binding Bob's name to a public key, and if a remote entity can prove knowledge of Bob's private key, then (barring key compromise) that remote entity must be Bob. However, in reality, many websites—and the majority of the most popular ones—are hosted at least in part by third parties such as Content Delivery Networks (CDNs) or web hosting providers. Put simply: administrators of websites who deal with (extremely) sensitive user data are giving their private keys to third parties. Importantly, this sharing of keys is undetectable by most users, and widely unknown even among researchers.

In this paper, we perform a large-scale measurement study of key sharing in today's web. We analyze the prevalence with which websites trust third-party hosting providers with their secret keys, as well as the impact that this trust has on responsible key management practices, such as revocation. Our results reveal that key sharing is extremely common, with a small handful of hosting providers having keys from the majority of the most popular websites. We also find that hosting providers often manage their customers' keys, and that they tend to react more slowly yet more thoroughly to compromised or potentially compromised keys.

1. INTRODUCTION

Online, end-to-end authentication is a fundamental first step to secure communication. On the web, Secure Sockets Layer (SSL) and Transport Layer Security (TLS)[1] are responsible for authentication for HTTPS traffic. Coupled with a Public Key Infrastructure (PKI), SSL/TLS provides verifiable identities via certificate chains and private communication via encryption. Owing to the pervasiveness and success of SSL/TLS, users have developed a natural expectation that, if their browser shows that they are connected to a website with a "secure" lock icon, then they have a secure end-to-end link with a server that is under that website's sole control.

However, the economics and performance demands of the Internet complicate this simplified model. Web services benefit from not only deploying content on servers they control, but also employing *third-party hosting providers* like Akamai, CloudFlare, and Amazon's EC2 service to assist in delivering their content. Many of the world's most popular websites are hosted at least in part on Content Delivery Networks (CDNs) so as to benefit from worldwide deployment and low-latency connectivity to users. Less popular websites are also often served by third-party hosting providers, in part to avoid having to set up and maintain a server and the associated infrastructure on their own. These hosting arrangements are often non-obvious to users, and yet, with HTTPS, they can have profound security implications.

Consider what happens when a user visits an HTTPS website, `example.com`, served by a third party such as a CDN: the user's TCP connection terminates at one of the CDN's servers, but the SSL/TLS handshake results in an authenticated connection, convincing the user's browser that it is speaking directly to `example.com`. The only way the server could have authenticated itself as `example.com` is if it had one of `example.com`'s private keys. This is precisely what happens today: *website administrators share their private keys with third-party hosting providers*, even though this violates one of the fundamental assumptions underlying end-to-end authentication and security—that all private keys should be kept private.

Such sharing of keys with CDNs has been pointed out by prior work, notably by Liang et al. [23]. However, the prevalence of key sharing, and its implications on the security of the HTTPS ecosystem, have remained unstudied and difficult to quantify. Moreover, websites share their private keys with a much broader class of third-party hosting providers than just CDNs, including cloud providers like Amazon AWS and web hosting services like Rackspace. The extent to which hosting providers play an active role in managing or accessing their customers' keys varies across provider and type of service—as we will see, for instance, some CDNs go so far as to manage their customers' certificates on their behalf. Whatever the role, merely having physical access to a website's private key can have severe security implications. We therefore consider a domain to have "shared" its private key if we infer that the private key is hosted at an IP address belonging to a different organization than the one that owns the domain (see §2.3).

In this paper, we quantify private key sharing within the HTTPS ecosystem at an Internet-wide scale, with two high-

[1]TLS is the successor of SSL, but both use the same certificates. We refer to "SSL certificates," but our findings apply equally to both.

CCS'16, October 24 – 28, 2016, Vienna, Austria

ISBN 978-1-4503-4139-4/16/10...$15.00

DOI: http://dx.doi.org/10.1145/2976749.2978301

level questions in mind: (1) *to what extent do websites trust third parties with their private keys?* and (2) *what implications does key sharing have on the management of the certificates?*

Answering these questions has required us to develop a set of new measurement techniques to determine which hosting provider has access to a given certificate, and whether two domains belong to the same organization (company, government entity, etc.). We apply these techniques to Internet-wide scans of SSL certificates, along with a confluence of other datasets, to perform the first *large-scale* study of key sharing within the HTTPS ecosystem.

Our results paint a grim, yet nuanced picture of the trust relationships in the web's PKI. We find for instance that over 76% of all organizations share at least one of their private keys with a third-party hosting provider, and that as a result, compromising the most popular such hosting provider could provide access to the private keys for 60% of the domains in the Alexa top-1K. Moreover, we observe many instances where hosting providers not only gain access to their customers' private keys, but they also take on the responsibility of *managing* their certificates. Interestingly, we find that this outsourcing of certificate management often leads to *better* certificate management, as measured by more thorough certificate revocation and reissuing behavior.

Our findings build upon a large body of work on measuring the web's HTTPS ecosystem [15, 16, 19, 25, 33]. These prior studies focus primarily on the trust relationships between websites, the certificate authorities (CAs) who issue certificates, and the browsers that verify certificates. We complement these findings by exploring a heretofore overlooked yet key player in the HTTPS ecosystem: CDNs and other hosting providers. As we will show, trust relationships with hosting providers can be difficult to ascertain; whereas the relationship between between a CA and a website is made explicit in the website's SSL certificate, no formal relationship need be stated for a third party to host a website. Our techniques aim to shed light on these trust relationships.

In this paper, we make the following contributions:

- We present a set of novel techniques that apply a confluence of datasets to obtain a clearer picture of who *owns*, who *serves*, and who *manages* which certificates.
- We apply these techniques to perform the first ever Internet-wide analysis of private key sharing between websites and third-party hosting providers, and show that sharing is prevalent, that it is driven by economic factors, and that a small number of hosting providers have aggregated a large stockpile of private keys.
- We show that many websites *outsource* management of their certificates to third parties, and we present the first empirical evidence that self-managed certificates tend to be *less secure* than those managed by third parties.
- We make all of our code and resulting datasets publicly available at `https://securepki.org`.

The rest of this paper is organized as follows. We provide a background on SSL/TLS and third-party hosting providers in §2. We then describe in §3 the various datasets we use. In §4, we present the measurement techniques that we apply in our study of key sharing (§5) and key management (§6). We present related work in §7 and conclude in §8.

2. BACKGROUND

Key sharing in the web's PKI is facilitated by recent additions to the SSL protocol and extensions to X.509 certificates. Here, we describe these mechanisms, and why using them for key sharing violates the spirit behind their design.

2.1 SSL certificates

An SSL certificate is a signed attestation binding a *subject* to a public key. Valid certificates are issued by Certificate Authorities (CAs), who in turn have their own certificates, and so on, terminating at a small set of self-signed root certificates. Thus, there is a logical *chain* of certificates—leading from a root certificate through zero or more *intermediate* certificates, to a *leaf* certificate—wherein the certificate at level i is signed with the private key corresponding to the certificate at level $i-1$ (with the exception of the self-signed certificate at the root). On the Internet, X.509 [4] is the most commonly used certificate management standard, and these certificates are commonly used as part of the SSL/TLS protocol (e.g., in HTTPS, IMAPS, etc).

In SSL certificates, the subject is contained in the `Common Name` field; for leaf certificates, the `Common Name` is a domain name (e.g., `www.example.com`). SSL certificates also allow *wildcard* domains in the `Common Name`, so a certificate with a `Common Name` of `*.example.com` would cover both `foo.example.com` and `bar.example.com`. Thus, when a client contacts a server, it is necessary to verify that the domain name the client intended to contact is in the `Common Name` of the certificate. If this is not the case, the client should reject the connection, as it may have been intercepted by a third party (i.e., a man-in-the-middle attack).

The original SSL protocol required that the server present its certificate without knowing which domain name the client was contacting. This effectively prevented servers from supporting more than one domain per IP address, as a server could only serve a single certificate per IP address, and each certificate could contain only a single `Common Name`.[2] As a result, two extensions to the X.509 certificate specification and TLS protocol were developed:

`SAN` list The `Subject Alternate Names` (`SAN`) extension allows a certificate to specify multiple alternate domain names to which the certificate should apply, effectively allowing a certificate to have multiple `Common Name`s. For instance, a certificate with a `SAN` list [`*.google.com`, `*.youtube.com`] would be accepted for both `www.google.com` and `m.youtube.com`.

SNI The `Server Name Indication` (SNI) extension to the TLS protocol allows a client to specify which domain it is trying to contact before the server presents its certificate. If both the client and the server support SNI, this allows the server to host SSL certificates for different domains on a single IP address; the server simply examines the SNI field to select which certificate it should send to the client.

2.2 Hosting providers

Many of our findings in the paper apply to web hosting services and Content Delivery Networks (CDNs), which we collectively refer to as *hosting providers*. Popular websites now commonly use hosting providers for distributing con-

[2] Modulo wildcard `Common Name`s, which allow the server to serve multiple sub-domains from a single domain using only a single IP address.

tent and, frequently, for providing security services such as denial-of-service attack mitigation [17].

Websites are increasingly using third-party providers to host HTTPS content, but unfortunately, limitations of the TLS protocols have made this challenging for hosting providers. TLS has historically assumed that a given IP address would be used to host only a single website's domains, and that therefore it would suffice for any given IP address to serve a single certificate. To support multiple customers' HTTPS content, hosting providers generally use one or more of the following approaches:

One customer per IP address The most straightforward approach involves having a single customer's certificate (possibly with multiple domains in a `SAN` list) allocated to any given IP address. This has the benefit of not requiring clients to support SNI, but comes at a high monetary cost, as IPv4 addresses have grown more scarce.

Multiple certificates per IP address Alternatively, hosting providers can host multiple certificates on any given IP address by using SNI. This is less expensive than dedicating an IP address to a single customer, but unfortunately older clients like Internet Explorer on Windows XP and Android 2.x devices do not support SNI. Hosting providers are often hesitant to implement a solution that leaves these clients unable to access customer websites.[3]

"Cruise-liner" certificates Finally, if a hosting provider can obtain custom certificates on behalf of its customers [23], it can craft certificates with `SAN` lists containing domains from multiple *distinct customers*. One such `SAN` list we observe contains `monsanto.com` (an agrochemical corporation), `aaa.com` (an automobile association), and `jazzercise.com` (a workout program), along with dozens of other companies.[4] This approach lets the hosting provider use a single certificate per IP address (therefore not requiring clients to support SNI) *and* support multiple customers per IP address (therefore not requiring purchase of many IP addresses). However, there are also downsides [23], such as not being able to support Extended Validation certificates. We refer to these as *cruise-liner certificates*, as multiple distinct customers share a ride with one another.[5]

2.3 What we mean by "key sharing"

The overall goal of this paper is to quantify when one party has made its certificate's private key available to another party. In general, it is difficult to ascertain when this has occurred as an outsider, so we rely on the evidence we do have available to us: the IP address(es) we observe advertising the certificate. Specifically, we say that *key sharing* has taken place if any of the parties named in a certificate (the `Common Name` or entries in the `SAN` list) are not the same as the organization who owns the IP address from which it is advertised.

This is a rather coarse-grained definition of key sharing, and the true relationships between the owner of the key and the owner of the IP address can be subtler. In particular, our definition of key sharing captures three broad classes of behavior: *First*, a website may explicitly hand over its private keys to a hosting provider. For example, a customer may simply upload its certificate and private key, as is commonly done when websites subscribe to CDN services.

Second, a website may give its hosting provider physical access to its private keys, even though the website does not explicitly hand over the keys themselves. There are many different mechanisms by which this could be done. For example, a website may use a cloud-based virtual machine (where the provider has the ability to read the key from the VM's memory) or the website may use a co-location service (where the provider has the ability to physically access the website's servers). Our definition of "key sharing" encompasses both of these because the IP address is owned by the hosting provider but serves keys owned by its customers.

In all of these cases, the website is not only trusting the hosting provider not to access the keys, but also trusting the provider to prevent both external and internal attackers from accessing the keys. While websites may raise the difficulty of such attacks through the use of tamper-resistant hardware or software obfuscation, ultimately, they are trusting the provider to some degree. We are therefore comfortable with its inclusion in our definition of "key sharing.".

Third, a website may run its *own* servers within a third-party network, where the network operator has no physical access to the website's servers. For example, such a website could operate its own datacenter—with restricted physical access—all within another's network. This is arguably not "key sharing" since the network operator has no access to the website's key material; unfortunately, we are unaware of any way to determine at-scale and in an automated manner who has physical access to a given IP address. Thus, in the remainder of the paper, we identify cases of *potential* key sharing, though we believe that in the vast majority of cases, the owner of an IP address does have physical access to the machine using that address.

2.4 Why study private key sharing?

The security of any public key encryption system rests on keeping private keys private; sharing private keys across entities violates these assumptions. To quote the RFC for SSL certificates [4]:

> The protection afforded private keys is a critical security factor. On a small scale, failure of users to protect their private keys will permit an attacker to masquerade as them or decrypt their personal information. On a larger scale, compromise of a CA's private signing key may have a catastrophic effect.

Similarly, a single website choosing to share its private key with a hosting provider may seem relatively innocuous, but *large numbers* of websites sharing with a small number of hosting providers may lead to even greater centralization of trust than was previously realized. Prior work [23] showed that websites share keys with their CDNs, but the community at large has lacked the tools to measure the extent of key sharing and the implications it has had on the administration of private keys. This paper develops novel techniques and applies them to perform the first large-scale study of key sharing in the web's PKI. Our results expose trust relationships in the HTTPS ecosystem, complementing a large body of work (see §7) that has studied similar trust relationships between websites and CAs.

[3] `https://community.akamai.com/thread/1314#2168`

[4] `https://censys.io/certificates/74f611c7a9524673df03801be9778c96dcc31a6a807cb36e7a2d3b031cb805b2`

[5] Note that cruise-liner certificates cannot be modified piecemeal; adding or removing any customer's domain requires generating an entirely new certificate.

3. DATASETS

Our Internet-wide study of key sharing in the HTTPS ecosystem is driven by four datasets:

SSL certificates We use SSL certificates from full IPv4 scans as the basis of our measurements. We obtain our collection of SSL certificates from (roughly) weekly scans of port 443 over the entire IPv4 address space, made available by Rapid7 [30]. In this paper, we use 74 scans conducted between October 30, 2013 and March 30, 2015. Overall, we observe 38,514,130 unique SSL certificates.

It is worth noting that these scans cover all SSL certificates *except* those served solely using `SNI`, since the scans only obtain the default certificate from each IP address [32]. However, SNI is not yet used on a wide scale by major CDNs, and Akamai did not even offer `SNI` support for its customers until late 2015 [22] (months after the end of our scans). Thus, we believe the lack of `SNI` domains in our dataset is likely to not have a significant effect on our results.

We use a similar process as prior work [25] to separate valid and invalid certificates. In brief, starting with the root store from OS X 10.9.2 [27], we first use `openssl` to identify 1,946 valid intermediate (CA) certificates. We then identify 5,067,476 valid leaf certificates using these root and intermediate certificates, covering 2,552,936 unique domains (including domains in `SAN` lists).[6] We ignore invalid certificates in this paper [6].

Reverse DNS The Domain Name System (DNS) allows the owners of IP addresses to publish *reverse DNS* entries for each IP address. Many organizations publish reverse DNS information that provides clues as to the owner of the IP address.[7] For example, the DNS entry for `www.nest.com` (a home devices company owned by Google) points to `50.16.224.42`. The reverse DNS entry for that IP address maps to `ec2-50-16-224-42.compute-1.amazonaws.com`, telling us that Nest uses Amazon's EC2 service to serve their website.

Our SSL scans [30] also contain information on the IP address(es) that advertised each certificate. To obtain information about the entity that controls this IP address, we use full IPv4 reverse DNS scans [29] that are also conducted by Rapid7. Unfortunately, the DNS standard does not require address owners to provide reverse DNS entries—let alone informative ones—and is only a recommended (though common) practice.

AS Number and Organization Because reverse DNS entries are not always available, we use additional information to fill in these gaps. Ownership of IP address space is divided up at the highest level across *autonomous systems* (ASes), each representing a network under the control of a single entity. Each AS is assigned an AS Number (ASN): for example, MIT is AS 3 and the Chicago Public Schools are AS 1416 [26]. CAIDA collects and publishes mappings between IP addresses and ASNs via their RouteViews datasets [7]. We download daily snapshots of these mappings to determine the ASNs an IP address is in, if it has no reverse DNS entry. Accurate IP-to-AS mapping remains an open problem [5]; we use RouteViews because it provides historical data and likely works well for end-hosts.

Additionally, a given organization may have multiple ASNs under its control. For example, AT&T owns 160 unique ASNs. To aggregate these, we use CAIDA's AS-to-Organization dataset [8] to group together ASes owned by the same organization. By combining these datasets, we obtain a mapping from an IP address to the organization that advertises a route to that IP address.

WHOIS The datasets described thus far reveal information about the IP addresses that are advertising SSL certificates, but not about the *domains* present in the certificates. For that, we rely on WHOIS [12], a protocol for querying domain registrars to obtain data on the domain owner. In practice, WHOIS data often contains fields such as the contact information for the owner of the domain, the contact for technical issues, where to send abuse complaints, and so on. These often take the form of email addresses, and any given WHOIS record can have multiple points of contact.

Unfortunately, the WHOIS infrastructure is distributed across registrars and resellers, and there is no standard format [24]. Additionally, obtaining WHOIS data at scale is challenging, as most registrars rate-limit queries. Thus, we obtain our WHOIS data on the 2.5M domains from two sources:

1. **Liu et al. [24]** In prior work [24], Liu et al. built a parser for WHOIS records. We obtained a copy of their `.com` dataset, covering 985,517 WHOIS domain records that appeared in our certificates.

2. **Bulk WHOIS services** We used two bulk WHOIS data sources[8] to obtain WHOIS information on the remaining domains. Through these two, we were able to obtain WHOIS information for 1,779,308 domains.

In the end, we were able to obtain WHOIS information for 2,197,292 (86.0%) of our domains. The domains where we were unable to find WHOIS information were typically domains that have either expired or whose registrars did not publish WHOIS information.

4. METHODOLOGY

The central goal of this paper is to empirically study the prevalence and ramifications of key sharing in the HTTPS ecosystem. We would like to answer questions such as:

- How many website organizations share their private keys with third-party hosting providers?
- How many distinct organizations appear in a given certificate's `SAN` list?
- Which third-party hosting providers manage their customers' certificates?

To answer these questions and others, we must have a way to determine: (§4.1) whether two domains belong to the same organization, (§4.2) which hosting provider serves a given certificate, and (§4.3) whether a given hosting provider is a third-party provider or the website itself (a "first-party" provider).

[6] When referring to a *domain* in this paper, we mean one level beyond the Top Level Domain. For example, `www.example.com`, `example.com`, and `foo.bar.example.com` are all in the `example.com` domain.

[7] The reverse DNS record for IP address `a.b.c.d` is stored under a special `PTR` record at `d.c.b.a.in-addr.arpa`.

[8] `http://bulkwhoisapi.com` and `http://whoisxmlapi.com`, both of which charge for their data.

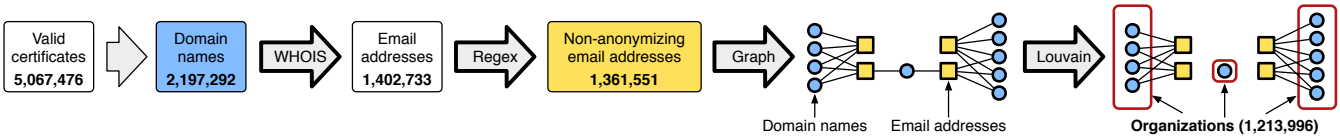


Figure 1: To determine which domains belong to the same organization, we construct a bipartite graph between all domain names and the (non-anonymizing) email addresses from their WHOIS records. We remove spurious edges (such as those from some registrars) by iteratively applying Louvain community detection. The resulting clusters of domain names correspond to distinct organizations.

Organization	*#Domains*	*Examples*
Nestlé	788	`nestle.com`, `nestle.com.sg`, `dogchow.ca`, `purinaone.co.nz`, `polandspring.com`, ...
Google	338	`google.com`, `google.hk`, `golang.org`, `blogspot.com`, `zagat.com`, `madewithcode.com` ...
Reuters	258	`reuters.com`, `thomsonscientific.com`, `manuscriptcentral.com`, `trust.org`, `techstreet.com`...
Disney	228	`disney.com`, `babyzone.com`, `abcdmedia.com`, `disneyunitedway.com` `disneycareers.com`, ...
Univ. of Maine System	9	`uma.edu`, `umaine.edu`, `lewiston.k12.me.us`, `machias.edu`, `umfk.edu`, ...

Table 1: Examples of domains owned by the same organization that are linked by our methodology. (These are not the top five groups.)

It is surprisingly difficult to determine if two domains have the same owner, and whether a domain is hosted by a third party; there is no global support to ask such queries, and we are unaware of a dataset that captures them at the scale our study demands. In this section, we present novel techniques to develop such a dataset. We apply them to understand key sharing (§5) and management (§6), but we believe our techniques to be broadly applicable.

4.1 Determining who owns a domain

The first technique we present determines whether two domain names are owned by the same organization. This is an important tool in our study because it allows us to identify cruise-liner certificates (as opposed to certificates with many domain names from a single organization, as with Google). Also, reporting on how many *organizations* share their keys avoids over-inflating numbers—a single organization's decision to use a third-party hosting provider could result in all of its domains' keys being shared, and some organizations own hundreds of domains.

4.1.1 Certificate scans are not enough

Traditionally, studies of the HTTPS ecosystem treat a given certificate or a given `Common Name` as the measurable unit [14, 15, 19, 25, 35], but the mapping between organizations and certificates/domains is not so clear in practice. *First*, a given organization may have many distinct domains, some of which are easily grouped together by inspecting the domain names themselves (e.g., Google is the administrative authority of `google.com`, `google.co.uk`, `google.de`, and so on) while others are less straightforward (Google is also the administrative authority for `gstatic.com`, `youtube.com`, and `blogspot.com`). These domains sometimes appear on the same certificate in the form of `SAN` lists, but they can also appear on separate certificates altogether. In other words, there may be one organization that maps to multiple domains, and each of those domains may (separately or together) map to multiple certificates.

Second, we find that *many certificates contain more than one organization*. This, too, is accomplished with `SAN` lists: the *spirit* of a `SAN` list is that it captures different names of the same entity, but in practice, some hosting providers will lump together multiple entities into the same `SAN` list (the "cruise-liner" certificates from §2.2).

Given these challenges, we conclude that our certificate scans dataset alone is not enough to infer which organizations host and appear on a given certificate. To this end, we complement this dataset with information from WHOIS.

4.1.2 Some WHOIS email addresses are too much

To determine whether two domains refer to the same organization, we inspect the email addresses contained within the domains' WHOIS records. Our intuition is that, if two domains share a contact email address, then they are likely administered by the same organization. However, we find that some email addresses in WHOIS records must be filtered, for two reasons:

First, many WHOIS records do not contain a uniquely identifying email address; a common practice is to register for a domain name through a privacy-preserving service, with the goal of hiding the contact information of the domain's owner. For example, we see 14,145 unique domains in our dataset that have the email address `contact@privacyprotect.org` in their WHOIS record. representing such a service. We therefore need to filter out such anonymizing email addresses, as domains that share them should not be linked together. Some domain privacy services provide a per-customer obfuscated email contact (e.g., `82af4cc@privacydomain.com`); we do not filter these, as they serve as a pseudonym for a single organization.[9]

Second, many domain registrars and hosting providers will place *their own* email address into the WHOIS record as one of the technical points of contact. For example, the domain registrar for the Tanzanian domain `.tz`, IT FARM, places their own email address of `support@itfarm.co.tz` into many of the `.tz` WHOIS records (including that of `google.co.tz`). These, too, need to be filtered, as the registrar's organization is distinct from their customers'

4.1.3 Domain ownership methodology

Our methodology for identifying whether two domains are owned by the same organization addresses the above challenges through four key steps, corresponding to the block arrows in Figure 1:

[9] We infer that these pseudonymous email addresses map to organizations, and not domains, because 87.2% of such email addresses map to more than one domain.

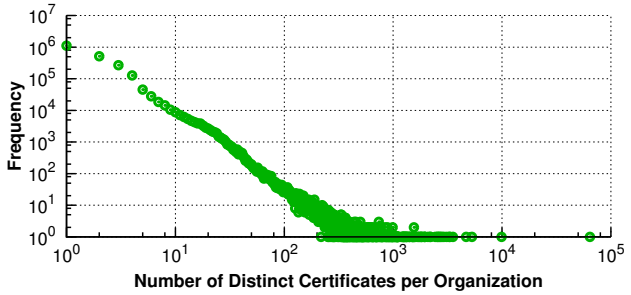


Figure 2: 49.9% of all organizations in our dataset appear on more than one certificate. The most frequently appearing organization, `CloudFlare Inc.`, appears on 65,700 distinct certificates, into which it inserts itself along with its customers.

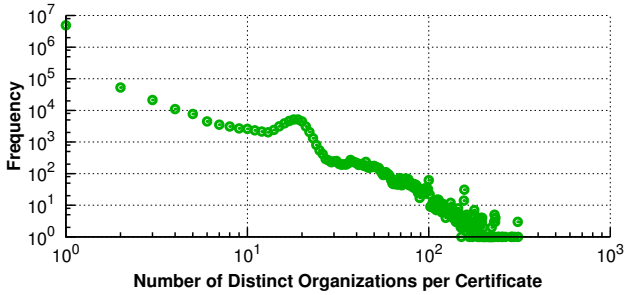


Figure 3: Although the vast majority (96.8%) of certificates in our dataset contain but a single organization (the expected behavior), 161,810 have two or more, and some certificates contain over *300* distinct organizations in their `SAN` lists.

1. **WHOIS extraction** We begin by extracting *all* email addresses that appear in the WHOIS record for each domain (recall from §3 that WHOIS records may contain multiple email addresses).

2. **Email sanitization** We then remove the email addresses of privacy-preserving WHOIS services. We first develop a blacklist (based on a regular expression[10] and a list of known services) to remove common patterns of privacy-preserving emails; these remove 40,745 email addresses. To further refine this dataset, we manually inspected each of the remaining 486 email addresses that appear in more than 100 WHOIS records[11] and identified additional WHOIS privacy services; this removed an additional 437 email addresses.[12]

3. **Graph construction** Next, we construct a bipartite graph between the 2,197,292 domains with WHOIS records and 1,402,733 non-anonymizing email addresses. We then find the connected components in this bipartite graph; each of these components represents a potential group of domains owned by the same organization.

4. **Community detection** Unfortunately, there are cases where too many domains are linked together due to registrars or hosting providers inserting their own email addresses into WHOIS records. We divide up each connected component by repeatedly applying the Louvain community detection algorithm [3]. Essentially, it finds tightly grouped clusters of domains within each component that are weakly connected to the rest of the component. We repeatedly apply the algorithm until either there is only a single domain/email address in the cluster, or when the domains *of the email addresses* are present in the cluster as well.

This algorithm produces clusters of domains, each cluster representing a separate organization. Clusters of size one correspond to organizations that control a single domain.

[10] `(private|privac|whois|abuse|cctld|^support@|@.*domain.*|@.*hosting.*|^nic[^A-Za-z]|^[^0-9][0-9]?[^0-9][0-9]?[^0-9]@)`

[11] We chose this threshold of 100 somewhat arbitrarily, though in our manual inspection, we found this to capture most of the common anonymizing services.

[12] While this is a small number of email addresses, when we construct the bipartite graph, they correspond to nodes with many edges across many components, and so their removal is critical to identifying organizations.

4.1.4 *Domain ownership results*

We apply this methodology to our dataset of 2.5M domains and obtain a set of 1,213,996 clusters of domains owned by a single organization.[13] We manually inspected the largest of these and found them to correspond to large governmental organizations and conglomerates; a few examples are shown in Table 1. Interestingly, we observe that our methodology captures both obvious cases of domains controlled by a single organization (e.g., `nestle.com` and `nestle.com.sg`) and non-obvious cases (e.g., `google.com` and `zagat.com`; Google recently acquired Zagat).

To briefly explore how the resulting organizations are spread across certificates, Figure 2 plots the number of certificates on which each organization appears. While just over 50% of organizations appear on a single certificate, some organizations appear on *thousands* of certificates. Figure 3 shows the number of unique organizations present per certificate. The vast majority (96.8%) of certificates have only a single organization present in them, but we observe 161,810 certificates with multiple organizations. The bump at ∼15–30 organizations per certificate is largely attributable to CloudFlare's use of cruise-liner certificates.

These results provide the first evidence that there is significant overlap of organizations across leaf SSL certificates. We explore this phenomenon in more detail in §5 and §6.

4.2 Determining a site's hosting providers

The next building block we need is the ability to determine which organizations host a given certificate. Our certificate scans dataset includes the 130,320,517 IP addresses from which we saw certificates being advertised. We seek a method to convert these IP addresses to organization names that we can compare to the organizations we infer in §4.2.

For each IP address in our dataset, we first try to locate its reverse DNS record. We are able to do so for only 69.8% of them (91,075,112). Often, these reverse DNS names include not only the organization name but also a unique identifier for the server we contacted; as we are only interested in linking organizations, we strip off their subdomains (e.g., `ec2-xxx.compute-1.amazonaws.com` becomes `amazonaws.com`).

For the remaining 30.2% of IP addresses (39,245,405), we look up the Autonomous System Number (ASN) and corresponding Organization Name (see §3). We manually ex-

[13] Technically speaking, our clusters represent groups of domains that share the same administrative point of contact. In practice, we have found these clusters map well to real-world entities.

amined the 100 most frequent such Organization Names, and only kept the 54 that we identified as third-party hosting providers; the others were either the domain-owning organizations themselves (e.g., Yahoo Japan) or ISPs (e.g., AT&T). This allows us to map 10,231,246 additional IP addresses to organization names. There are cases where a single hosting provider has multiple AS Organization Names (e.g., OVH shows up as both `OVH Systems` and `OVH SAS`). We manually inspected the list of the 54 AS Organization Names and merged them into 46 unique hosting providers.

Some hosting providers appear both in the reverse DNS list as well as in the AS Organization Name list (e.g., SoftLayer shows up as both `softlayer.com` and `SoftLayer Technologies Inc.`). We unified them to avoid overcounting third-party hosting providers. To do so, we first simplified the Organization Name by removing all generic words (e.g. `Peer 1 Network (USA) Inc.` becomes `Peer 1`), removing all non-alphanumeric characters, and converting all letters to lowercase (`Peer 1` becomes `peer1`). We then manually inspected all reverse DNS entries for which the simplified organization name was a substring of that entry (e.g. `peer1` is a substring of `mypeer1.com`), and manually removed any false positives (e.g., `sungard`, the simplification of `Sungard Availability Services LP.`, is a substring of `sungarden.com`, but they do not represent the same organization). For the remaining pairs of AS Organization Name and reverse DNS entry, we infer that they correspond to the same organization.

All together, we are able to map 101,306,358 IP addresses (77.7%) to hosting providers represented by either a reverse DNS domain or an AS organization. Table 2 shows the 20 inferred hosting providers with the most organizations as customers.

4.3 First- vs. third-party hosting

The techniques from §4.1 give us the organizations stored in each certificate, in the form of clusters of domain names, and the techniques from §4.2 give us the name of an organization that owns an IP address, either in the form of a domain name (if there is a valid reverse DNS entry) or an AS Organization Name. Here, we combine these to determine if a certificate is first- or third-party hosted.

At a high level, a certificate is first-party hosted if the certificate contains only one organization, and all of the IP addresses serving that certificate are owned by that same organization (see §2.3). Conversely, if there is more than one organization on the certificate (e.g., with cruise-liner certificates) or if any of the IP addresses from which it is served are owned by an organization not in the certificate, then we conclude it is third-party hosted.

When we have the reverse DNS names of the IP addresses hosting a certificate, comparing the hosting organization and the certificate organization is straightforward: we simply check whether the reverse DNS name is included in the cluster of domains from the algorithm in §4.1.

Otherwise, when all we can determine from the hosting IP address is the AS Organization Name, we determine if it matches any of the domain names from the certificate using the same process of matching AS Organization Names and domains as described in §4.2. For example, a certificate with a `Common Name` of `ssl12039.cloudflare.com` and an AS Organization Name of `CloudFlare Inc.` would match because the simplified name, `cloudflare`, is a substring of the `Common Name`. As in §4.2, we manually checked each equivalence, and removed spurious results.

#Organizations	*#Domains*	*Hosting Provider*
266,110	277,891	`secureserver.net`
151,628	175,089	`amazonaws.com`
113,400	118,460	`Unified Layer`
78,370	87,078	`CloudFlare Inc.`
64,370	74,966	`Rackspace Ltd.`
46,366	50,173	`SoftLayer Technologies Inc.`
33,676	37,657	`your-server.de`
31,051	32,324	`CyrusOne LLC`
26,956	28,875	`linode.com`
24,345	26,351	`SAKURA Internet Inc.`
20,742	22,075	`OVH SAS`
20,045	27,849	`Peer 1 Network (USA) Inc.`
17,473	18,020	`Digital Ocean Inc.`
15,643	22,732	`ClaraNET LTD`
15,530	17,081	`Liquid Web Inc.`
15,440	22,671	`akamaitechnologies.com`
14,907	15,997	`theplanet.com`
14,707	15,766	`comcastbusiness.net`
13,760	14,281	`1&1 Internet AG`
13,572	14,978	`cloud-ips.com`

Table 2: The top 20 hosting providers from our dataset with the most number of customers' private keys, and the number of domains they serve over HTTPS.

4.4 Summary

In this section, we presented a set of techniques that allow us to reason at the level of organization names (e.g., `amazonaws.com` and `CloudFlare Inc.`) about who has access to what keys and who hosts whose content. Though we lack a ground truth dataset to evaluate our techniques concretely, our manual inspection of their output found them to be highly accurate.[14]

Perhaps most frustrating of all, however, is that if conscientious users wished to understand with whom they are actually communicating, they would be faced with the same dearth of ground truth as us. In other words, because it is challenging even to determine who owns and who is hosting a particular domain, *the trust relationships between organizations and hosting providers is far from transparent.* In the remainder of this paper, we apply our new techniques to shed light on these trust relationships, beginning with the prevalence and ramifications of organizations sharing their private keys with hosting providers.

5. TRUST

Sharing private keys with a third party involves a great deal of trust: the nature of today's PKI is such that, with a domain's private key, one can impersonate that domain with arbitrary data. This has profound impact on the potential security and reliability of the PKI. While there have been many studies of the trust relationships between organizations and CAs, this is the first wide-scale study of trust relationships between organizations and hosting providers with respect to private cryptographic keys. We investigate these trust assumptions along several axes, including how common key sharing is, whether there are several large players who have aggregated many keys, and how transparent

[14] Our code and data, including this classification, is publicly available at `https://securepki.org`.

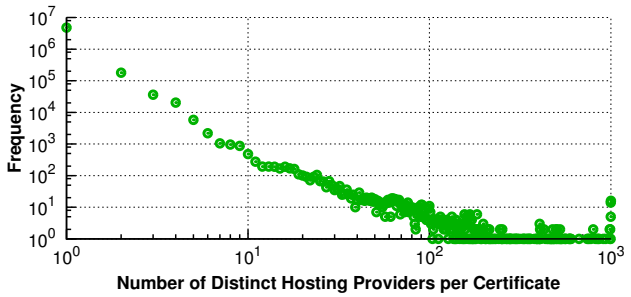


Figure 4: Frequency distribution of the number of hosting providers serving a given certificate. 91.2% of all certificates have a single hosting provider, but 363,043 certificates have more.

these trust relationships are.

Recall from §2.3 that we say that "key sharing" has occurred when any of the parties named in a certificate are not in the same organization that owns the IP addresses from which the certificate is advertised. Because we cannot always infer the nature of such sharing—in rare cases, it may be possible that the website has sole physical access to servers in another's network—we can only definitively detect *potential* instances of key sharing. Whatever the intent and means of sharing, the more private keys a hosting provider has, the more enticing an attack target it becomes.

5.1 How many organizations share keys?

We begin our analysis by investigating how common it is for organizations to share their private cryptographic keys with third-party hosting providers. Ideally, this would not be happening *at all*.

As an initial view, Figure 4 shows the number of *distinct* hosting providers that serve a given certificate (and therefore have the corresponding private key). Surprisingly, we find that 8.8% (363,043) of all observed certificates are hosted by more than one hosting provider—for these certificates, there is certainly at least one third party with access to an organization's private key. In some instances, the keys are shared among *hundreds* of third parties. There is a significant outlier worth noting: Google operates a "global cache" that provides content over HTTPS from servers located in thousands of partner networks throughout the world [18].

To understand how many of these private keys are hosted specifically by companies other than the organizations themselves, Figure 5 shows the distribution of the number of third-party hosting providers that serve a given certificate in our dataset. This shows that ***76.5% of all identified organizations share at least one private key with a third-party hosting provider***. A majority, 62.9%, share with a single third party, but many organizations share one or more of their keys with tens to thousands. Again, the outlier here is Google, which hosts HTTPS data as part of the Google global cache.

We also show in Figure 5 how many specific domains' private keys are hosted by third parties. Interestingly, the keys for some domains are shared across thousands of distinct third-party hosting providers. The two lines in Figure 5 track closely (with the number of domains slightly skewed to the left); this is because the vast majority (89.9%) of organizations on the web have a single domain name.

These results show that sharing secret keys is not an obscure act; rather, it is typical behavior. Also, many domains share their secret keys with multiple hosting providers. To better understand *why* key sharing takes place, we next look at how often sharing happens as a function of website popularity.

Website popularity Figure 6 shows the fraction of Alexa top-1M domains that share (a) at least one and (b) all of their private keys with a third-party hosting provider. We make two important observations from this figure. First, key sharing is prevalent across the full spectrum of website popularity, on average. Second, we observe an interesting, non-linear relationship between popularity and the likelihood of key sharing; the most popular and least popular websites are both more likely to share their keys. For example, 22.7% of the Alexa top-10K share *all* of their private keys. We expect this is driven by two distinct factors: popular websites have incentive to host their content on globally distributed CDNs for better availability and performance, while unpopular websites are likely to use hosting providers rather than manage their own servers.

These results indicate that ***economic incentives are a main driving force for key sharing on the web***. Because they are able to benefit from economies of scale, CDNs attract popular organizations, and hosting providers attract those seeking low-cost alternatives to running a web server.

5.2 How many keys do providers have?

A natural ramification of hosting providers exhibiting economies of scale is that a relatively small number of host-

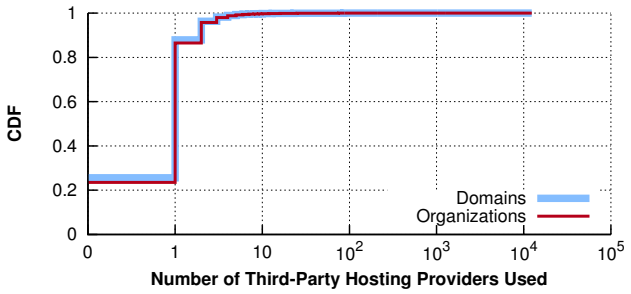


Figure 5: The prevalence of key sharing. Only 23.5% of organizations strictly host their own HTTPS content; the other 76.5% share their private keys, some with *thousands* of distinct third-party hosting providers.

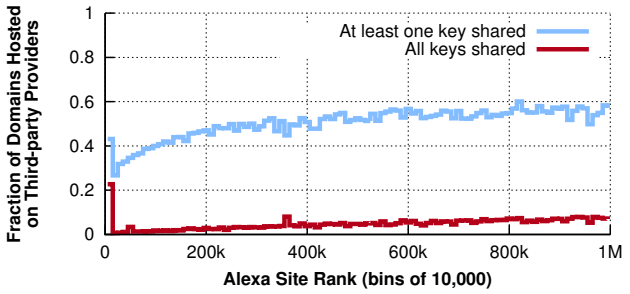


Figure 6: Key sharing as a function of website popularity. The most popular and most unpopular websites tend to share their keys with more third parties. 43.8% of all domains share at least one private key with a third party; 7.7% share all of their keys.

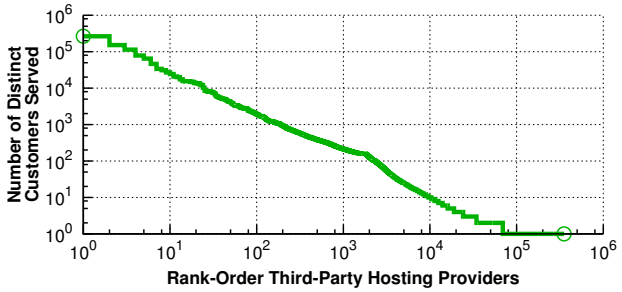


Figure 7: How many organizations' private keys third-party hosting providers have aggregated. Some providers have access to the private keys of *thousands* of organizations' private keys. (The circles highlight the beginning and end of the distribution.)

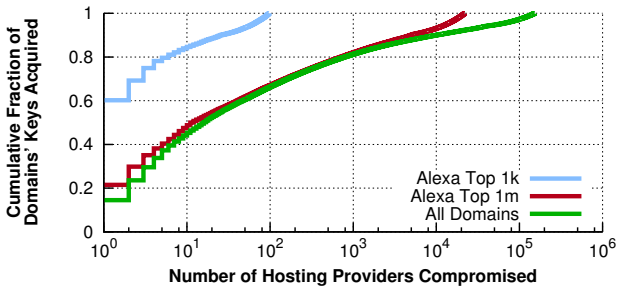


Figure 8: The distribution of the coverage of distinct *domains'* keys across third-party hosting providers. Were an attacker to compromise 10 hosting providers of its choosing, it would gain access to keys from over 45% of *all* domains in our dataset.

ing providers are likely to aggregate a disproportionate number of keys. We next turn to analyzing the extent to which keys have been aggregated.

Figure 7 shows, for each hosting provider in our dataset, the number of organizations for which it has at least one of their private keys. Each point represents a hosting provider, sorted in decreasing order of the number of distinct organizations' HTTPS traffic it serves. These results reveal heavy aggregation of keys among a relatively small set of hosting providers; ***many hosting providers have aggregated tens of thousands of their customers' keys***.

To explore how much of a potential threat key aggregation poses to the HTTPS ecosystem, we show in Figure 8 the number of distinct hosting providers an attacker would have to compromise (x) in order to obtain at least one key from y fraction of all observed domains. To compute this, for each value x, we computed the set of x hosting providers that maximized the coverage of certificates; that is, we are simulating an attacker with perfect knowledge and ability to compromise the x hosting providers who will collectively yield the most keys. We break this down by the Alexa top-1K, top-1M, and all domains. For example, compromising a *single* hosting provider would potentially gain access to 60.2% of the domains' keys from the Alexa top-1K. Across all domains, compromising 10 hosting providers would potentially reveal 45.3% of all observed domains' private keys. Of course, this analysis makes the simplifying assumption that a successful attack on a hosting provider reveals *all* of its customers' keys; we do not expect this to be likely for any but an extremely powerful attacker, but it concretely highlights the extent to which a small number of providers have aggregated a large coverage of private keys.

We conclude from this analysis that private keys from leaf certificates have been widely disseminated across different third-party hosting providers, and many providers have aggregated many of the most popular websites' keys. Unfortunately, today's protocols require hosting providers to have their customers' keys in order to serve their HTTPS content. Mitigating this is an important area of future work (§8).

5.3 How are SAN lists used?

The *spirit* of `SAN` lists is to allow an organization to use a single certificate for their many names (Google's `SAN` list, for example, includes `google.co.uk`, `google.pl`, and so on). However, recall from §2 that some hosting providers lump *different organizations* together in the `SAN` list of a single "cruise-liner" certificate.

Cruise-liner certificates allow providers to serve many customers with a single HTTPS certificate (and IP address), but they have ramifications on key sharing and management. Who on a cruise-liner certificate deserves access to the certificate's corresponding private key, given that whoever has it can impersonate all others on the certificate? Who among them has the right to *revoke* the certificate, if so doing potentially renders invalid a certificate the others rely on? Cruise-liner certificates are not covered explicitly by X.509, but we can infer that, in all likelihood, only the hosting provider has the private keys and right to revoke. Thus, to fully understand who has access to (and management over) private keys, we explore here the prevalence of cruise-liner certificates, and how commonly CAs issue them.

Table 3 presents various measures of the prevalence of `SAN` lists and who hosts and issues them; we discuss each row in turn. Only 4.0% of the certificates in our dataset do not have any `SAN` list, while 92.8% have a `SAN` list that comprises a single organization, which matches the organization in the `Common Name` field; these are the expected uses of `SAN` lists. The remaining certificates, 3.2% (161,810 in total), contain a `SAN` list that comprises one or more distinct organizations that differ from the organization in the `Common Name` field. While this is a relatively small percentage of certificates, it constitutes a far larger percentage of all domains (11.3%)

	...one organization, no `SAN` list	*...one organization, with `SAN` list*	*...multiple organizations (with `SAN` list)*
#Total certificates with...	203,394	4,692,393	161,810
#Domains on certificates with...	124,746	2,265,090	305,904
#Organizations on certificates with...	109,994	1,994,279	255,901
#Third-party services hosting certificates with...	22,346	329,577	15,143
#CAs that issue certificates with...	536	662	266

Table 3: Prevalence of different `SAN` list policies. The *spirit* of a `SAN` list is that it contain multiple domain names belonging to a single organization. In practice, some hosting providers lump together multiple organizations onto a single certificate.

and all organizations (10.8%) in our dataset.

The final two rows of Table 3 pertain to who issues and who uses "cruise-liner" certificates. We find a surprising number of third-party hosting providers that serve them (15,143) and certificate authorities that issue them (266). This latter number is particularly worrisome, as it indicates that there are many CAs that will issue a certificate to those who are not in reality who the certificate claims to be. This is facilitated by the use of `DV` (domain validation) certificates, whose vetting process includes only being able to demonstrate the ability to control content hosted on a particular domain. (In fact, one of the criticisms of cruise-liner certificates is that they cannot be EV [23].)

These results show that "cruise-liner" certificates—though arguably a violation of the *spirit* of X.509 `SAN` lists—have become a common means of sharing keys; they are widely used by hosting providers, and are widely supported by CAs.

5.4 Summary

In this section, we have shown that websites commonly share their private keys with third-party hosting providers, that a small set of hosting providers have aggregated a huge proportion of others' keys, and that cruise-liner certificates are a common means of key sharing. These findings collectively reveal an immense amount of trust that websites place in hosting providers. Our results complement the large body of work on the aggregation of trust among a small number of *certificate authorities* in the web's PKI. Like with CAs, users must adopt trust in hosting providers tacitly and often unknowingly—unlike with CAs, however, the trust relationships between websites and hosting providers is not explicitly stated in certificates, and required us to develop new techniques to detect (§4).

6. CERTIFICATE MANAGEMENT

The previous section showed that key sharing is widespread in the web's PKI, and that a small number of hosting providers have consolidated many organizations' private keys. We now turn our investigation to the effects that this trust has on *certificate management*, particularly as it pertains to revoking and reissuing compromised certificates (we describe best practices in §6.3). We seek to understand who tends to be responsible for performing such management tasks—the organizations listed in certificates or the hosting providers serving it—and whether outsourcing leads to better or worse certificate management practices.

6.1 Determining who manages a certificate

To evaluate certificate management practices at scale, we must first be able to determine who manages the certificates: is it the organization(s) on the certificates or the hosting provider serving the certificate? Determining who is revoking or reissuing a certificate is nontrivial: revocations and reissues do not express who exactly requested them (after all, the PKI was designed on the premise that the entity listed on the certificate is the sole owner of the secret key). Liang et al. [23] studied three CDNs' revocation behavior by signing up as customers, requesting a revocation, and observing it directly, but we seek to understand the broad ecosystem, and this manual process does not scale to our data.

Our insight is that hosting providers who manage their customers' certificates are responsible for obtaining many new certificates, and would therefore, out of convenience, likely gravitate towards a small set of certificate authorities when obtaining certificates. We apply this insight in two steps. First, we generate the distribution, across all issuing certificates, of the number of leaf certificates they were used to sign. As has been reported previously [15], there is a bias towards a small set of CAs, so this distribution is far from uniform—across the *entire* population of certificates, we find that approximately 76% of all leaf certificates are signed with the keys corresponding to merely 1% of all issuing certificates. Therefore, when the population of users (mostly) obtains their own certificates, we anticipate that they will follow a similar distribution. However, when a hosting provider manages certificates on its customers' behalf, our insight dictates that the distribution will be skewed even *more* heavily towards a small set of issuing certificates.

Thus, in our second step, for each hosting provider h, we compute the fraction c of h's certificates that are signed using its single most commonly appearing issuing certificate. If this one issuing certificate is responsible for at least half of all of the leaf certificates h hosts ($c \geq 50\%$)—an extremely skewed distribution—then we conclude that the hosting provider likely manages certificates on behalf of its customers. Otherwise, we deduce that its customers manage certificates for themselves.

We find this to work well in practice: domains that we know do not manage their customers' certificates have c's well below our threshold (`amazonaws.com`: 24.3%, `linode.com`: 19.5%), while those we know to manage their customers' certificates—especially those who host "cruise-liner" certificates—are typically well above the threshold (`cloudflare.com`: 95.5%, `incapsula.com`: 65.1%).

One of the limitations of this technique is that it does not capture the fact that some hosting providers appear to offer a mix of administration practices. Akamai and CloudFlare [10], for example, manage some of their customers' certificates, but also allow them to manage their own. Various web hosting providers offer, at a higher price, to manage their customers' certificates for them.

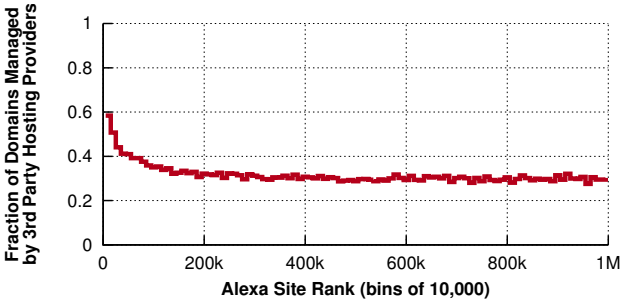


Figure 9: The prevalence of outsourcing certificate management as a function of Alexa ranking. More popular websites are more inclined to host at least one of their certificates on a hosting provider that manages certificates for its customers.

6.2 Prevalence of outsourcing

Figure 9 shows the fraction of domains, as a function of Alexa ranking, that have at least one certificate hosted by a service provider that, according to our technique, manages certificates on behalf of its customers. These results show that ***third-party management is common*** across all lev-

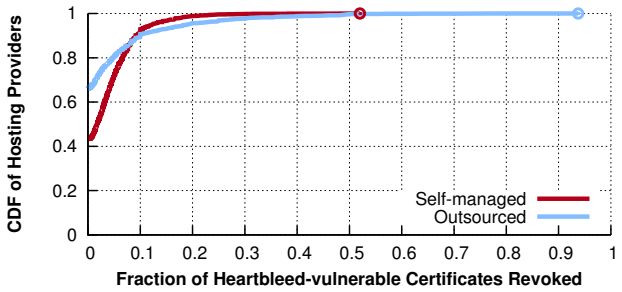


Figure 10: The distribution of how thoroughly Heartbleed-vulnerable certificates were revoked, broken down by those hosted at providers that managed their customers' certificates ("Outsourced") and those whose customers managed their own certificates ("Self-managed").

els of popularity; over 33% of all domains have their HTTPS certificates managed by another organization. But, surprisingly, ***popular websites outsource certificate management more often***. We had anticipated that, while they may use third-party hosting providers, popular websites would still maintain control over certificate management. Because such important websites make use of third-party management, we strive to answer in this section: *are third-parties managing certificates well?*

6.3 Certificate revocation and reissue

Certificates require some degree of manual management. Occasionally, certificates expire and must be reissued; and in the event of a key compromise, a responsible administrator revokes the compromised certificate and reissues a new certificate with a new key. Ideally, revocations and reissues would take place as soon possible after a vulnerability were announced. To evaluate whether this held in practice, Zhang et al. [35] used the Heartbleed [20] vulnerability as a "natural experiment": on April 7, 2014, a large population (122,832 certificates, by their count) realized they were vulnerable all at once, and had to immediately take steps to patch, reissue, and revoke. They found that administrators were slow and incomplete when revoking and reissue certificates.

Other studies have shown that, while most administrators correctly patch their servers, relatively few obtained new certificates with new keys, and even fewer properly revoked their certificates [14, 34, 35].

In this section, we ask a complementary question that, to our knowledge, has not been investigated in prior studies of the SSL ecosystem: *what impact does outsourcing certificate management have on the speed and thoroughness of proper administration?*

6.3.1 Revocation rates

We begin by investigating whether third-party hosting providers revoke certificates more quickly and thoroughly than customers who manage their own certificates. To perform this analysis, we follow the same methodology as Zhang et al. [35], and use the Hearbleed vulnerability as a sort of "natural experiment." Their publicly available datasets provide the set of certificates that were vulnerable to Heartbleed—and thus should have been revoked and reissued—as well as the time that the certificates were revoked and reissued (if they ever were).

Figure 10 shows the distribution of revocation rates for self-managed versus third-party-managed certificates. This result shows that the quality of outsourced management is far more skewed than self-management. The median revocation rate after Heartbleed across all self-managing websites 1.7%, while 66% of hosting providers who managed their customers' certificates *did not revoke a single certificate*! However, some third-party managing providers revoked 94% of their Heartbleed-vulnerable certificates, while the best self-managing websites in our dataset revoked only 52%. This indicates that a small set of third-party hosting providers have highly thorough administrators, but most exhibit poor management behavior. Next we see how this translates to overall certificate revocation rates, and whether third parties react more quickly to vulnerability announcements.

Figure 11 shows a "survivability" plot, with the fraction of Heartbleed-vulnerable certificates that had *not yet* been revoked in the month after Heartbleed was announced. We show three lines: the "Combined" line corresponds to all certificates, independent of hosting provider, and is precisely the line reported by Zhang et al. [35]. The other lines correspond to the certificates served by hosting providers who perform management for their customers, and those who do not.

This figure shows that ***outsourced management is slightly slower to react, but is ultimately more thorough***. We see that it was not until approximately eight days after the Heartbleed announcement when third-party-managed certificates saw a greater revocation rate than self-

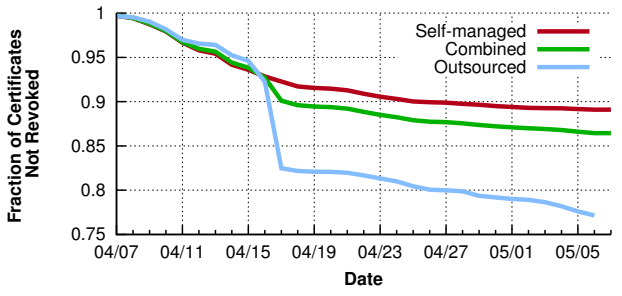


Figure 11: Revocation rates of the Heartbleed-vulnerable certificates in the month after the Heartbleed announcement. Third-parties managing certificates were slower to revoke, but were ultimately more thorough.

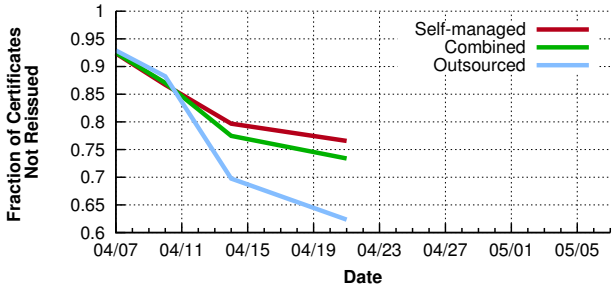
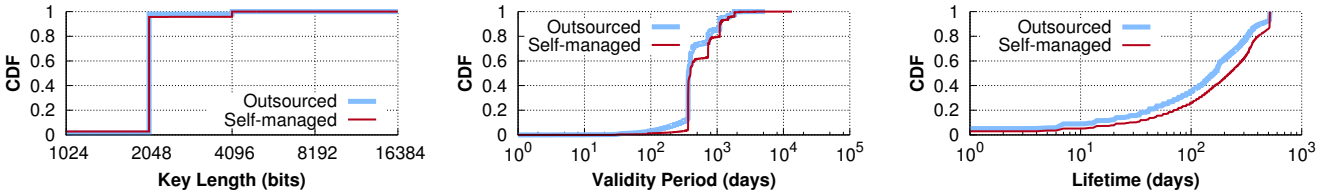


Figure 12: Reissue rates of the Heartbleed-vulnerable certificates in the month after the Heartbleed announcement. Certificates across all types of management tended to be reissued days faster than they were revoked. Like with Figure 11, outsourced management led to slower but more thorough reissue behavior.



(a) Key lengths. **(b)** Validity period (`NotAfter` – `NotBefore`). **(c)** Lifetime (First seen – Last seen).

Figure 13: Outsourced and self-managed certificates differ slightly with respect to core certificate features. In general, self-managed certificates tend to have worse security, in terms of shorter keys and longer validity periods.

managed ones. After that time, self-managed certificates never caught back up. We believe this interesting behavior can be attributed to the fact that a single hosting provider can be responsible for thousands of private keys (§4.2), and yet there is typically not a large team dedicated to revoking and reissuing certificates. As a result, the delay of even a single administrator at a hosting provider that manages certificates can result in delayed revocations of many distinct organizations' certificates.

6.3.2 *Reissue rates*

In addition to revoking compromised certificates, website administrators must also reissue new certificates (with new keys) to replace the old ones. Zhang et al. [35] showed that, across the entire corpus of certificates vulnerable to Heartbleed, a mere 27% had reissued within three weeks of Heartbleed's public announcement. Like with revocations, we seek to understand how these reissue rates differentiate between self- and third-party-managed certificates.

Figure 12 shows the survivability plot for reissuing (as in Figure 11). We note that this plot stops significantly sooner than the revocation plot as it is based upon the dataset from Zhang et al. [35], which only spans the first two weeks after the vulnerability was announced.

Similar to post-Heartbleed revocations, third-party management delayed reissues slightly, but caught up far more quickly; they reissued extensively one week after Heartbleed, but did not revoke extensively until 10 days after Heartbleed. Zhang et al. [35] showed that there is often a delay of several days between reissue and revocation, but the ∼3 week difference for third-party certificate management is abnormally high. While the origins of this additional delay are not easily identified, delays between reissuing and revoking may be due to factors such as lack of support for bulk revocations needed by third-parties.

6.4 Certificate quality

Prior studies of the SSL ecosystem [1,15,19,34] have shown that many administrators choose their keys poorly. Here, we expand upon these prior findings by evaluating whether there is a correlation between centralized management and the quality of the keys chosen.

Figure 13 compares several different features of self-managed and outsourced certificates across our entire corpus of leaf certificates (3,275,635 self-managed and 1,781,962 outsourced): (a) Key lengths in self-managed certificates are nearly identical to those managed by third-party hosting providers. We observe a slightly greater fraction of 1024-bit keys among self-managed certificates and a slightly lesser fraction of 2048-bit keys. Though this difference is slight, the number of samples we observe (3.3M self-managed and 1.8M outsourced certificates), we believe it to be statistically significant. (b) Every certificate defines a date at which it should start (`NotBefore`) and finish (`NotAfter`) being considered valid. A certificate's validity period—the time between these dates—is ideally short so that, in the event of a key compromise, if the administrator fails to revoke the certificate, it will expire more quickly; conversely, Zhang et al. [35] demonstrated that many non-revoked Heartbleed-vulnerable certificates were not set to expire for upwards of six years after Heartbleed was announced. Outsourced certificates exhibit the better security practice of shorter validity periods than their self-managed counterparts. (c) Finally, the lifetime of a certificate (defined by the number of days between the first and last certificate scans in which we saw the certificate) is not directly a measure of security, but it may reflect administrators who are more active in replacing their compromised or expiring certificates. Here, too, outsourced certificates exhibit lower lifetimes, and therefore likely better security practices, than self-managed ones.

In general, these results indicate that revoking and reissuing certificates are not the only acts that third-party hosting providers tend to do better; they also tend to create better certificates, as well.

Another measure of certificate quality pertains to the type of validation the issuing CA used when vetting the subject. The strongest form of validation, Extended Validation (EV), is of particular note, because browsers treat them differently, showing more information about them in address bars and even taking greater care in checking for revocations of EV certificates over other forms of validation [25]. We would therefore expect that there would be a trend towards better security practices when considering EV certificates. Interestingly, we find that, of all EV certificates in our dataset, 99.1% are *hosted* by a third party, and 27.6% are *managed* by that third party. Moreover, 2.7% of all third-party-managed certificates are EV, while 3.8% of all self-managed certificates are. Although EV certificates do not, in and of themselves, offer better security, we find it surprising that even these more extensively vetted certificates are shared with third parties who manage certificates for their customers.

6.5 Summary

This section presented a technique for determining who manages a given certificate—the domain owner or a third-party hosting provider—and applied it to a dataset of vulnerable certificates to measure the effect that outsourcing has on certificate management. Surprisingly, while sharing private keys with a third party is a clear violation of the semantics and security properties of online authentication, in

practice, overall certificate management *improves* with outsourcing. The results from this section and §5 paint a complicated picture of the web's PKI: the economics of hosting an online secure service can lead to centralization that is both dangerous and, in some ways, helpful. One important area of future work is to find a way to reconcile good centralized management with poor centralized trust aggregation.

7. RELATED WORK

HTTPS ecosystem Much work has gone into understanding and improving the SSL certificate ecosystem, including measurements of CAs, certificates they issued, and client root stores [15,16,19,28,33], techniques for improving the transparency and accountability of CAs [21], measurements of the cost of HTTPS security [13], and alternate architectures to the current CA-based systems [2,11,31]. Our work complements these, as most are focused on understanding the centralization of the set of CAs and the properties of client root stores. In contrast, we primarily focus on the *leaf* certificates, and how the sharing of private keys for these certificates can lead to a similar centralization of trust.

Certificate reissues/revocations Several closely related papers [14, 25, 34, 35] have explored the patterns of reissuing and revoking certificates, using either the Heartbleed vulnerability or the Debian random number generator bug as a way to get visibility into system administrators' behavior. Essentially, these papers use the fact that they can measure whether a server was vulnerable to infer whether the administrator *should* have patched and reissued/revoked their certificate. Our work extends these results, using techniques like the distribution of issuer CAs to get visibility into hosting companies' behavior and policies (i.e., do hosting companies manage certificates, or do the customers?).

CDNs and HTTPS More recent work by Liang et al. has closely examined how CDNs manage SSL certificates when distributing HTTPS content for customers [23]. They observe the distinction between custom and shared certificates, many instances of mismanagement, CAs neglecting to revoke certificates, and private key sharing. Our work extends the work by Liang et al. by developing more thorough methodologies for measuring when keys are shared (by collapsing domains into organizations) and for determining whether the hosting provider or the customers are actually managing the certificates. Taken together, these results point to many places in the SSL ecosystem that are in need of improvement.

Finally, a recent technique by CloudFlare [9] offers the opportunity for organizations to use hosting providers without having to share their private keys (though the CDN still has all of the session keys). While this approach has not yet been widely deployed, it is an interesting first step towards reducing much of the centralization of trust that we observe.

8. CONCLUSION

In this paper, we presented the first large-scale study of key sharing within the HTTPS ecosystem. Surprisingly, although key sharing has profound security implications, we had to develop several novel techniques to infer who owns which domain names, which organization hosts a domain, and who hosts a certificate. We applied these techniques to Internet-wide scans of SSL certificates and identified an immense amount of trust being aggregated among a small set of hosting providers. In particular, we found that 76.5% of all organizations on the web that we identified share at least one private key with a third-party hosting provider, and that a collection of ten hosting providers have private keys to 45.3% of *all* domains we observed in our scans. We also compared how well third-party hosting providers manage their customers' certificates as compared to self-managed certificates, and found that third parties are slower to react to large-scale vulnerabilities but eventually react more thoroughly.

Our findings complement a wide body of work that has studied trust and management in the web's PKI. While centralization of trust has long been known for CA certificates, our results paint the first complete picture of its prevalence for leaf certificates, as well.

Collectively, our results reveal some of the widespread ramifications of deploying today's PKI on third-party hosting providers like CDNs. However, the web has come largely to rely on HTTPS and third-party hosting. Future work is necessary to handle this reality without requiring such immense trust in hosting providers. One improvement would be to make the centralization of private key material more transparent; we were forced to combine a large number of data sources in order to determine who has access to what key material. Looking forward, we posit that new techniques could be developed to enable third-party hosting while mitigating their access to their customers' keys. The recent Keyless SSL [9] protocol is a first step in this direction, allowing operators to retain control of their private keys while allowing third-party hosting providers to accept connections on their behalf. An important open question is whether it is possible eliminate CDNs' need to have session keys, as well, while still being able to assist in preventing various attacks [17].

Our analysis code and data, including our classifications of domains into organizations (§4), are publicly available at `https://securepki.org`

Acknowledgments

We thank the anonymous reviewers for their helpful comments. We also thank Suqi Liu and his co-authors [24] for sharing their dataset of parsed `.com` WHOIS records. This research was supported by NSF grants CNS-1409249, CNS-1421444, CNS-1563320, and CNS-1564143, and by the NSA as part of a Science of Security lablet.

9. REFERENCES

[1] D. Akhawe, B. Amann, M. Vallentin, and R. Sommer. Here's My Cert, So Trust Me, Maybe?: Understanding TLS Errors on the Web. *WWW*, 2013.

[2] A. Bates, J. Pletcher, T. Nichols, B. Hollembaek, and K. R.B. Butler. Forced Perspectives: Evaluating an SSL Trust Enhancement at Scale. *IMC*, 2014.

[3] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebre. Fast unfolding of community hierarchies in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 10(10), 2008.

[4] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, IETF, 2008. `http://www.ietf.org/rfc/rfc5280.txt`.

[5] K. Chen, D. Choffnes, R. Potharaju, Y. Chen, F. Bustamante, D. Pei, and Y. Zhao. Where the Sidewalk Ends: Extending the Internet as Graph Using Traceroutes from P2P Users. *IEEE ToC*, 4(63), 2014.

[6] T. Chung, Y. Liu, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson. Measuring and Applying Invalid SSL Certificates: The Silent Majority. *IMC*, 2016.

[7] CAIDA Routeviews Prefix to AS Mappings Dataset. `http://www.caida.org/data/routing/routeviews-prefix2as.xml`.

[8] CAIDA AS Organizations Dataset. `http://www.caida.org/data/as-organizations/`.

[9] CloudFlare Keyless SSL. `https://blog.cloudflare.com/keyless-ssl-the-nitty-gritty-technical-details/`.

[10] CloudFlare support: How do I upload a custom SSL certificate? `https://support.cloudflare.com/hc/en-us/articles/200170466-How-do-I-upload-a-custom-SSL-certificate-Business-or-Enterprise-only-`.

[11] Convergence. `http://convergence.io`.

[12] L. Daigle. WHOIS Protocol Specification. RFC 3912, IETF, 2004. `http://www.ietf.org/rfc/rfc3912.txt`.

[13] N. David, F. Alessandro, L. Ilias, G. Yan, M. Marco, M. Maurizio, P. Konstantina, and S. Peter. The cost of the S in HTTPS. *CoNEXT*, 2014.

[14] Z. Durumeric, J. Kasten, D. Adrian, J. A. Halderman, M. Bailey, F. Li, N. Weaver, J. Amann, J. Beekman, M. Payer, and V. Paxson. The Matter of Heartbleed. *IMC*, 2014.

[15] Z. Durumeric, J. Kasten, M. Bailey, and J. A. Halderman. Analysis of the HTTPS Certificate Ecosystem. *IMC*, 2013.

[16] EFF SSL Observatory. `https://www.eff.org/observatory`.

[17] D. Gillman, Y. Lin, B. Maggs, and R. K. Sitaraman. Protecting Websites from Attack with Secure Delivery Networks. *Computer*, 48(4), IEEE, 2015.

[18] Google Global Cache. `https://peering.google.com/#/infrastructure`.

[19] 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. *IMC*, 2011.

[20] Heartbleed Bug. `http://heartbleed.com`.

[21] B. Laurie, A. Langley, and E. Kasper. Certificate Transparency. RFC 6962, IETF, 2013. `http://www.ietf.org/rfc/rfc6962.txt`.

[22] G. Lord. Secure CDN: new certificate options now available. Akamai blog, 2015. `https://community.akamai.com/community/whatsnew/blog/2016/02/05/new-secure-cdn-offerings-now-available`.

[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. *IEEE S&P*, 2014.

[24] S. Liu, I. Foster, S. Savage, G. M. Voelker, and L. K. Saul. Who is. com? Learning to Parse WHOIS Records. *IMC*, 2015.

[25] Y. Liu, W. Tome, L. Zhang, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, A. Schulman, and C. Wilson. An End-to-End Measurement of Certificate Revocation in the Web's PKI. *IMC*, 2015.

[26] List of Autonomous Systems. `http://www.cidr-report.org/as2.0/autnums.html`.

[27] OS X Yosemite: List of available trusted root certificates. `https://support.apple.com/en-us/HT202858`.

[28] H. Perl, S. Fahl, and M. Smith. You Won't Be Needing These Any More: On Removing Unused Certificates from Trust Stores. *FC*, 2014.

[29] Rapid7 Reverse DNS Scans. `https://scans.io/study/sonar.rdns`.

[30] Rapid7 SSL Certificate Scans. `https://scans.io/study/sonar.ssl`.

[31] The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. RFC 6698, IETF, 2012. `https://tools.ietf.org/html/rfc6698`.

[32] B. VanderSloot, J. Amann, M. Bernhard, Z. Durumeric, M. Bailey, and J. A. Halderman. Towards a Complete View of the Certificate Ecosystem. *IMC*, 2016.

[33] N. Vallina-Rodriguez, J. Amann, C. Kreibich, N. Weaver, and V. Paxson. A Tangled Mass: The Android Root Certificate Stores. *CoNEXT*, 2014.

[34] S. Yilek, E. Rescorla, H. Shacham, B. Enright, and S. Savage. When Private Keys Are Public: Results from the 2008 Debian OpenSSL Vulnerability. *IMC*, 2009.

[35] L. Zhang, D. Choffnes, T. Dumitraş, D. Levin, A. Mislove, A. Schulman, and C. Wilson. Analysis of SSL certificate reissues and revocations in the wake of Heartbleed. *IMC*, 2014.